MARMARA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

WEB KAZIMA (SCRAPING) TEKNİĞİ İLE WWW'DE ANAHTAR KELİME TABANLI LİTERATÜR TARAMASI UYGULAMASININ GERÇEKLEŞTİRİLMESİ

MUHAMMED ALİ MUTLU

YÜKSEK LİSANS TEZİ

Bilgisayar Mühendisliği Anabilim Dalı
Bilgisayar Mühendisliği Yüksek Lisans Programı

DANIŞMAN
Dr. Öğr. Üyesi Eyüp Emre ÜLKÜ

EŞ-DANIŞMAN
Doç. Dr. Kazım Yıldız

İSTANBUL, 2021

MARMARA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

WEB KAZIMA (SCRAPING) TEKNİĞİ İLE WWW'DE ANAHTAR KELİME TABANLI LİTERATÜR TARAMASI UYGULAMASININ GERÇEKLEŞTİRİLMESİ

MUHAMMED ALİ MUTLU (523618016)

YÜKSEK LİSANS TEZİ

Bilgisayar Mühendisliği Anabilim Dalı
Bilgisayar Mühendisliği Yüksek Lisans Programı

DANIŞMAN
Dr. Öğr. Üyesi Eyüp Emre ÜLKÜ

EŞ-DANIŞMAN
Doç. Dr. Kazım Yıldız

İSTANBUL, 2021

TEŞEKKÜR

Çalışma sürecinde her türlü yol gösterici olan, olumlu tavrıyla beni cesaretlendiren, akademik bilgi birikimiyle çalışmama farklı açılardan bakmamı sağlayan, beraber çalışmaktan ve her zaman öğrencisi olmaktan mutluluk duyduğum değerli danışman hocam Dr. Öğretim Üyesi Eyüp Emre Ülkü'ye sonsuz teşekkür ederim.

Tez sürecimin başlangıcında bana ilham olan ve tüm iş süreçlerimi buna göre düzenlememde yardımcı olan yöneticim ve ağabeyim Sedat Yağcı'ya teşekkür ederim.

Son olarak sadece bu çalışma sürecinde değil hayatımdaki her türlü zorlu yolda beni cesaretlendiren ve yanımda duran eşim Rabia Mutlu'ya, aldığım tüm kararları destekleyen ve başarı basamaklarını tek tek tırmanmayı bana yol olarak gösteren babam Veli Mutlu'ya, sevgisini her an hissettirip çalışkan karakterini bize sevgisiyle aktaran annem Rukiye Mutlu'ya, hayatımda rol model olarak seçtiğim ağabeylerim Fatih'e ve Ahmet'e ablam Gül'e sonsuz teşekkür ederim.

Aralık **2021** Muhammed Ali Mutlu

İÇİNDEKİLER

TEŞEKKÜR i
İÇİNDEKİLER ii
ÖZET iv
ABSTRACT v
KISALTMALAR vi
ŞEKİL LİSTESİ vii
1. GİRİŞ 1
2. LİTERATÜR ÖZETİ 5
3. MATERYAL METOD 12
3.1. Tarama (Web Crawling) 12
3.1.1. Google Search JSON API 13
3.1.1.1. API anahtarı oluşturulması 13
3.1.1.2. Arama motoru oluşturulması 14
3.1.2. Tarama (Crawler) botları 17
3.2. Veri Kazıma (Web Scraping) 17
3.2.1. Yasallık kontrolü 17
3.2.1.1. Yardımcı kütüphaneler 18
3.2.2. Kazıyıcı (Scraper) botları 19
3.2.2.1. Yardımcı kütüphaneler 19
3.2.2.2. Springer veritabanı için geliştirilen algoritma 25
3.2.2.3. ACM veritabanı için geliştirilen algoritma 27
3.2.2.4. IEEE veritabanı için geliştirilen algoritma 28
3.2.2.5. Researchgate veritabanı için geliştirilen algoritma 29
3.2.2.6. Marmara vetis sistemi için geliştirilen algoritma 31
3.2.2.7. ScienceDirect için geliştirilen algoritma 31
3.2.2.8. Scopus için geliştirilen algoritma 31
3.2.2.9. Academia için geliştirilen algoritma 31
3.3. Arkayüz (Backend) Geliştirmeleri 32
3.3.1. Mimari 32
3.3.2. Veritabanı katmanı 33
3.3.2.1. Google Firebase Cloud Firestore 34
3.3.3. Servis katmanı 38
3.3.3.1. Uygulama Programlama Arayüzü (API) nedir? 38
3.3.3.2. FAST API 39
3.3.3.3. Pydantic şema kütüphanesi 39

3.3.3.4. Swagger 40
3.3.3.5. Uvicorn 50
3.3.3.6. Proje iskeleti 51
3.4. Önyüz (Frontend) Geliştirmeleri 57
3.4.1. Uygulama katmanı 57
3.4.1.1. Tek Sayfa Uygulama (Single Page Application) nedir? 58
3.4.1.2. React 59
3.4.1.3. Next 62
3.4.1.4. Redux 62
3.4.1.5. Tailwind 63
3.4.1.6. Proje iskeleti 63
4. SONUÇLAR VE TARTIŞMA 73
4.1. Uygulama Kullanımı 73
4.2. Uygulama Akışı 84
4.2.1. Tarama akışı 85
4.2.2. Kazıma akışı 86
4.3. Çıktılar 87
5. BULGULAR 92
KAYNAKLAR 94
ÖZGEÇMİŞ 97

