KOCAELİ ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

LİSANS TEZİ

HOBİ BAHÇESİ WEB UYGULAMASI

MUHAMMED KOÇ

KOCAELİ 2024

KOCAELİ ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

LİSANS TEZİ

HOBİ BAHÇESİ WEB UYGULAMASI

MUHAMMED KOÇ

Dr. Öğr. Üyesi Hikmetcan ÖZCAN .....
Danışman, Kocaeli Üniv.

Doç.Dr. Sevinç İLHAN OMURCA .....
Jüri Üyesi, Kocaeli Üniv.

Dr. Öğr. Üyesi Orhan AKBULUT .....
Jüri Üyesi, Kocaeli Üniv.

Tezin Savunulduğu Tarih: 05.01.2024

ÖNSÖZ VE TEŞEKKÜR

Bu tez çalışması kullanıcılara internet üzerinden hobi bahçesi kiralama ve ek hizmet temini sağlamak amacıyla gerçekleştirilmiştir.

Tez çalışmamızda desteğini esirgemeyen, çalışmalarımıza yön veren, bize güvenen ve yüreklendiren danışmanımız Hikmetcan Özcan'a sonsuz teşekkürlerimizi sunarız.

Hayatımız boyunca bize güç veren en büyük destekçilerimiz, her aşamada sıkıntılarımızı ve mutluluklarımızı paylaşan sevgili ailelerimize teşekkürlerimizi sunarız.

Ocak - 2024 Muhammed KOÇ

Bu dokümandaki tüm bilgiler, etik ve akademik kurallar çerçevesinde elde edilip sunulmuştur. Ayrıca yine bu kurallar çerçevesinde kendime ait olmayan ve kendimin üretmediği ve başka kaynaklardan elde edilen bilgiler ve materyaller (text, resim, şekil, tablo vb.) gerekli şekilde referans edilmiş ve dokümanda belirtilmiştir.

Öğrenci No: 190202040

Adı Soyadı: Muhammed KOÇ

İmza:……..

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

SİMGELER VE KISALTMALAR DİZİNİ

SQL : Structured Query Language
DB : Database
MVC : Model-View-Controller
MSSQL : Microsoft SQL
EF Core : Entity Framework Core
API : Application Programming Interface
LINQ : Language Integrated Query

ÖZET

Bu çalışmanın amacı müşterilerin kendi ihtiyaçlarına, hobilerine yönelik olarak düzenleyebileceği küçük ölçekli bahçelerin yani hobi bahçelerinin internet üzerinden kiralanması ve ilgili hizmetlerin sağlanmasıdır. Hobi bahçesi sektörü günümüz dünyasında şehirleşmenin artması sebebiyle insanların tarımdan ve bahçe işlerinden uzaklaşmasının ardından bu ihtiyaca cevap vermesi sebebiyle giderek popüler hale gelmektedir. Buna karşın bu ihtiyacı karşılaması gereken hizmetler henüz yeterli seviyede değildir. Piyasada hobi bahçesi kiralamak için kullanılan web siteleri incelenmiş, yetersiz bulunmuş ve oradaki eksikler dikkate alınarak projedeki sistem tasarımı üzerinde planlamalar yapılmıştır. Projemiz bu konudaki ulaşılabilirlik problemini çözmek için kullanımı ve erişimi kolay olan kendi web sitesi üzerinden işlemleri yürütecektir.

Planlanan uygulama için MVC tasarım deseni tercih edilmiştir ve seçtiğimiz tasarım desenine uygun şekilde uygulama geliştirme imkanı sunması, güncelliği ve stabilitesi sebebiyle arka yüzde çalışmak üzere .NET Core 7. versiyonu kullanılmıştır. Geliştiricilerin deneyimi sebebiyle veritabanı olarak MsSql ve ilgili veritabanı işlemlerini kolaylaştırmak amacıyla EFCore (Entity Framework Core) kullanılmıştır. Uygulamanın ekranları Figma üzerinden tasarlanmıştır. Ekranlar JavaScript kütüphaneleri ve bootstrap kullanılarak geliştirilmektedir. Web sitesi üzerinden farklı arsalar hobi bahçesi kullanımına açılacak olup bu arsalar uyguladığımız alan bölme yöntemi sayesinde önceden belirlenmiş boyutlarda eşit olarak bölünecektir.

Anahtar Kelimeler: Hobi Bahçesi, E-ticaret, EfCore, .NET Core, MVC, MSSQL, Alan Eşit Parçalara Ayırma

ABSTRACT

The aim of this study is to rent small-scale gardens that customers can organize for their own needs and hobbies, i.e. community gardens, over the internet and to provide related services. The hobby garden sector is becoming increasingly popular in today's world because it meets this need after people move away from agriculture and gardening due to increasing urbanization. However, the services that should meet this need are not yet sufficient. The websites used to rent community gardens in the market were examined, found to be inadequate and plans were made on the system design in the project based on the deficiencies there. Our project will carry out transactions through its own website, which is easy to use and access, to solve the accessibility problem in this regard.

MVC design pattern was preferred for the planned application, and .NET Core version 7 was used to work on the backend due to its up-to-dateness and stability and the possibility of developing applications in accordance with the design pattern we chose. Due to the experience of the developers, MsSql was used as the database and EFCore (Entity Framework Core) was used to facilitate the related database operations. The screens of the application were designed through Figma. Screens are developed using JavaScript libraries and bootstrap. Different plots of land will be opened for community garden use through the website, and these plots will be divided equally in predetermined sizes thanks to the area division method we apply.

Keywords: Community Garden, E-commerce, Hobby Garden, EFCore, .NET Core, MVC, MSSQL, Equal Division of Polygon

GİRİŞ

İnsanlık geçtiğimiz yüzyıl boyunca köy hayatından uzaklaşarak topraktan uzaklaşmaya başlamıştır ve bu durum günümüzde de devam etmektedir. Buna karşın insanların toprakla uğraşma, kendi bahçesinde ürettiği işlem görmemiş ürünleri tüketme isteği hala mevcuttur. Bu isteğin giderek artması sebebiyle hobi bahçeleri günden güne daha çok talep görmektedir.

Hobi bahçeleri genellikle kentlerde yaşayan insanların sebze, meyve ve süs bitkisi yetiştirebildiği küçük ölçekli arazilerdir. Genellikle insanlar bu alanları doğayla iç içe olmak, toprakla etkileşime girmek ve şehir hayatından uzaklaşmak amacıyla tercih eder. Kullanıcı kendisine tahsis edilen alana, arazinin bulunduğu çevresel faktörleri de göz önüne alarak, tercihlerine göre farklı türlerde ekim yapabilir.

Hobi bahçelerinin yürütülmesi genellikle kiralama sistemi ile çalışır. Hem ülkemizde hem de dünyada yerel yönetimlerin bu hizmeti sağladığı görülmektedir. Yerel yönetimler belirli bir süreliğine kullanıcılarına arazi içerisinden uygun bir parseli tahsis eder. Yönetim bahçe kullanıcıları tarafından oluşturulan bir dernek ile sağlanır. Hobi bahçesi olarak tesis edilecek alanlar var olan tarımı engellemeyecek nispeten daha verimsiz alanlara kurulmalıdır. Bu tercihi yaparken yeşil alan bakımından eksik alanlar seçilebilir. Kullanıcıların çoğunluğu şehirden geldiği için ulaşım bakımından elverişli alanların tercih edilmesi, kullanıcıların tercih etme sebeplerinden biridir.

Hobi bahçesi uygulamalarına 1750'li yıllarda İngiltere'de bulunan Birmingham'da kurulmuş olan "Guinea Gardens" olarak bilinen alanlarda başlanmıştır. Aslen bu bahçeler yardım yapmak amacıyla kurulmuş, bu yolla az gelirli işçilere destek sağlanmıştır. Günümüzde İngiltere'de 77000 bahçe bulunmaktadır. Bu süreci daha sonra dünyada farklı tiplerde birçok ülkede de hobi bahçeleri oluşturulması takip etmiştir. Günümüzde Almanya'da 1 milyondan fazla hobi bahçesi mevcuttur.[1]

Ülkemizde de şehirleşmenin yaygınlaşması ve kırsal arazide yaşayan insanların genellikle kendi bahçelerine sahip olması sebebiyle hobi bahçeciliği 1980'li yıllarda başlamıştır. Nüfusu daha yoğun olan şehirlerden başlayarak ülke geneline yayılmış bir sistemdir. Bursa Küçük Bahçe Tesisleri 1986 yılında ülkemizde kurulmuş olan ilk hobi bahçesidir. Daha sonra 1989 yılında İzmir'de "İzmir Kent Bahçeleri" ilk olmak üzere farklı projeler başlamıştır. Giderek artan bu talebe karşılık olarak birçok farklı bölgede de hobi bahçeleri oluşturulmuştur.[1]

Verimli tarım arazilerinin de hobi bahçesi olarak kullanılması sebebiyle toprağın bozulmasına veya insan faaliyetleri sonucu zarar görmesine sebebiyet vermiştir. Ülkemizde de tarım arazilerinin hobi bahçesi olarak kullanımını yasaklayan Hobi Bahçeleri ve 5403 Sayılı Toprak Koruma Kanunu tarım arazilerinin hobi bahçesi olarak kullanımını yasaklamıştır. Bu kanunla birlikte hobi bahçeleri varlığını sürdürmeye devam edecektir ancak süreçten dolayı oluşabilecek problemlerin önlenmesi amaçlanmıştır.[2]

1. AMAÇ VE HEDEF

Projede bir admin paneli ve user paneli bulunması gerekmektedir. Admin kendine ait olan e-posta ve admin şifresi ile sisteme giriş yapması ve arayüzden tüm müşterilerin bilgilerine ve müşterilerin kiraladıkları haritalara ve o haritalardaki ek hizmetlere ulaşabilmesi gerekmektedir. Müşteriler ise sisteme giriş yaptığında sadece kendilerine ait olan haritaları görüntüleyip ve o haritalara ait olan ek hizmetleri görüntüleyebilmeliler. Burada adminden farklı olarak işlem yapılmaktadır ve her bir müşteri sisteme kendi e-posta ve şifresi ile girmektedir.

Admin sisteme girdikten sonra üzerinde işlem yapmak istediği müşteriyi belirledikten sonra o müşteri için istediği haritanın kullanılabilir olan parsellerinden kiralama yapabilmelidir. Ayrıca seçtiği parsellere istediği ek hizmetleri atayabilmeliler. Bu kiralama ve seçim işlemlerinde sınırlama yoktur ve herhangi bir ücret uygulanmamalıdır.

Kiralama işleminin yanında admin istediği müşterinin parselinin yerini başka bir parselle değiştirebilmelidir ama bu ya o haritada ya başka bir haritada olması fark etmemelidir. Yine bu işlem için ücret uygulanmamalıdır. Ayrıca admin herhangi bir kısıt olmadan istediği müşterinin istediği parselinin ek hizmetlerine değiştirebilmekte yeni ek hizmet ekleyip ek hizmeti kaldırabilmelidir. Admin istediği müşterinin üzerinde bulunan parseli de silebilme yetkisine sahip olmalıdır. Admin işlemlerinde ücretlendirme değerlendirilmemektedir.

Müşteriler ise sisteme giriş yaptıklarında kiralama işlemi için müsait olan haritalardan seçim yaptıktan sonra, bu seçimin tekli veya çoklu olması durumu değiştirmeyecektir, seçilen haritaların üzerine ek hizmetleri ekleyebilmeliler. Parsel ve ek hizmet seçimi yapıldıktan sonra işlemi tamamlama butonuna basıldığında bunlar alışveriş sepetine eklenmelidirler. Daha sonradan alışveriş sepetini tamamlayıp karşılığındaki ücreti ödeyerek sisteme eklenmesi sağlamalıdır. Müşteriler kiralama işleminin yanında daha önceden kiralamış oldukları parsellerin yerine aynı haritada ki müsait olan başka bir parselde veya başka bir haritada müsait olan bir parselde yerlerini taşıyarak haklarını ve ek hizmetlerini o parsele taşıyıp eski parseli bırakabilmeliler. Bunun için alışveriş sepeti artık kullanılmayıp direkt ücretlendirme yapılması gerekmektedir. Aynı zamanda müşteriler daha önceden kiralamış oldukları parsellerdeki ek hizmetlere artırıp azalma haklarına sahiptirler. Bunun için yine bir alışveriş sepeti mantığı kullanılmayıp direkt işleme dökülmelidir.

Sisteme yeni bir harita ve parsel eklenmesi admine bağlı bir işlemdir ve admin panelinde gerçekleşmelidir. Admin girilecek olan haritanın konumlarını sisteme yazıp o alanın kaç parçaya bölünmesini istediğini yazdıktan sonra sistem otomatik olarak o kadar sayıda parseli o alanda oluşturup bunu görselleştirmesi ve müşterilerin satın alabilmesi için uygun hale getirmesi gerekmektedir. Böylelikle sisteme yeni bir harita giriş sağlanmaktadır. Giriş yapan müşteriler yeni haritadan istedikleri gibi parselleri sağlayabilirler burada önemli olan başka bir unsur girilen sayıdaki parselin hepsi birbirine eşit olacak şekilde harita üzerinde yerleştirilmesi ve bunun sağlanmasıdır.

2. HOBİ BAHÇESİ

2.1. Hobi Bahçelerinin Topluma Sağladığı Katkılar

- Sağlık: Kullanıcılar kendi ürünlerini hormonsuz bir şekilde yetiştirip tüketerek daha sağlıklı besinler tüketebilir, bahçeyle ilgilendiği süreçte hem doğa içinde olduğundan şehir hayatına göre daha sağlıklı bir ortamda bulunmuş olur hem de bahçeyle ilgilenirken fiziksel aktivite gerçekleştirir.

- Finans: Hobi bahçelerinin hem kullanıcı hem de arazi sahibi için maddi faydaları vardır. Kullanıcılar kendi ürünlerini yetiştirerek satın aldığı ürün miktarını azaltabilir veya yetiştirdiği ürünlerin satışını gerçekleştirebilir. Arazi sahipleri ise arazisini kiralayarak gelir elde etmektedir.

- Yaşam Becerileri: Kullanıcılar, temel tarım bilgilerinin ek olarak planlama, organizasyon ve ekip çalışması gibi önemli yaşam becerilerini de öğrenirler.

- Daha Yeşil Bir Şehir: Hobi bahçesi yapılan alanlar genellikle verimli kullanılmayan şehir içi alanlardan veya atıl durumdaki arazilerden seçilir. Bu sayede şehir içindeki yeşil alan miktarı artar ve insanların hayat kalitesine olumlu etki sağlar.[3]

2.2. Hobi Bahçelerinin Türleri

- Bölünmüş Bahçeler: Arazinin daha ufak parçalara bölünerek farklı kullanıcılara tahsis edilmesi. Kullanıcılar kendi alanlarında kendi isteklerine uygun olarak farklı ürünler yetiştirir. Bu tarz bahçelerde de bazı parseller ortak üretim için ayrılmış olabilir. Projede üzerinde durulacak olan hobi bahçesi türü bölünmüş bahçelerdir.

- Kooperatif Bahçeleri: Ortak olarak ürün yetiştirilen hobi bahçeleridir. Bölünmüş alanlardan farklı olarak yetiştirilecek üründe kullanıcı bireysel olarak tercihte bulunmaz. Dernekler gibi çok sayıda üyesinin bulunduğu yapılar için uygundur.

- Gençlik Bahçeleri: Genellikle yerel yönetimler veya okul gibi yapılar tarafından genç yaştaki kullanıcıların kullanımına tahsis edilmiş alanlardır. Eğitim veya deney amaçlı kullanılabilir. Öğrencilerin ortak çalışmayı öğrenmesi, tarım bilgisi edinmesi gibi faydaları vardır. Amerika'da çeşitli örnekleri bulunur.

- Terapi Bahçeleri: Bu hobi bahçesi türü genel olarak hastane, bakımevi gibi yapılarda bulunur. Bahçelerin kullanan kitle düşünülerek özel olarak tasarlanması gerekir. Tekerlekli sandalye kullanımına uygun olması, daha az güç ile tarım yapmayı sağlayabilecek araçlar barındırması veya görme engelliler için uygun işaretler barındırması gerekebilir.[3]

3. LİTERATÜR ÇALIŞMASI

Konya'da oluşturulacak yeni bir hobi bahçesi için araştırma yapılmıştır. 2015 yılında yazılmış bu yüksek lisans tezinden çıkarılmış sonuçlara göre kullanıcılar hobi bahçesinin bulunacağı konum için şehre yakın ancak şehrin gürültüsünden de uzak olacak bir mesafede olmasını tercih etmektedir. Çalışmada incelenen 28 hobi bahçesinin şehre en yakınının 2 km, en uzağının 17 km uzaklıkta olduğu saptanmıştır. Bu verilere bakıldığında şehir merkezine olan ortalama uzaklığın 7 km olduğu belirlenmiş ve elde edilen rakam bu çalışma için şehir merkezine uzaklığın değerlendirilmesinde ortalama değer olarak kabul edilmiştir. Buna ek olarak arazi I. sınıf tarım arazisi olmalı ve arazinin %2'den düşük eğime sahip olması gerekmektedir.[4]

2016 yılında Yeni Zelanda'da yapılan bir araştırma hobi bahçelerinin toplumun refahına katkısı olduğunu ortaya koymuştur. Refah, toplumsal memnuniyetin ve nüfusun ilerlemesinin bir göstergesi olarak sağlıkçılar, devlet kurumları ve akademisyenler için giderek daha popüler bir ölçüt haline gelen çok boyutlu bir yapıdır. Refah, hastalığın yokluğundan daha fazlasıdır; dayanıklılık, olumlu duygusal deneyimler ve genel yaşam memnuniyeti ile optimal fiziksel ve zihinsel işlevselliği kapsar. Refahın hobi bahçeleri bağlamında ele alınması önemlidir, çünkü refah hobi bahçelerinin amaçlanan nihai hedefi olmasa da, toplum bahçelerine katılımın sonuçlarının çoğu refahı olumlu yönde etkilemektedir.[5]

2017 yılında Balıkesir'de bulunan hobi bahçeleri üzerine yapılmış bir araştırma hobi bahçelerini kullanan kişiler hakkında genel bir fikir sahibi olma imkanı vermiştir. Bu araştırmaya göre hobi bahçelerini kullanan kişiler ağırlıklı olarak 50 yaş ve üzeri emekli kesimdir. İnsanların genel olarak bu uygulamadan beklentisi maddi olarak gelir elde etmekten ziyade şehirden uzaklaşma imkanına sahip olmak ve kendi ürününü yetiştirmektir. Bu bilgilere ek olarak hobi bahçelerini kullanan kişiler genellikle daha önce tarımsal faaliyetlerde bulunmamış kişilerdir. Ağırlıklı olarak kullanıcılar bahçeleriyle haftalık olarak 3-5 saatlik bir zaman diliminde uğraşmaktadır.[6]

3.1. Poligonu Eşit Parçalara Bölmek

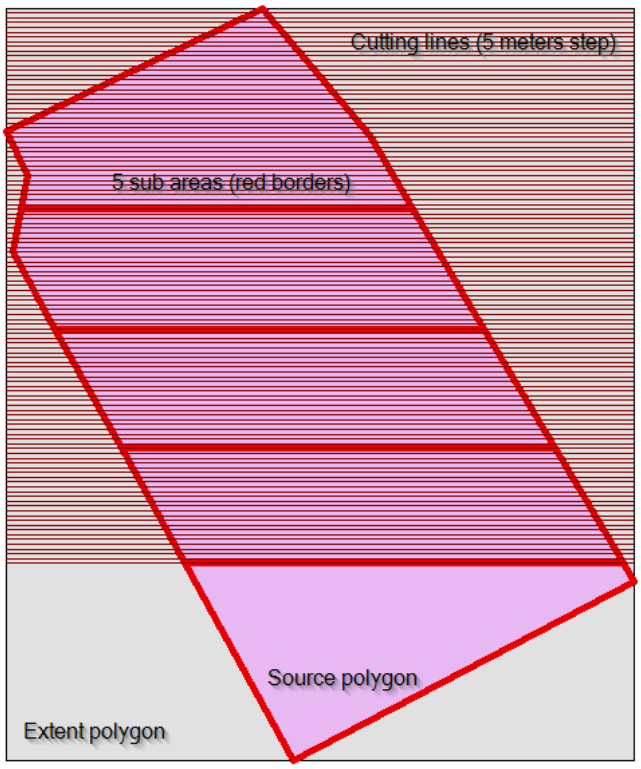
Yaptığımız araştırma çalışmasında karşımıza çıkan problemlerden birisi de arsaların farklı büyüklük ve şekillerde olmasından kaynaklanan eşit parsellere bölme problemidir. Bu problem için literatür araştırması yapılmıştır.

3.1.1. Bir Poligonu Arcpy ile Belirli Sayıda Eşit Alana Bölme

ArcGIS, Esri tarafından geliştirilen ve bakımı yapılan bir istemci, sunucu ve çevrimiçi coğrafi bilgi sistemi yazılımı ailesidir. ArcGIS, "parcel fabric" ile parsel bölünmesini sağlar. Ne yazık ki, parsel yapısını kurmak için gereksinimleri fazla ve parsellerin bölünmesinden önce öğrenilmesi gereken bilgi miktarı fazladır. Bu süreci kolaylaştırmak için sabit bir enlem veya

boylam çizgisi boyunca bir çokgen alanını ikiye bölen veya ikiye bölen bir çizgiyi hesaplayan Polygon Bisector adlı bir ArcGIS özel komut dosyası aracı mevcuttur.

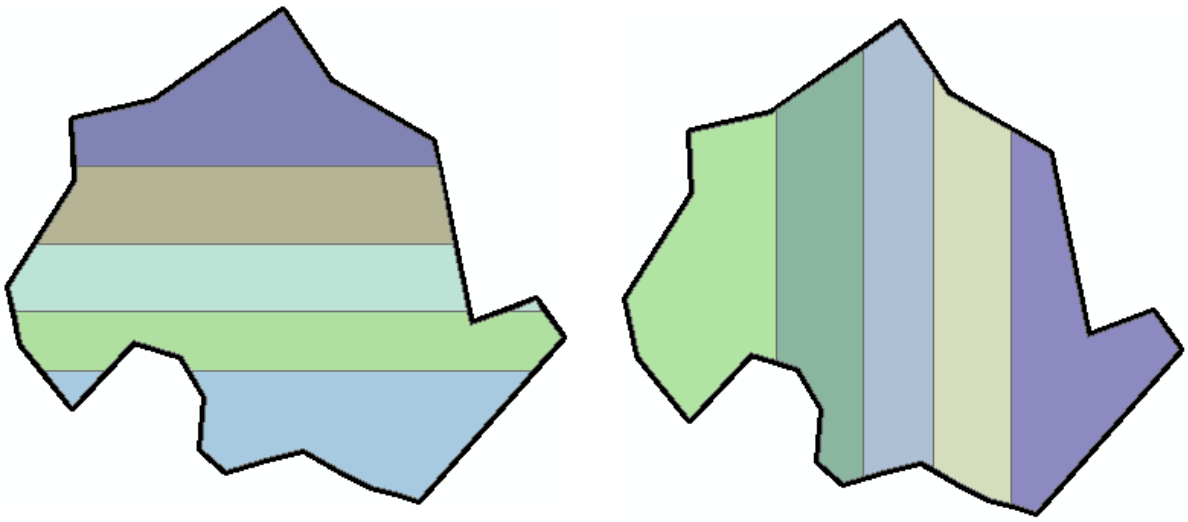
Bu araç ikinin üssünü (2, 4, 8, 16, 32 ve benzeri) takip eden bir sayı oluşturan bir çokgenin bölünmesi senaryosunda oldukça verimlidir. Bunun nedeni, bir çokgeni eşit alana sahip iki çokgene böldükten sonra, her biri tekrar iki parçaya bölünebilir ve bu böyle devam eder. Ancak bu senaryo bizim durumumuzda geçerli değildir. O yüzden farklı bir yaklaşım uygulanabilir:
1. Bir poligonun ölçüsü alınır.
2. Koordinatlarda küçük bir kaydırma ile poligonun kapsamının köşeleri kullanılarak bir çoklu çizgi oluşturulur.
3. Bu çizgiyi kullanarak poligon ikiye bölünür.
4. En küçük poligonun alanının ne olduğunu bulunur.
5. Eğer alan 200 metrekareden küçükse, çizgi tekrar kaydırılır ve 2-4 adımları tekrar çalıştırılır.
6. Alan 200 metrekare veya daha büyükse, bu madde geçilir ve 2-5 adımları uygulanarak kalan poligonlarla çalışma devam eder.
7.Orijinal poligon başarılı bir şekilde eşit alanlara bölündüğünde, bunlar kaynak poligon nitelikleriyle birlikte yeni bir özellik sınıfına eklenir.



Şekil 3.1.1 Kesme Çizgilerinin Poligon Ölçüsü İle Gösterimi [7]

Ancak bu yaklaşımın birkaç dezavantajı vardır. İlk olarak, poligon çok büyükse ve poligonun parçalarının minimum farkla aynı alana sahip olması gerekiyorsa, kesme çizgilerini birkaç santimetre kaydırmak, poligonun ikiye ayrılması ve sonucun değerlendirilmesi gerektiğinden aracın çalıştırılması çok zaman alacaktır. İkinci olarak, kesim çizgilerinin yalnızca Kuzey-Güney veya Batı-Doğu yönü arasında seçim yapılabilir. Açıyı ayrıca belirlenmez.

Ancak bu araç, verilen kullanım alanı için verimli çalışmaktadır. Kesme çizgisini hareket ettirmek için adım olarak 0,5 metre kullanıldığında, en büyük ve en küçük alt poligonlar arasındaki fark sadece %1 civarındadır. Aynı kod, kaydırma değeri 0,05 metre (5 cm) olarak ayarlanmış aynı poligon için çalıştırıldığında, fark değerinin %0,1 civarındadır.[7]



Şekil 3.1.2 Poligon Alt Bölümlemesinin Gösterimi(Batı-Doğu Solda, Kuzey-Güney Sağda) [7]

3.1.2. Turf.js İle Geospatial Analiz

Web tarafında verilen geometrik veriyi analiz etmek için kullanılabilecek araçlardan biri de turf.js kütüphanesidir. Turf.js, mekansal analiz için kullanılan açık kaynaklı bir JavaScript kütüphanesidir. Geleneksel uzamsal işlemleri, GeoJSON verilerini oluşturmaya yönelik yardımcı işlevleri, veri sınıflandırma ve istatistik araçlarını içerir. Turf, web sitesine istemci tarafı eklentisi olarak eklenebilir veya sunucu tarafında Node.js ile çalıştırılabilir. Kaynak kodu GitHub'da mevcuttur.

Turf.js'in temel özellikleri arasında coğrafi analizlerin gerçekleştirilmesi bulunmaktadır. Örneğin, iki nokta arasındaki mesafeyi hesaplama, bir alanın alanını ölçme veya poligon içindeki noktaları belirleme gibi fonksiyonlar, coğrafi veriler üzerinde detaylı analiz imkanı sağlar.

Geometri işlemleri kapsamında Turf.js, nokta, çizgi ve poligon gibi temel geometrik öğeler üzerinde çeşitli operasyonlar yapma yeteneği sunar. Bu operasyonlar arasında geometrileri birleştirme, kesme veya bir geometrinin merkezini bulma gibi işlemler bulunur.

Turf.js, harita işlemleri için de uygun bir çözüm sunar. Harita üzerinde nokta oluşturma, çizgi çizme veya belirli bir bölgedeki noktaları vurgulama gibi işlemlerle, kullanıcıya interaktif harita deneyimleri oluşturma imkanı tanır.

Topolojik işlemler kapsamında Turf.js, geometriler arasındaki topolojik ilişkileri kontrol etmek için kullanılan fonksiyonları içerir. İki geometri arasındaki kesişim, iç içe geçme

durumu gibi durumları belirleyerek, coğrafi analizlerin daha derinlemesine yapılmasına olanak sağlar.

Son olarak, Turf.js, GeoJSON formatıyla uyumlu çalışma özelliği sunar. Bu sayede coğrafi verilerin standart bir formatta temsil edilmesi ve Turf.js'in bu verilerle etkili bir şekilde çalışabilmesi sağlanır.[8]

4. YÖNTEM

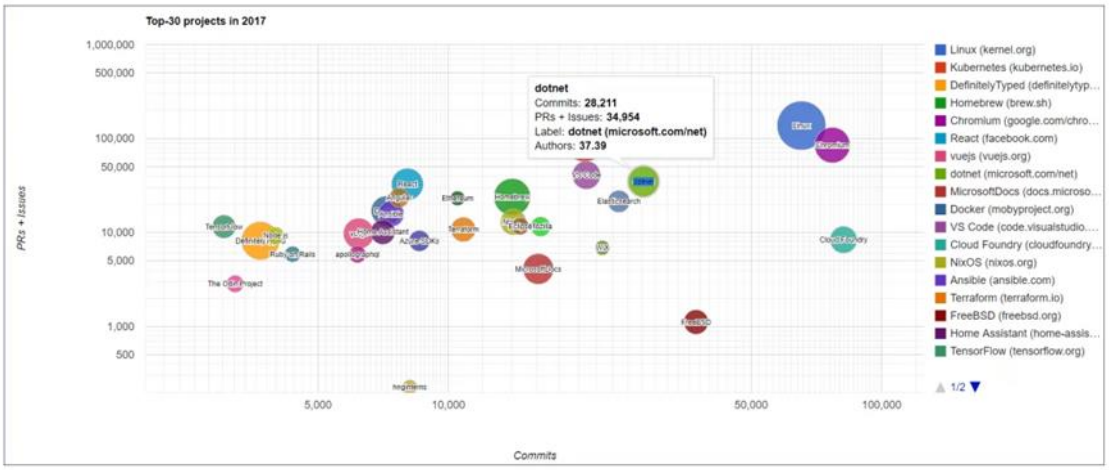
4.1. Kullandığımız Teknolojiler

4.1.1. NET Core

Projede .NET Core Framework kullanılmasına karar verilmiştir. Çünkü:

.NET Core sürekli olarak gelişmekte ve kullanıcılarına daha fazla özellik ve kolaylıkla hizmet vermeye devam etmektedir. Eski versiyonlarına göre kullanım alanı genişlemiş ve Windows, Linux ve MacOs işletim sistemlerinde de uygulama geliştirme imkanı sunmaktadır. .NET Core ile pek çok alanda uygulama yazmak mümkündür. En başta web uygulamaları olmakla beraber mobil uygulamaları, masaüstü uygulamaları, bulut hizmetleri, mikroservisler, API'ler, oyunlar ve IoT uygulamaları geliştirmeye imkan sunar ve bu da en avantajlı yanlarından birisidir. .NET Core tek bir dille sınırlı değildir pek çok dil ile kullanılabilmektedir.[9] Bununla birlikte .NET Core, genellikle C# kullanılması tercih edilmektedir. Geliştirme ortamı olarak genellikle Visual Studio ve Visual Studio Code tercih edilmektedir. Bu projede iki farklı geliştirme ortamını da kullanarak ikisinin de farklı özelliklerinden faydalanıldı.

C#, güçlü bir nesne yönelimli programlama dilidir ve .NET Core'un temel dilidir. Ayrıca, .NET Core ücretsiz ve açık kaynak bir platformdur. 2017 yılında, dotnet projesi topluluktan 28.000'den fazla taahhüt aldı ve en hızlı 30 açık kaynak proje arasında yer aldı.[11]



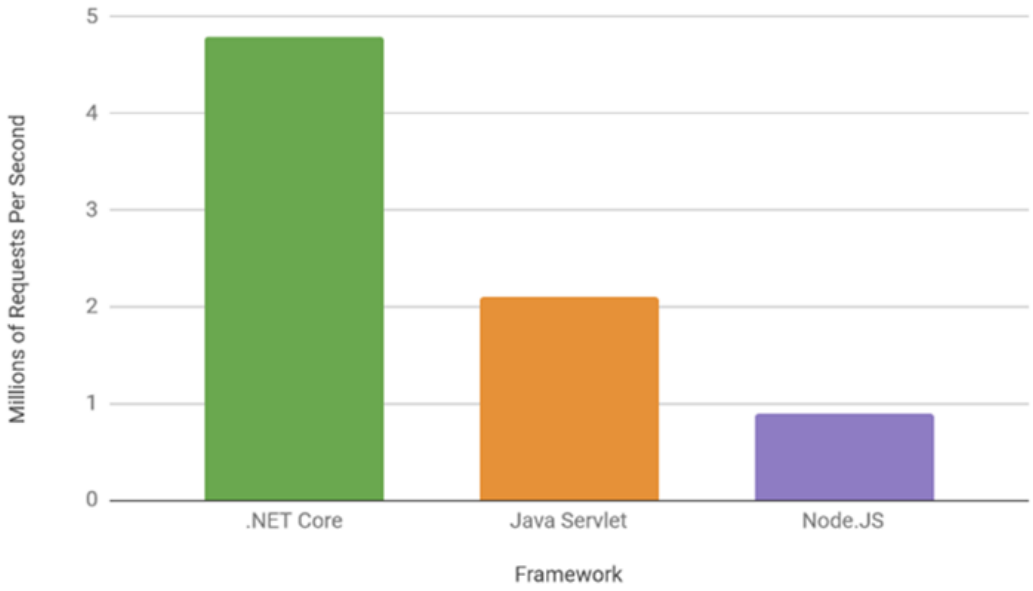
Şekil 4.1.1 2017 de ki Top 30 Proje [12]

Şu anda hala geliştirilmeye devam eden büyük bir framework olarak kullanılmaktadır. Tarih içinde çıkarılmış olan versiyonları aşağıdaki tablodaki gibidir:

Version	Release date	Released with	Latest update	Latest update date	Support ends[24]
.NET Core 1.0	June 27, 2016[25]	Visual Studio 2015 Update 3	1.0.16	May 14, 2019	June 27, 2019
.NET Core 1.1	November 16, 2016[26]	Visual Studio 2017 Version 15.0	1.1.13	May 14, 2019	June 27, 2019
.NET Core 2.0	August 14, 2017[16]	Visual Studio 2017 Version 15.3	2.0.9	July 10, 2018	October 1, 2018
.NET Core 2.1	May 30, 2018[17]	Visual Studio 2017 Version 15.7	2.1.39 (LTS)	July 11, 2023	August 21, 2021
.NET Core 2.2	December 4, 2018[18]	Visual Studio 2019 Version 16.0	2.2.8	November 19, 2019	December 23, 2019
.NET Core 3.0	September 23, 2019[27]	Visual Studio 2019 Version 16.3	3.0.3	February 18, 2020	March 3, 2020
.NET Core 3.1	December 3, 2019[28]	Visual Studio 2019 Version 16.4	3.1.32 (LTS)	December 13, 2022	December 13, 2022
.NET 5	November 10, 2020[29]	Visual Studio 2019 Version 16.8	5.0.17	May 10, 2022	May 10, 2022
.NET 6	November 8, 2021[30]	Visual Studio 2022 Version 17.0	6.0.23 (LTS)	October 10, 2023	November 12, 2024
.NET 7	November 8, 2022[23]	Visual Studio 2022 Version 17.4	7.0.12	October 10, 2023	May 14, 2024
.NET 8	November 14, 2023[31]	Visual Studio 2022 Version 17.8	8.0.0 (LTS)		November 10, 2026
.NET 9	November 2024 (projected)				May 2026 (projected)
Legend: Old version	Older version, still maintained	**Latest version**	Latest preview version	Future release		

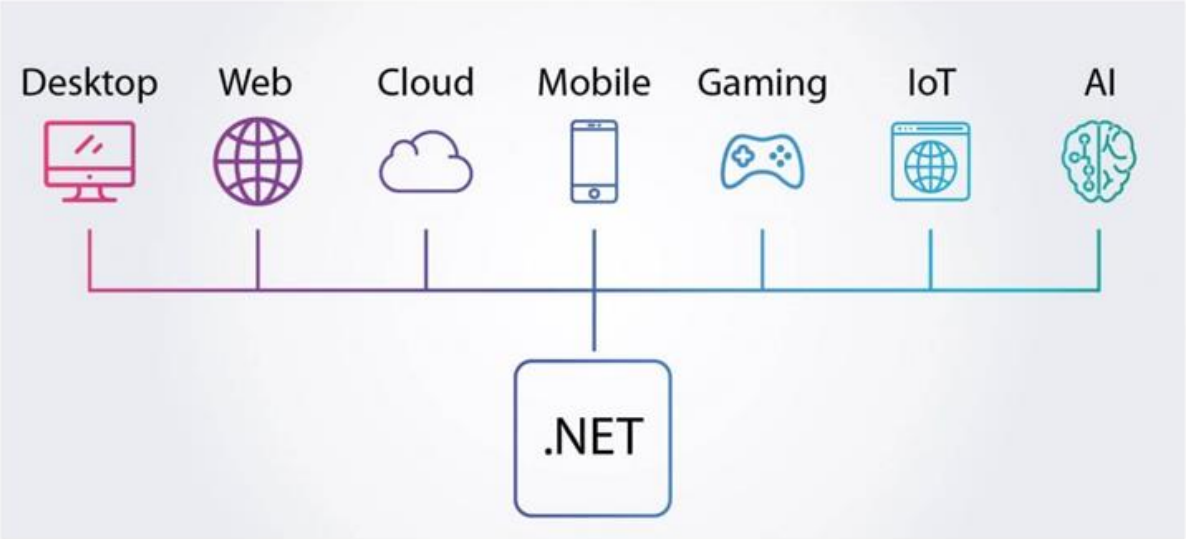
Tablo 4.1.1 .NET Core'un Yıllar İçindeki Versiyonları [11]

Performans açısından değerlendirildiğinde .NET Core, diğer sunucu tarafı frameworkleri olan Java Servlet ve Node.js'ten önemli ölçüde daha hızlıdır. ASP.NET Core, mükemmel destekle asenkron programlama kalıplarını kullanır. Zaman uyumsuz programlama, artık yaygın .NET sınıflarında ve birçok üçüncü taraf kütüphanede uygulanmış durumdadır. Tüm bu özellikler, frameworkün performansını artırır. MVC'nin eski (Core olmayan) sürümünden 23 kat daha fazla isteği işleyebilir. Ayrıca, Node.js'den yaklaşık 5 kat daha hızlıdır. [10] Bu da sektörde .NET Core kullanımını önemli bir yerde tutmaktadır ve Node.js'e göre eski olmasına rağmen güçlü bir rakip yapar.



Şekil 4.1.2 .NET Core ve Diğer Frameworkler Arasındaki Performans Farkı [10]

.NET Core, farklı platformlarda çalışabilen, modüler, açık kaynaklı bir uygulama geliştirme frameworküdür. Bu framework, çeşitli uygulama türlerini destekleyerek geliştiricilere geniş bir esneklik sağlar. .NET Framework'ü, oyun, mobil, IoT ve yapay zeka gibi birçok alanda uygulama geliştirmek için geniş bir kullanım alanına sahiptir.



Şekil 4.1.3.NET Core'u Destekleyen Çeşitli Uygulamalar [12]

4.1.2. MVC Modeli

Proje, .NET Core'un MVC şablonunu temel alarak geliştirilmiştir. MVC şablonu projenin backend ve frontend kısımlarının bir uyum içinde çalışmasında yardımcı olduğu için çok faydalanılmıştır. MVC, Model-View-Controller prensiplerine dayanan bir tasarım desenidir ve uygulamanın modüler bir yapıya sahip olmasını sağlar.

4.1.3. MVC Tasarım Deseni

Model-View-Controller (MVC) mimari deseni, bir uygulamayı üç ana bileşene ayrılır:

Model: Veri ve iş mantığını içeren kısımdır. veritabanı işlemleri, veri doğrulama bu kısımda yapılır.
View: Kullanıcı arayüzünü oluşturan ve kullanıcıya bilgi sunan kısım burasıdır . HTML, CSS ve JavaScript gibi teknolojilerle entegre çalışır.
Controller: Kullanıcının taleplerini karşılayan, iş mantığını tetikleyen ve Model ile View arasındaki ilişkileri yapan kısımdır.

MVC Şablonunu kullanmanın bazı avantajları:

MVC, projeyi mantıklı ve bağımsız modüllere böler, bu da geliştirilebilirliği artırır ve bakımı kolaylaştırır. Her katmanın belirli bir sorumluluğu vardır, bu da kodun okunabilirliğini ve sürdürülebilirliğini artırır. Model, View ve Controller'ın ayrı ayrı test edilebilmesi, yazılım kalitesini artırır. View'lar ve Controller'lar değiştirilebilir, bu da projenin gereksinimlere daha iyi uyarlanmasını sağlar.

4.1.4. EF Core

EF Core, .NET tabanlı uygulamalarda veritabanı işlemlerini yönetmeyi ve nesne modelleri arasındaki ilişkileri veritabanındaki tablolarla eşleştirmeyi sağlar. First Code mantığı ile SQl sorgusu yazmak yerine nesne odaklı bir yaklaşım ile C# ve LINQ (Language Integrated Query) kullanarak veritabanı için gerekli olan sorgular ve işlemler yazılabilmektedir. LINQ kullanmak veritabanı sorgularının daha okunabilir olmasını sağlar. EF Core, farklı veritabanı sağlayıcıları için genişletilebilir bir yapı sunar. Veritabanı şemalarını kod tarafında yönetmeyi sağlayan "Migrations" özelliği sayesinde, uygulamanın gelişimi sırasında veritabanı şemalarını güncellemek daha kolay hale gelir. Asenkron programlama kullanımı ise veritabanı işlemlerini asenkron olarak gerçekleştirmeyi sağlar. Bu, uygulamanın daha iyi performans göstermesine

katkıda bulunur. Projede "7.0.12" versiyonu kullanılmaktadır.

```
<PackageReference Include="Microsoft.EntityFrameworkCore.Design" Version="7.0.12">
<IncludeAssets>runtime; build; native; contentfiles; analyzers; buildtransitive</IncludeAssets>
<PrivateAssets>all</PrivateAssets>
```

Şekil 4.1.4 EF CORE Kullanılan Version

4.1.5. Sql

SQL, bilgileri bir ilişkisel veritabanında depolamak ve işlemek için kullanılan bir programlama dilidir. İlişkisel veri tabanları, veri tablosu biçiminde satırlar ve sütunlar aracılığıyla farklı veri niteliklerini ve değerleri temsil eden çeşitli ilişkileri içerir. SQL ifadeleri kullanılarak, veri tabanındaki bilgileri depolanabilir, güncellenebilir, kaldırılabilir, aranabilir ve çekilebilir. Ayrıca, SQL kullanılması veritabanı performansını korumak ve optimize etmek için fayda sağlamaktadır. [13]

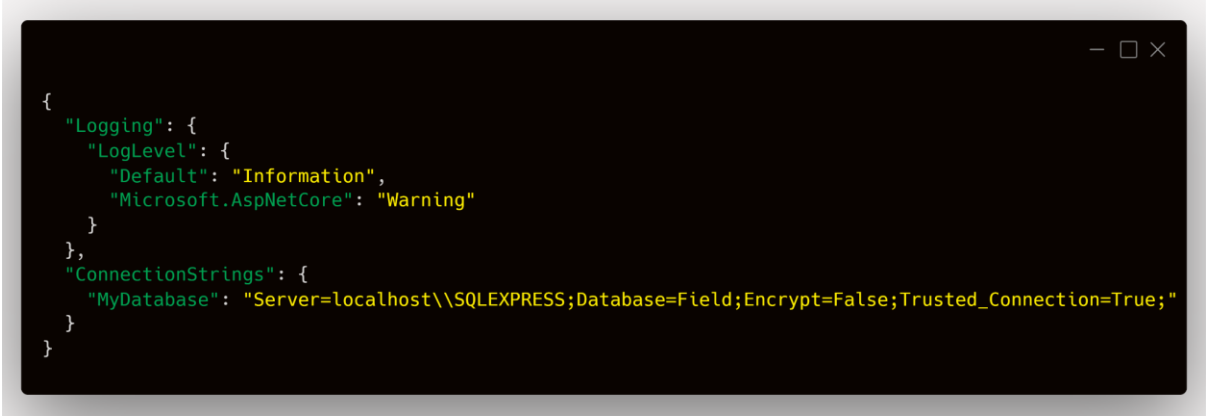
İki farklı türde SQL vardır. Birincisi SQL ikincisi NO SQL dir. SQL kullanmak veriyi herkesin ulaşabileceği bir şekilde tutmak için çok gereklidir projede ilk başta SQLite kullanılması planlanmıştır ancak daha sonrasında kullanım tercihlerinden dolayı MSSQL veritabanına geçiş yapılmasına karar verilmiştir. Projede kullanım kolaylığından ve projeye uygunluğu açısından SQL kullanılması tercih edilmiştir.

4.1.6. MS SQL Server Nedir?

Microsoft SQL olarak bilinen MSSQL bize SQL Server Management Studio (SSMS) gibi grafik arayüzlü yönetim araçları sağlar. Rakiplerine göre oldukça köklü ve sağlam bir yapısı vardır. Visual Studio kullanıcıları için kolaylaştırılmış eklentileriyle .NET Core için önemli bir teknoloji haline gelmiştir.[14] Projede "7.0.8" versiyonu kullanılmaktadır. Connection Strings kullanılarak enjeksiyon yapılmaktadır.

```
<PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer" Version="7.0.8" />
```

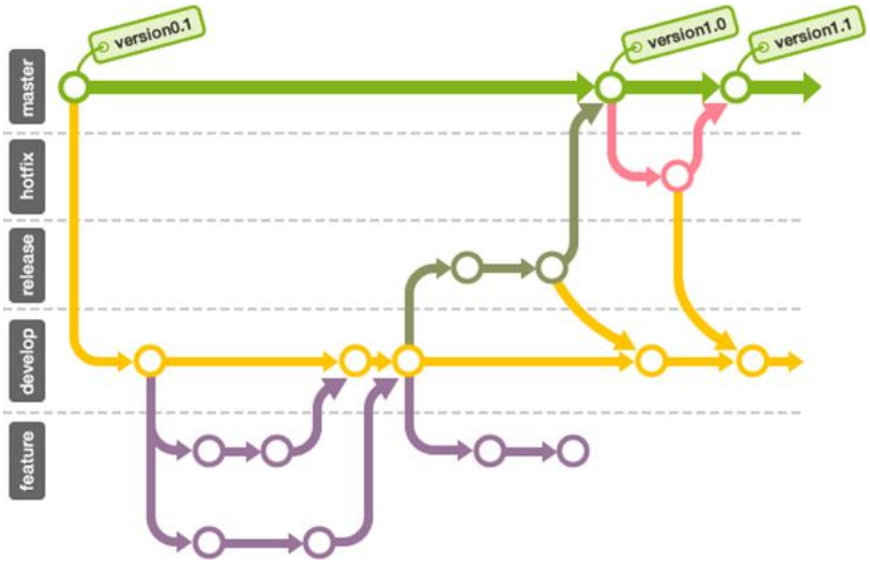
Şekil 4.1.5 MSSQL Kullanılan Version



Şekil 4.1.6 Connection Strings

4.1.7. Github

Github yazılan kodların düzenli bir şekilde gösterilmesi, düzenlenmesi ve kodların geçmiş sürümlerine ulaşılabilmesi için temel olarak depolama amacıyla kullanılan bir teknolojidir. GitHub üzerinde yeni bir depo oluşturmak oldukça basittir. İlk adım olarak, "New repository" seçeneğini kullanılır. Bu aşamada, depo adı, açıklama ve gizlilik ayarları gibi temel bilgiler girilir. İlgili repon un herkes tarafından görülmemesi gerekiyorsa public seçeneğini seçmek tehlike arz edebilir. Ardından, başlangıç kodları doğrudan GitHub üzerinden yüklenebilir veya Git kullanarak lokal bilgisayar üzerinden repoya başlangıç kodları iletilebilir. GitHub üzerinde yapılan tüm değişiklikler, Git mantığına uygun olarak kaydedilir ve sürüm yönetimi sağlanır. Git üzerinde değişiklikleri kaydetmek için Commit özelliğini kullanmak gerekmektedir. Bu, her bir değişikliğin bir "commit" olarak adlandırılmasını ve bu commit'lerin tarihçesinin tutulmasını içerir. Böylece, herhangi bir zamanda değişikliklerin tarihçesi göz atılabilir, ne zaman, hangi değişikliklerin yapıldığı ve kim tarafından yapıldığı detaylı bir şekilde incelenebilir ve gerekli durumlarda eski bir commite geri dönüş sağlanabilir. Ayrıca, GitHub'ın işbirliği yetenekleri sayesinde projeye katkıda bulunanların değişiklikleri takip etmeleri, geri bildirim sağlamaları ve proje üzerinde etkileşimde bulunmaları oldukça kolaydır. Bu, özellikle açık kaynaklı projelerde ve ekip çalışmalarda işbirliğini güçlendirir. [16]



Şekil 4.1.7 Git de Version Branch Yapısı [15]

4.1.8. Git

Yazılım geliştirme süreçlerinde önemli kolaylık ve hız sağlayan bir sürüm kontrol ve kod yönetim sistemi olan Git, birçok şirket tarafından tercih edilmektedir. [15]

Git, kullanımında birçok avantaj sunar. Bu avantajlardan bazıları şunlardır:

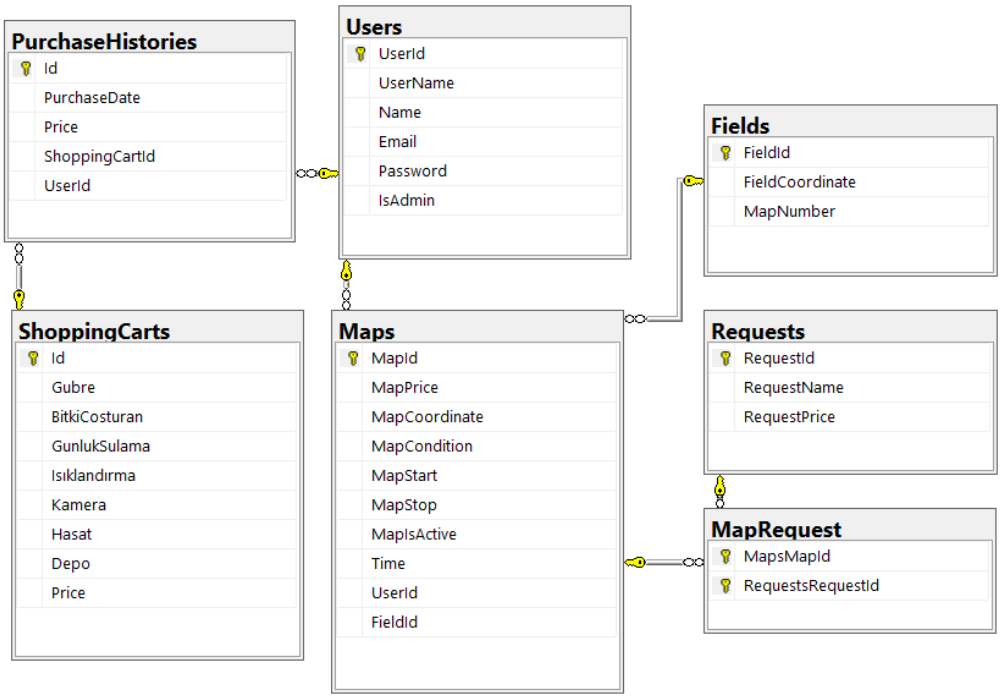
Verilerin etkili bir şekilde yedeklenmesi için Git, kullanıcılara güçlü bir yedekleme mekanizması sağlar. Bu sayede, projelerdeki değişikliklerin izlenmesi ve gerektiğinde geri alınması kolaylaşır. Git'in en güçlü özelliklerinden biri branch ve merge işlemleridir. Kullanıcılar, aynı anda birden fazla iş üzerinde çalışabilir ve ardından bu dalları başarılı bir şekilde birleştirebilirler. Çevrimdışı kullanım imkanı, internet bağlantısı olmadan da Git'in kullanılabilmesini sağlar. Bu özellik, kullanıcıların projelerine herhangi bir yerden erişimlerine ve çalışmalarına devam etmelerine olanak tanır. Ekip çalışmasında kolaylık, Git'in çok kullanıcılı ortamlar için optimize edilmiş olmasıyla ilgilidir. Kullanıcılar, aynı projede eş zamanlı olarak çalışabilir ve değişiklikleri uyumlu bir şekilde birleştirebilirler. Dağıtık yapı, Git'in her kullanıcının lokal bir kopyasını tutmasını sağlar. Bu, projelerin hızlı bir şekilde çoğaltılmasını ve paralel olarak çalışmayı mümkün kılar. Geçmişin kolay saklanması, Git'in her değişikliği bir "commit" olarak kaydetmesi sayesinde gerçekleşir. Bu sayede, projenin geçmişi ve değişikliklerin takibi kolaylaşır.[15]

Git ile çalışırken kullanılan başlıca komutlar şunlardır

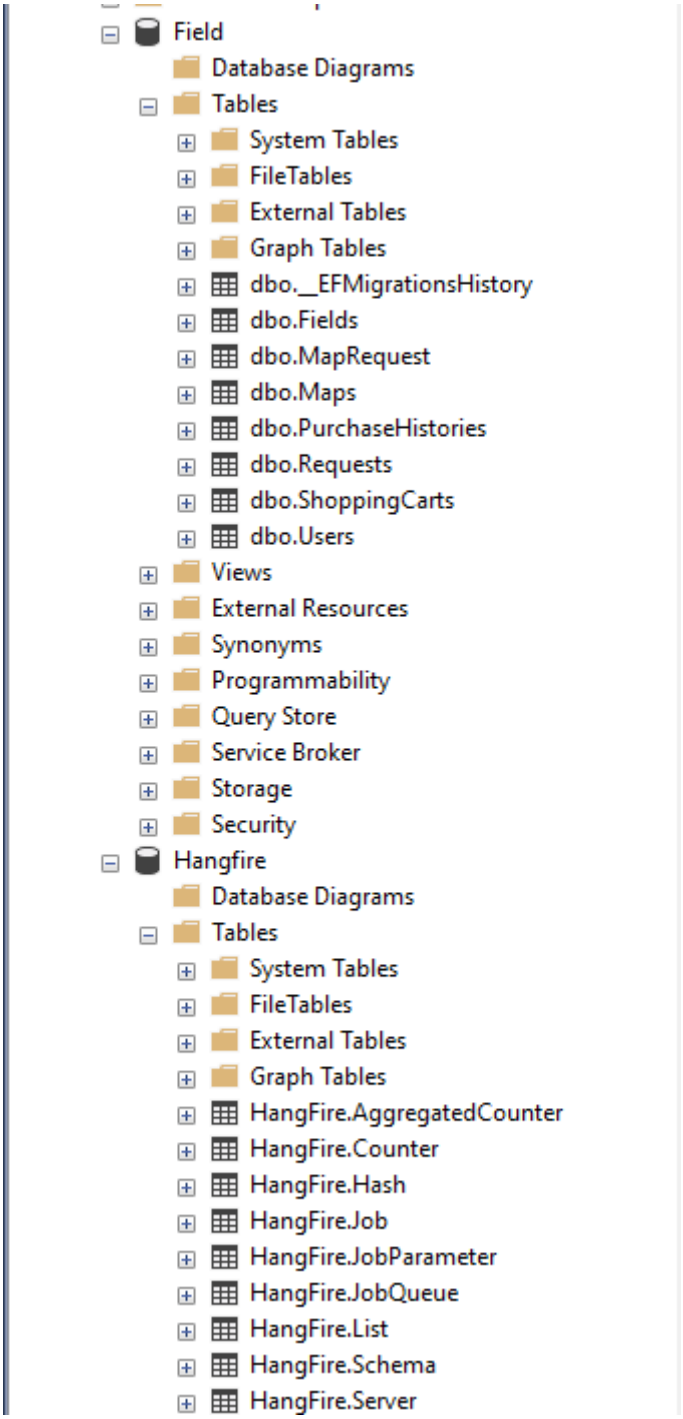
clone	Uzaktaki bir git deposundaki bir projeyi kopyalamak için kullanılır.
checkout	Farklı bir dal veya geçmişteki bir commit'e geçiş yapmak için kullanılır.
pull	Uzak depodan en son değişiklikleri almak ve yerel kopyayı güncellemek için kullanılır.
push	Yerelde yaptığınız değişiklikleri uzaktaki bir depoya göndermek için kullanılır.
commit	Yapılan değişiklikleri yerel depoya kaydetmek için kullanılır.
branch	Projede farklı geliştirme hatlarını (branch) oluşturmak ve yönetmek için kullanılır.
merge	İki ayrı dalı birleştirmek için kullanılır.
diff	Çalışma dizinindeki değişiklikleri göstermek için kullanılır.
log	Commit geçmişini görüntülemek için kullanılır.

4.2. Kullanılan SQL Tabloları

Proje Field Map User ShoppingCart Request PurchaseHistory tablolarını içermektir. Field tablosu Admin tarafından eklenen tüm bahçeleri kaplayan arsaya denk gelmektedir. Map Haritadaki her bir arsadaki parsellere yani kiralanacak olan hobi bahçelerine karşılık gelmektedir. User kayıt olan ve harita kiralayacak olan her bir müşteriyi temsil etmektedir. Request Bahçe kiralandıkt an sonra bahçede kullanımı talep edilen Ek hizmetleri içerir. PurchaseHistory Alış veriş sepeti olarak kullanılmakta ve müşteri bahçeyi kiralama işleminden sonra veriyi tutulacağı yerdir. Projedeki Field Map tabloları bire çok ilişkisi içerisindedir. Çünkü her arsada birçok parsel vardır. User pek çok Map'e sahip olduğundan User ve Map tabloları arasında bire çok ilişkisi vardır. Bir Map birçok ek hizmet talep edeceğinden ve ek hizmetler birçok Map'de bulunabileceğinden Map ve Request tabloları arasında çok a çok ilişki vardır. Bu yüzden MapRequest tablosu ekstra olarak çok a çok ilişkiyi tamamlamak için bulunmaktadır. Bunun haricinde Geçerli veri tanının yanında Hangfire veritabanıda sisteme eklenmiştir ve 2 farklı veri tabanı olarak kullanılmaktadır.



Şekil 4.2.1 Projedeki SQL Tabloları



Şekil 4.2.2 Projedeki Handfire Veritabanı

4.3. Kullanılan Yöntemler

Projede şu anda backend kısmı için admin ve user panelleri oluşturulmuştur.

```
[HttpPost]
public async Task<IActionResult> Login(LoginViewModel model)
{
  if (ModelState.IsValid)
  {
    var isUser = _userRepository.Users.FirstOrDefault(x => x.Email ==
model.Email && x.Password == model.Password);
    if (isUser != null)
    {
      var userClaims = new List<Claim>();
      userClaims.Add(new Claim(ClaimTypes.NameIdentifier,
isUser.UserId.ToString()));
      userClaims.Add(new Claim(ClaimTypes.Name, isUser.UserName ?? ""));
      userClaims.Add(new Claim(ClaimTypes.GivenName, isUser.Name ?? ""));
      if (isUser.Email == "info@muhammed.com")
      {
        userClaims.Add(new Claim(ClaimTypes.Role, "admin"));
      }
```

3.3.1 Login için kullanılan kod bloğunun bir kısmı.

Projede giriş yapıldığında profil sayfasına yönlendirilmektedir ve ardından burada user için kiralanan haritalarda bir günden az vakit kalmışsa o parseller için uyarı gelmektedir. Müşteriler kiralama işlerini yaparak parselleri sepete ekleyebilir ve ardından parselleri kiralama işlemi ile hesaplarına aktarabilirler. Kiralama geçmişlerini profilden görüntüleyebilirler. Haritada yer değiştirmeye başka haritalar arasında yer değiştirme işlemlerini ve ek hizmet talep etme işlemlerini yapabilmektedirler. Bunlar için alışveriş sepeti kullanılmamıştır. Bunlar direkt ödeme sayfasına yönlendirilerek tamamlanmaktadır. Admin ise herhangi bir ücrete tabi olmadan işlerini yapmaktadır.

Admin işlemleri admin Controller da sağlanmaktadır. User işlemleri ise User Controller da sağlanmaktadır. Admin ve User için profil sayfası oluşturulmuştur. View klasörlerinde Admin User ve Login için klasörler ve onların içinde .cshtml dosyaları bulunmaktadır. Controller ve Views arasındaki bağlantıyı sağlayan ViewModel dosyaları Models klasörün içinde bulunmaktadır.

Projede Mssql kullanılmıştır. Proje ilk başta eski Sqlite ile başlanılıp hız ve kolaylıklarından ötürü ve github ve git işlemlerindeki kolaylıklarından dolayı Mssql tercih edilmiştir. Projeye ilerleme aşamasında git, github ve discord gibi teknolojilerden faydalanılmıştır. Projeye bootstrap ile bir tema hazırlanmış ve giriş login ana sayfa profil sayfası ve diğer işlemler için bu tema uygulanmıştır. İstenilen harita için eşit parçalı Parseller oluşturulması algoritmaları

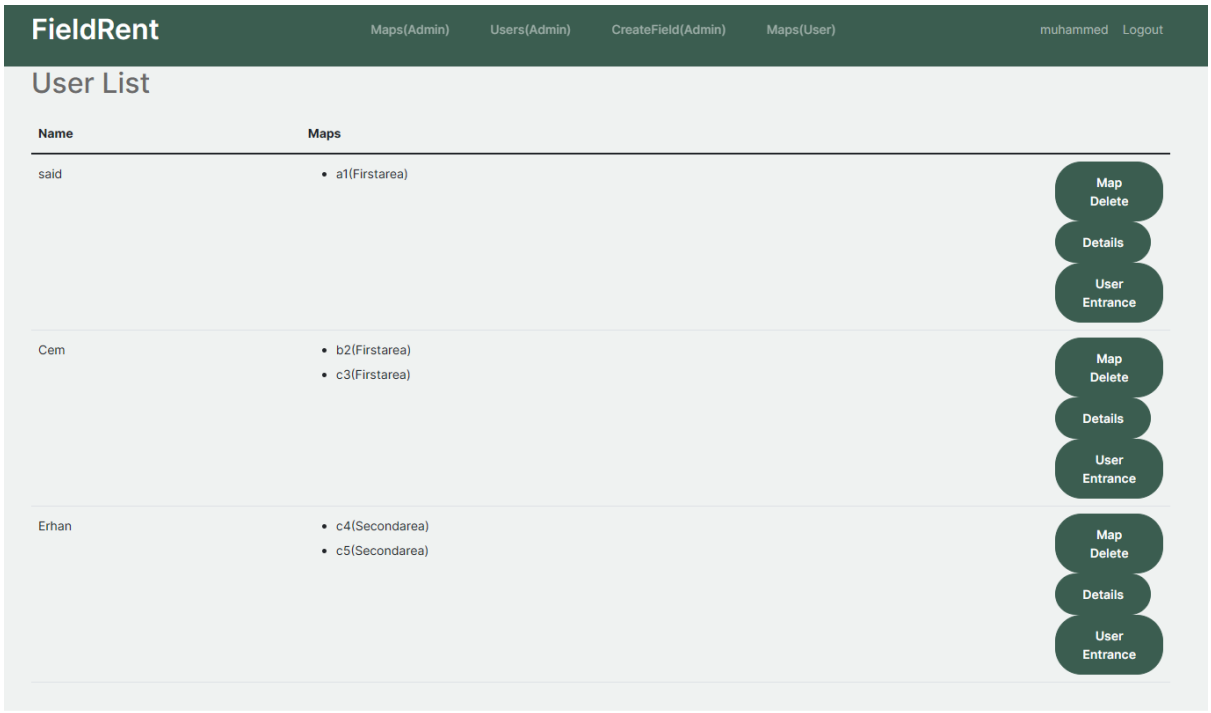
üzerinde araştırmalar yapılmıştır ve çeşitli algoritmalar bulunmuştur. Bunlar değerlendirilmeye alınmıştır.

Projede Repository Data klasöründe oluşturulmuş ve Data klasöründeki Abstract klasında temel fonksiyonlar oluşturulmuş ve inheritance ile Concrete Klasöründeki sınıflara aktarılmıştır.

Admin Yönetimi:

Admin yönetimi için başlıca 2 başlık vardır. Users Ve Maps sayfaları.

Maps: buralarda harita ve arseller tablolar halinde gözlemlenebilir.

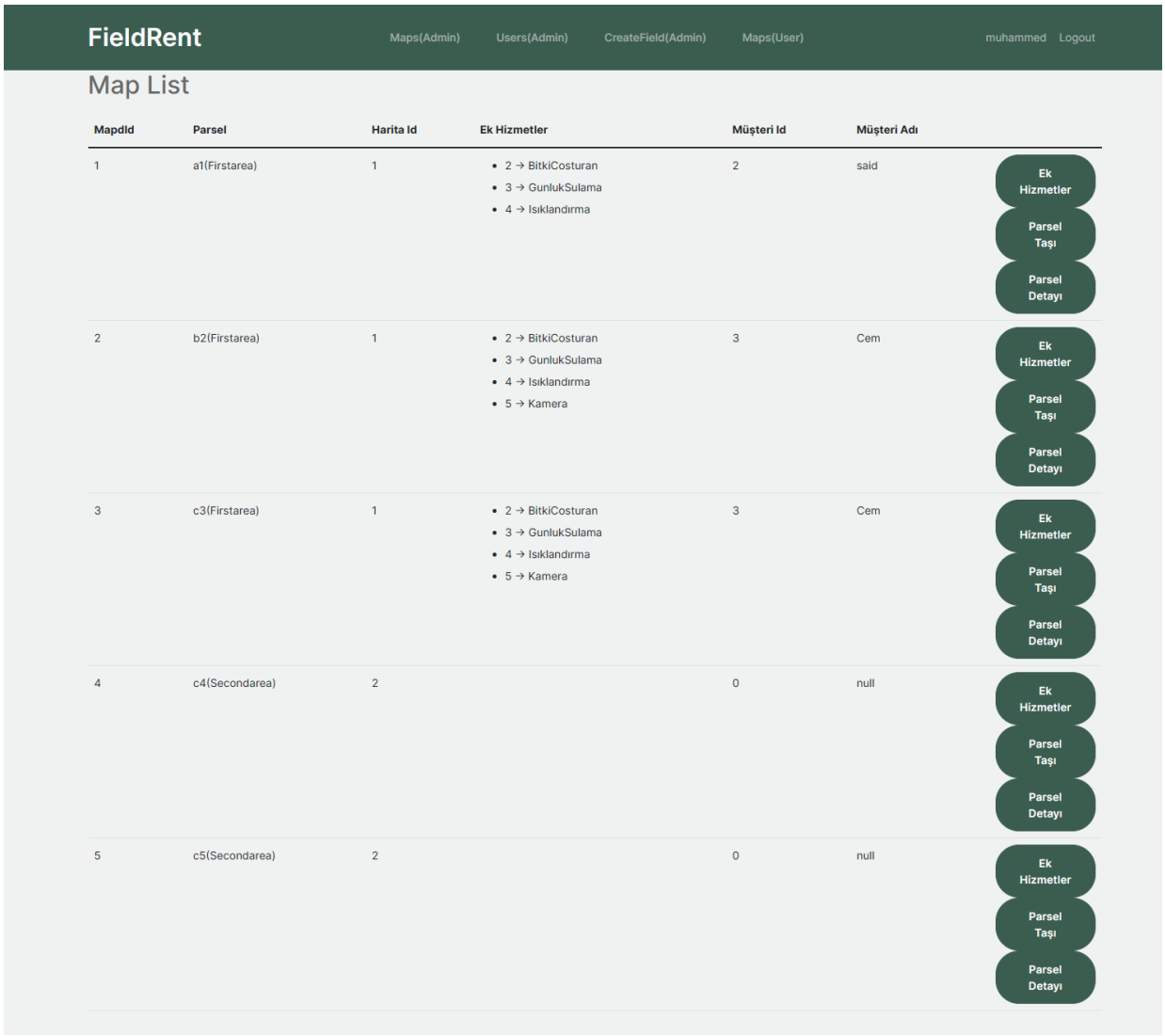


Şekil 4.3.1 Admin Users Sayfası

Users: Burada Müşteriler ve bilgileri liste halinde görüntülenebilir.

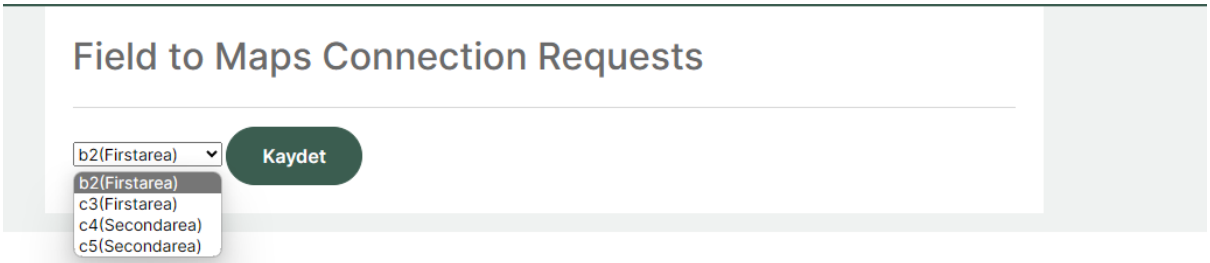
```
[HttpGet]
public async Task<IActionResult> Details(int? id)
{
   var user = await _userRepository.Users.Include(x => x.Maps).FirstOrDefaultAsync(x => x.UserId
== id);
   return View(user);
}
```

Db'deki Userları listelemek için kullanılan endpoint.



Şekil 4.3.2 Admin Maps Sayfası

Admin Panelinde Maps sayfasında istenilen parsel için yer değişikliği talebi yapılabilmektedir.

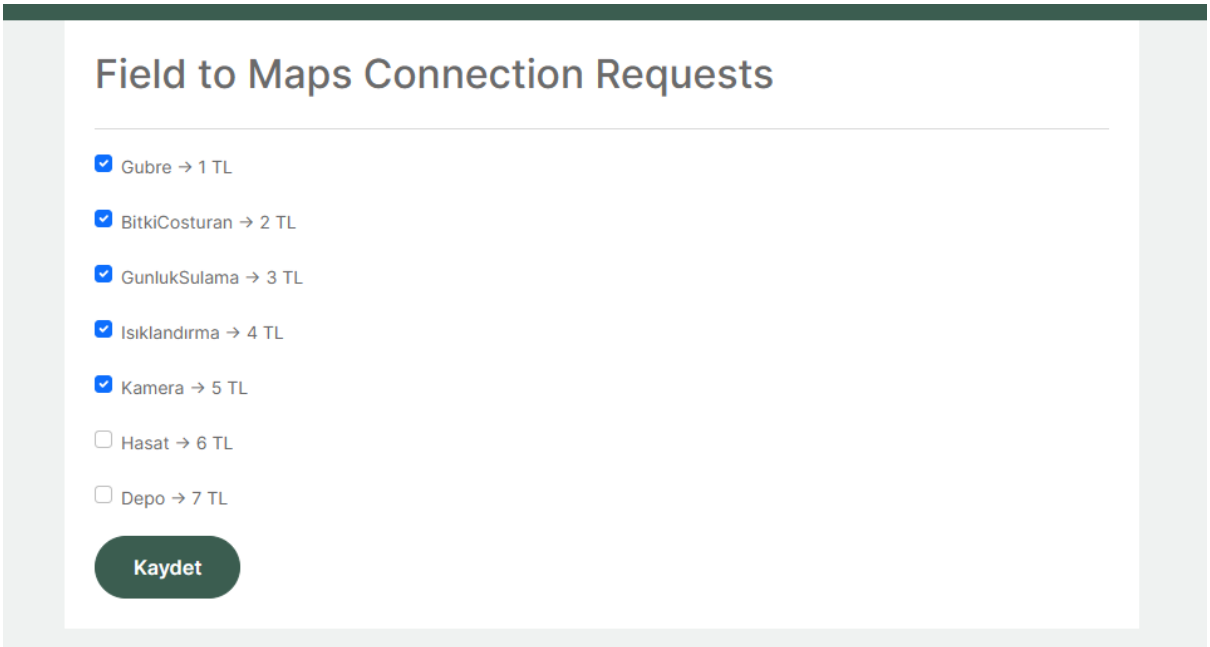


Şekil 4.3.3 Admin Yer Değiştirme Sayfası

```
[HttpGet]
public async Task<IActionResult> Change_Map(int? id)
{
   if (id == null)
   {
      return NotFound();
   }
   var map = await _mapRepository.Maps.FirstOrDefaultAsync(p => p.MapId
== id);
   ViewBag.mapslist = await _mapRepository.Maps.Where(x => x.UserId != id
&& x.MapIsActive == true).ToListAsync(); //.Where(x=>x.UserId!=id) tüm liste
için sil
   ViewBag.maplist2 = await _mapRepository.Maps.Where(x => x.MapIsActive
== true).ToListAsync();
   return View();
}
```

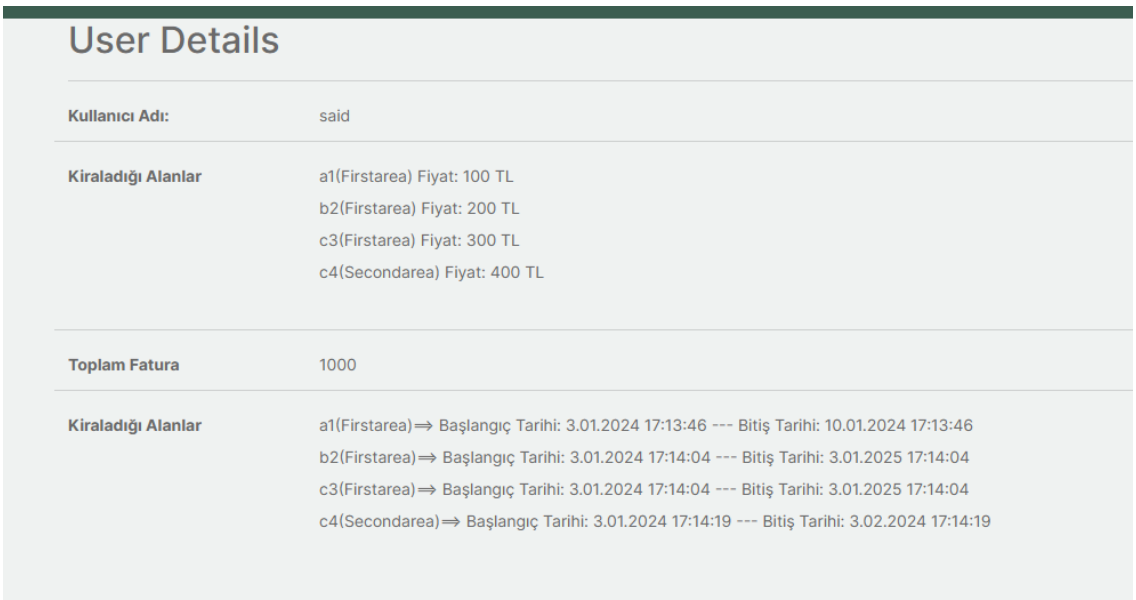
Map Change Endpointi

Admin Panelinde Maps sayfasında istenilen parsel için ek hizmetlerde ekleme veya çıkartma yapılabilir.



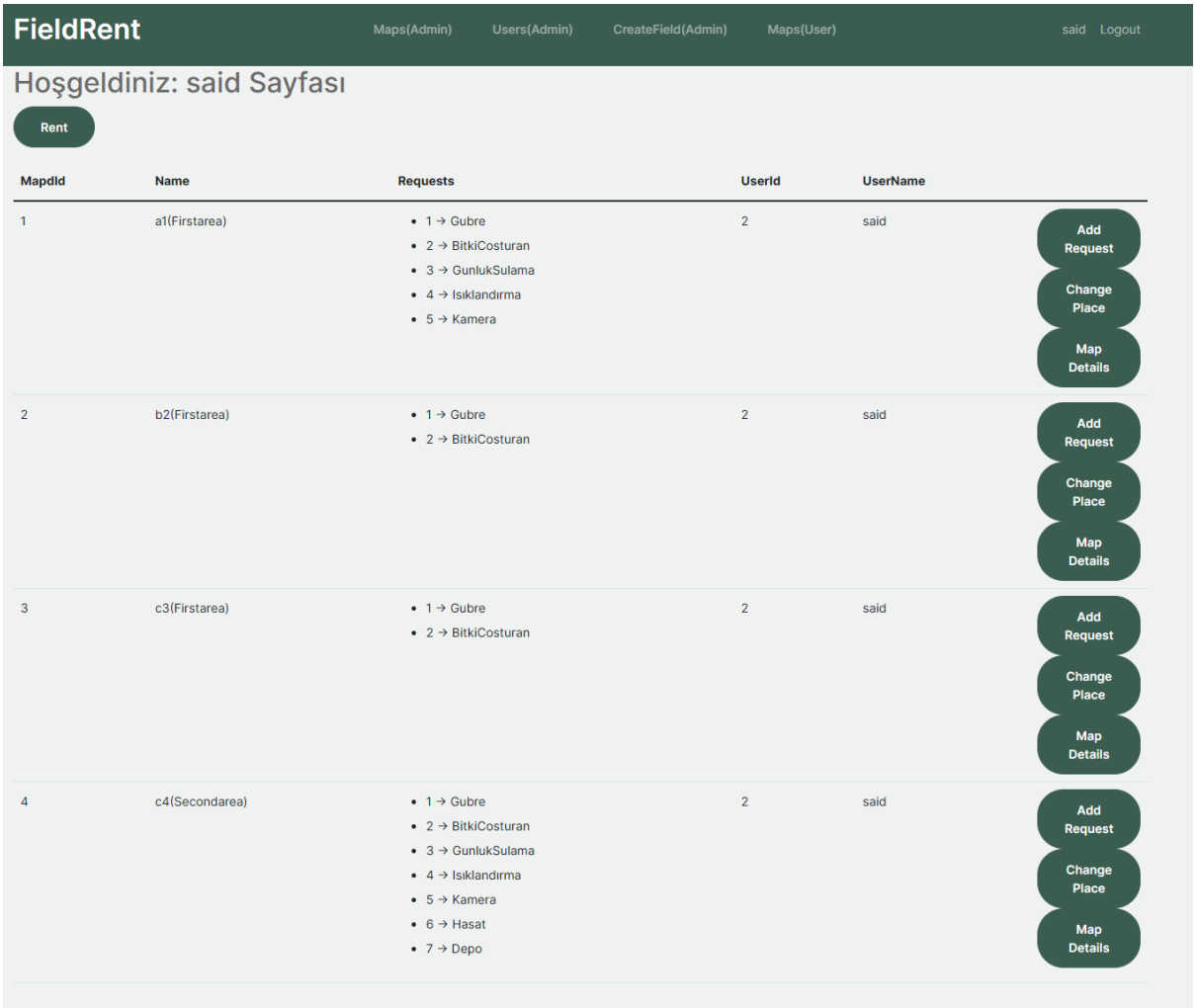
Şekil 4.3.4 Admin Ek Hizmet Sayfası

Admin panelinde seçilen usera ait bilgiler görüntülenebilmektedir.



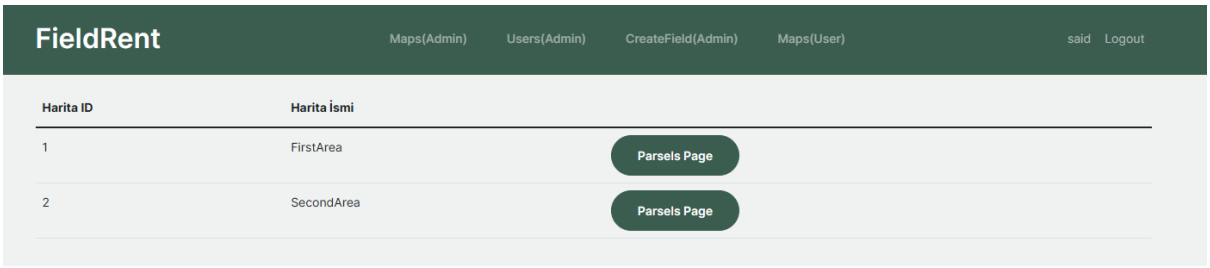
Şekil 4.3.5 Admin Users Detay Sayfası

User Yönetimi:
User da girişte karşımıza çıkan sayfa profil sayfası gibi çalışır ve kiralanan parselleri görüntüler.



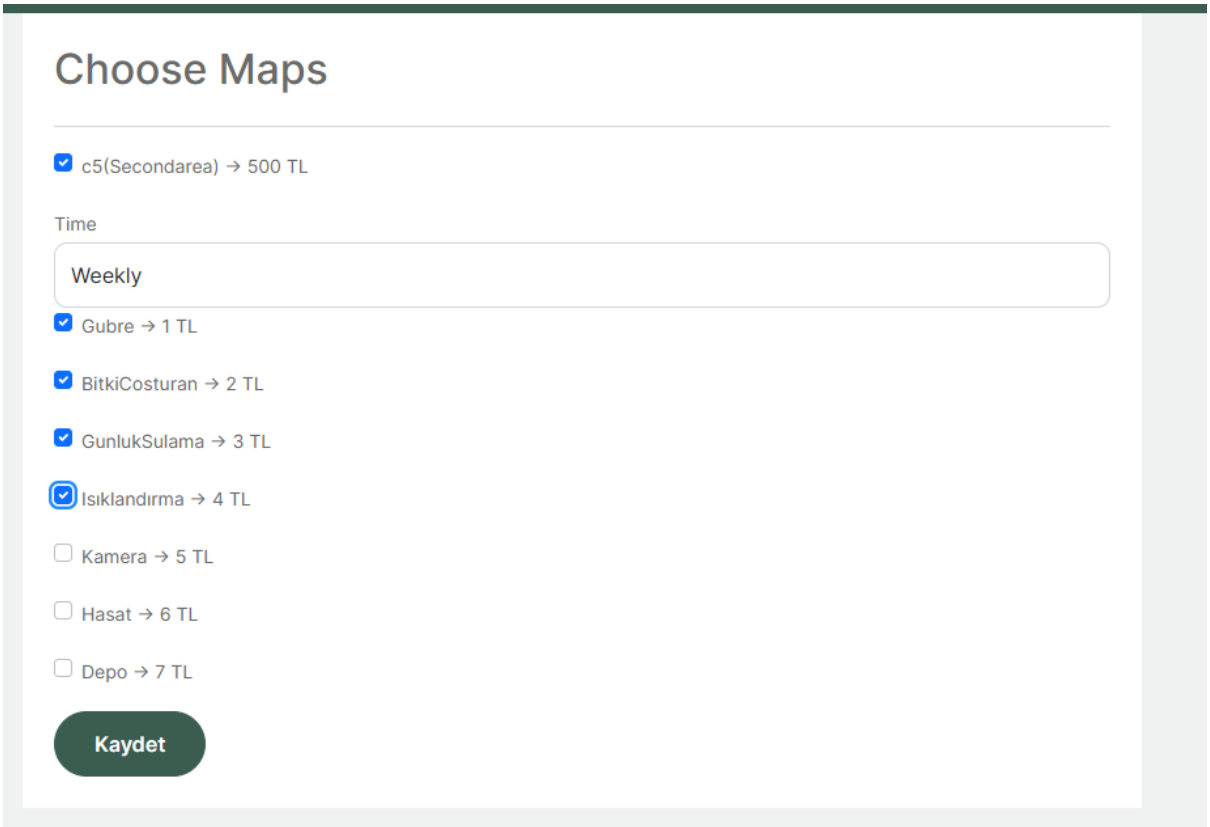
Şekil 4.3.6 Users Sayfası

User sayfasında kiralama işlemi için tıklandığında istenilen harita seçimi için seçim ekranı gelir.



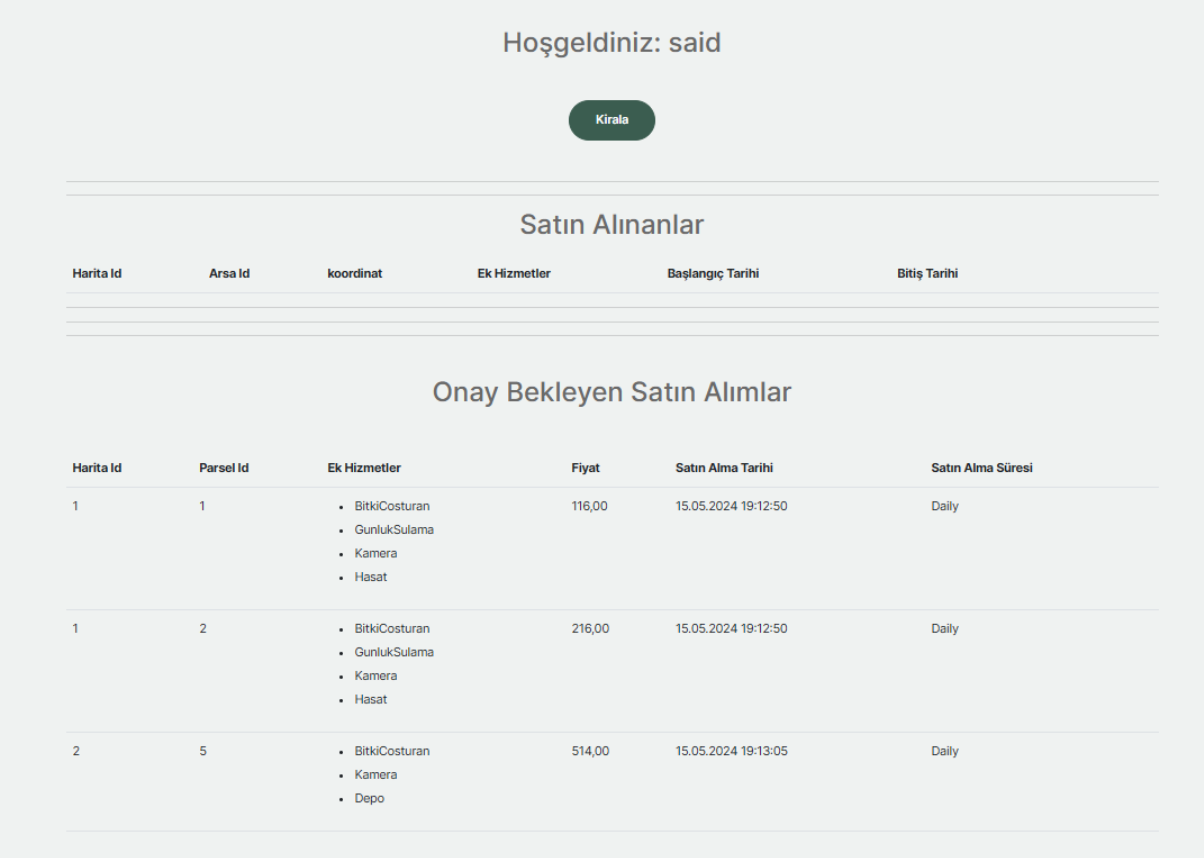
Şekil 4.3.7 Users Harita Seçme Sayfası

Harita seçildikten sonrada hangi parse/parsellerin seçilecei ve ne kadar süreliğine kiralanacağı ve hangi ek hizmetler talep edileceğine dair bir seçim tablosu gelir.

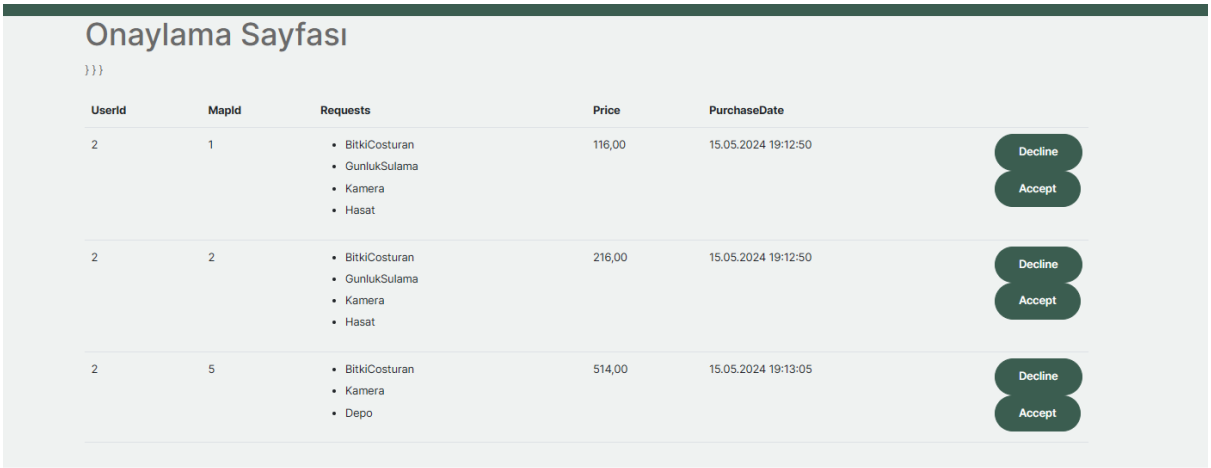


Şekil 4.3.8 Admin Parsel Kiralama Sayfası

Kiralama işlemi tamamlandıktan sonra bilgi ilgili kullanıcının sayfasına istekte bulunduğu anlaşılacak şekilde gelir. Ayrıca admin sayfasında da Onaylama ve Reddetme işlemi olarak görüntülenir.

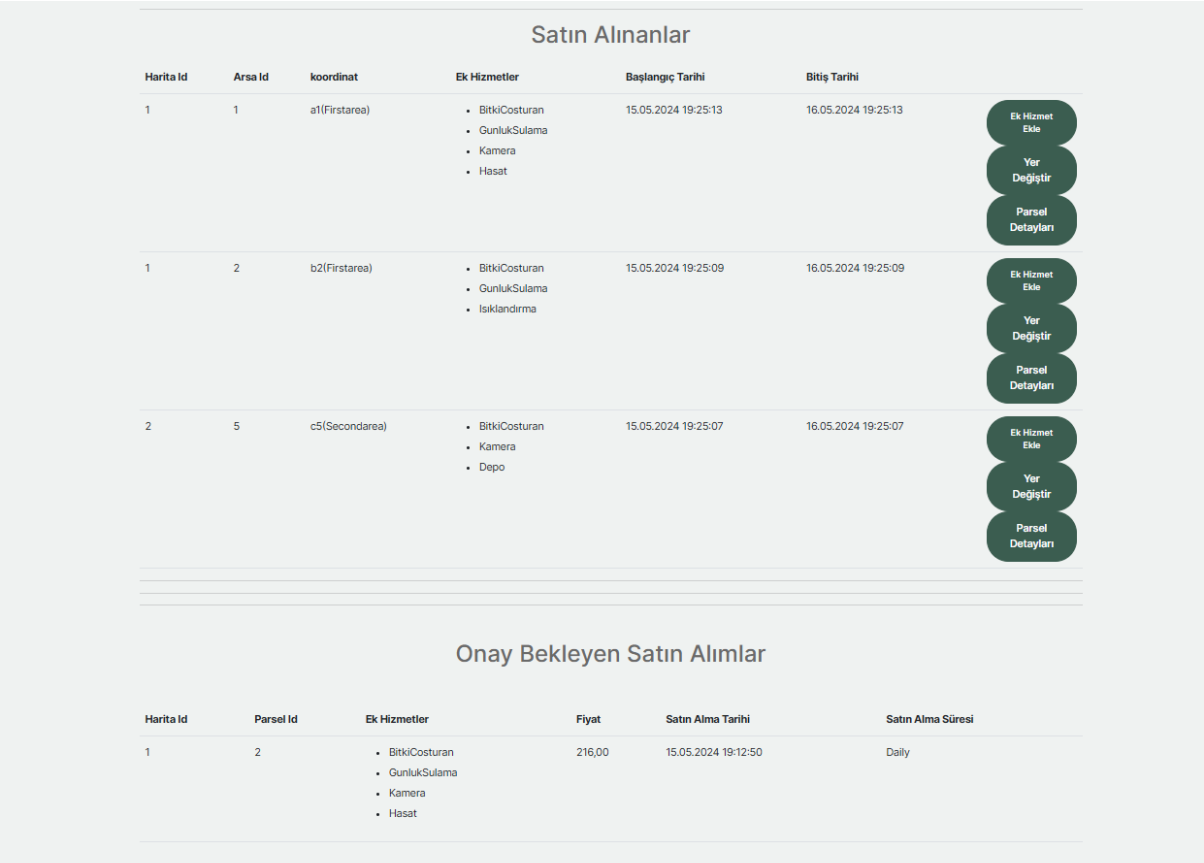


Şekil 4.3.9 User Kiralama İşlemi Sonrası



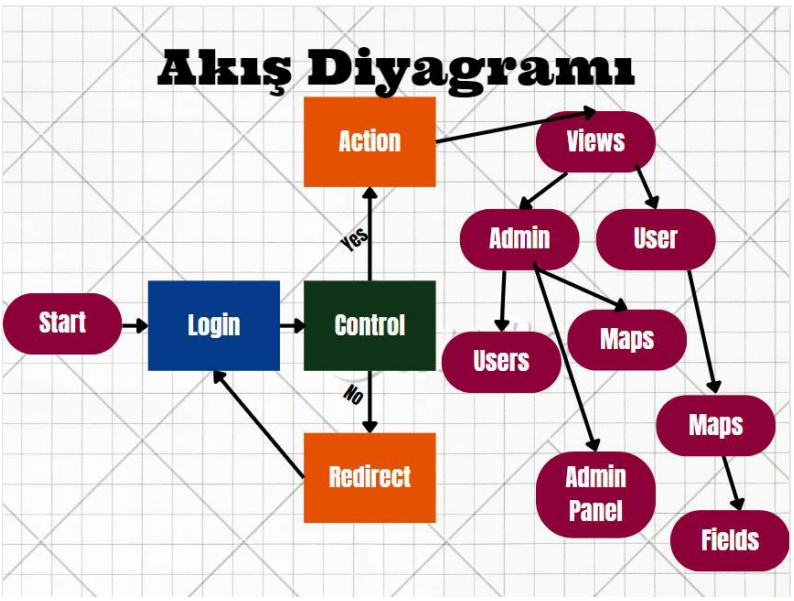
Şekil 4.3.8 Admin Onay Sayfası

Ardından admin sayfasında da Onaylama işlemi yapıldıktan sonra sayfan kaldırılır ve sisteme kiralama işlemi tamamlanmış şekilde girilir. Ayrıca geçerli kullanıcın sayfasında onay bekleyenler listesinden onaylananlar listesi geçer.



Şekil 4.3.9 User Kiralama Onay İşlemi Sonrası

Temel olarak sistemin temel ilerleyisi bu şekildedir. Sistemin akış diagramıda şekildeki gibidir.



Şekil 3.4.1 Akış Diyagramı

5. SONUÇLAR VE ÖNERİLER

İnsanların yoğun olduğu bu dönemde müşterilerin bizzat giderek kiralamalar yapacakları işlemlerde bu hizmetlerin online olarak sunulması çok önemlidir. Bu yüzden olabildiğince web site içerisinde görselleştirmenin ve hizmet sunumunun anlaşılabilir ve kolay olması sağlanmalı ve işlemler basit net olmalıdır.

Sunulan ek hizmetler daha geniş bir çerçevede sağlanmalı böylelikle kiralama yapan müşteriler istedikleri şekilde malzeme ve yardımları hatta belki istedikleri ürün tohumlarını temin ederek hobi bahçelerini kolaylıkla ve arzu ettikleri şekilde kullanabilmelidirler.

Ayrıca haritaların eşit alanlara bölünmesi üzerinde çalışmalar yapılıp daha çok parsel oluşturulabilecek şekilde alanların bölünebileceği algoritmalar üzerinde ilerlenebilir. Bu şekilde bir haritadan daha fazla hobi bahçesi üretmek mümkün olabilir. Bu doğrultuda ilerlemek hem müşteriler için hem de harita sahipleri için daha avantajlı olacaktır.

Piyasada hobi bahçesi kiralama işlemini yapan siteler ve bu alandaki projeler az olduğundan web sitesinde tecrübesiz kullanıcılar için gerekli bilgilendirmeler ve kullanım talimatları sağlanmalı bu şekilde hobi bahçelerinin online olarak kiralanması işlemi insanların arasında yaygınlaştırılarak web sitesine yeni müşteriler kazandırılmalıdır.

KAYNAKLAR

[1] Önder, S., Polat,A,T. (2008). Peyzaj Tasarım Süreci Kapsamında Konya Kenti İçin Yeni Bir Hobi Bahçesi Oluşturulması. Ziraat Fakültesi Dergisi 22 (46): (2008) 18-25.

[2] Bilgin,G.(2021).HOBİ BAHÇELERİ VE UYGULAMADA KARŞILAŞILAN HUKUKİ SORUNLAR. İmar Barışı - Yıkım ve İmar Para Cezası - İmar Suçları | Av.Gökhan Bilgin. https://www.gokhanbilgin.av.tr/yikim-ve-para-cezasi/hobi-bahcesi-imar. 10.11.2023

[3] Bradley,L.(2019).How to Create a Community Garden. NC State Extension. https://content.ces.ncsu.edu/how-to-organize-a-community-garden. 04.01.2024

[4] Kef,F,Ş (2015). Hobi Bahçelerinin Planlanması ve Tasarımı: Konya Karatay Karaaslan Hobi Bahçesi. Bartın Üniversitesi. Fen Bilimleri Enstitüsü Peyzaj Mimarlığı Anabilim Dalı. Bartın

[5] Egli,V, Oliver,M, Tautolo,E. (2016). The development of a model of community garden benefits to wellbeing. Preventive Medicine Reports Volume 3,348-352

[6] Aliağaoğlu,A,Alevkayalı,A (2017). BALIKESİR'DE HOBİ BAHÇELERİ: ÖZELLİKLER VE SORUNLAR. Marmara Coğrafya Dergisi. Volume 35,195-203

[7] Tereshenkov,A.(2017).Dividing a polygon into a given number of equal areas with arcpy.https://tereshenkov.wordpress.com/2017/09/10/dividing-a-polygon-into-a-given-number-of-equal-areas-with-arcpy . 19.11.2023

[8] Turf.js | Advanced geospatial analysis. https://turfjs.org/docs/ . 22.11.2023

[9] .NET Core Nedir? Tarihçesi ve Özellikleri. https://onerbilisim.com/net-core-nedir-tarihcesi-ve-ozellikleri/. 03.01.2024.

[10] Razvalinov A. ASP.NET Core 8 Pros and Cons. https://ukad-group.com/blog/aspnet-core-8-pros-and-cons/. 01.01.2024.

[11] .NET. (2023). https://en.wikipedia.org/wiki/.NET. 02.01.2024.

[12] TOP 10 ADVANTAGES OF .NET CORE. (2018). https://www.fortech.ro/top-advantages-net-core/. 06.11.2023.

[13] What is SQL (Structured Query Language)?. https://aws.amazon.com/tr/what-is/sql/. 23.12.2024.

[14] Microsoft SQL Server (2023). https://tr.wikipedia.org/wiki/Microsoft_SQL_Server. 13.011.2023.

[15] Git Nedir? Git Neden Kullanılmalı?. (2017). https://webmaster.kitchen/git-nedir-git-neden-kullanilmali/. 05.11.2023.

[16] GitHub Nedir ?. https://codigno.com/github-nedir/. 04.01.2024.

ÖZGEÇMİŞ

Muhammed Koç - 190202040
2000 yılında Sivas'ta doğdum. Sivas'ta Kadı Burhaneddin Ortaokulunda ve Gültepe Anadolu Lisesi'nde eğitimini tamamladım. 2019 yılında Kocaeli Bilgisayar mühendisliğini kazandım. Kocaeli Üniversitesinde eğitimime devam etmekteyim.Veri bilimi alanlarında çalışmalar yaptım. Şu anda .NET Core üzerinde çalışmalar yapmaktayım.