ÖZET

Web Kazıma (Scraping) Tekniği İle WWW'de Anahtar Kelime Tabanlı Literatür Taraması Uygulamasının Gerçekleştirilmesi

Proje, makale, tez gibi çalışmaların başarıya ulaşmasında rol oynayan etkenlerin başında çalışma öncesinde yapılan literatür taraması gelmektedir. Literatür çalışmasının kapsamlı ve verimli bir şekilde gerçekleştirilmesi yapılacak çalışmanın sağlam temellere oturmasını ve böylelikle yapılan işin kalitesini artırarak güncel ve yenilikçi bir çalışma ortaya konmasını sağlamaktadır. Günümüz teknolojilerinde artık tüm aramalar ve araştırmalar web üzerinden yapılabilmektedir. Arama motorları ise son yıllarda oldukça gelişmiş ve akıllı hale gelmiş durumdadırlar. Arama motorları ile araştırma yapmak ne kadar kolay olsa da kapsamlı ve sistematik bir araştırma yapmak oldukça uzun süren bir süreçtir. Bu uzun süreci hem daha kolay hem de daha verimli ve kapsamlı bir hale getirmek için teknolojideki güncel konular kapsamında kütüphaneler ve geliştirme araçları kullanıma sunulmuştur. Bu geliştirme araçları kullanılarak, araştırma yapmak için gerekli olan süre önemli ölçüde kısaltılabilmektedir.

Literatür taraması, var olan kaynaklar içerisinde belirli bir konunun detaylı biçimde araştırılması ve o konuya ait verilerin sistemli biçimde toplanması sürecidir. Tez kapsamında da literatür taramaları için yapılan araştırmaların daha kısa sürede daha verimli bir şekilde gerçekleştirilmesi hedeflenmiştir. Literatür taraması, sadece akademik çalışmalarda değil, yenilikçi her alanda başvurulması gereken ilk adımlardan birisidir. Herhangi bir konu üzerine çalışmaya başlamadan önce yapılması gereken ilk işlem literatür taramasıdır. Çalışma yapılacak konu ile ilgili daha önceden benzer çalışmalar yapılmış mı, yapıldıysa hangi yöntemler izlenmiş, yapılan işlerde ne tür eksiklikler var gibi konular hakkında fikir sahibi olunmasını sağlar. Literatür çalışması, yapılan çalışmanın telif sorunları ile karşılaşmaması için de oldukça önemlidir. Bu projenin amacı da literatür taraması işleminin hem kolaylaştırılması, hem hızlandırılması hem de daha verimli bir hale getirilmesidir. Google, Bing, Yandex vb. arama web siteleri ile ya da belirli başlı makale sitelerini tek tek gezip o sitenin arama araçlarıyla tarama yapmak yerine kullanımı kolay bir web uygulaması ile tüm WWW ortamını tarayıp ilgili makale sonuçlarının en uygun şekilde kullanıcıya ulaştırılması hedeflenmektedir.

ABSTRACT

Performing Keyword-Based Literature Search App on the WWW with Web Scraping Technique

The literature review that made before the study, plays a big role in the success of studies such as projects, arcticles and theses. Making the literature review in efficient and comprehensive methods builds the project on solid basis. In today's technologies, all searches and researches can be made over the web. On the other hand, search engines have become quite advanced and smart in recent years. Even if it is easy to research with search engines, making comprehensive and systematic research on web can take very long time. In order to make this long process easier, more efficient and comprehensive, libraries and development tools are available within the scope of trend topics in technology. By using these development tools, the research time that a person has to do, which may take days, can be reduced to hours and minutes.

Literature review is the process of investigating a specific subject in detail among available resources and systematically collecting data on that subject. Within the scope of the project, it was aimed to facilitate the researches for literature reviews. Literature review is one of the first steps to be applied not only in academic studies but also in every innovative field. The first thing to do before starting to work on any subject is a literature review. It provides an idea about whether similar works have been done before, what methods have been followed, and what kind of deficiencies are present in the work done. It is also very important for the work done to avoid copyright issues. The aim of this project is to facilitate, accelerate and make the literature review process more efficient. Instead of browsing search websites or specific article sites one by one and browsing with the search tools of that site, it is aimed to scan the entire WWW environment with an easy-to-use web application, and deliver them to the user in the most appropriate way.

KISALTMALAR

API	**:**	Application Programming Interface
ASGI	**:**	Asynchronous Server Gateway Interface
CSS	**:**	Cascading Style Sheet
ES6	**:**	Ecma Script Version 6
HTML	**:**	Hypertext Markup Language
JSON	**:**	Javascript Object Notation
JSX	**:**	Javascript XML
KPI	**:**	Key Performance Indicator
URL	**:**	Uniform Resource Locator
RDF	**:**	Resource Description Framework
REST	**:**	Representational State Transfer
SDK	**:**	Software Development Kit
SPA	**:**	Single Page Application
WWW	**:**	World Wide Web
XML	**:**	Extensible Markup Language

ŞEKİL LİSTESİ

Şekil 1. Uygulama geliştirme adımları 12
Şekil 2. Google Search JSON API için API anahtarının oluşturulması. 14
Şekil 3. Google ile kişiselleştirilmiş arama motoru oluşturulması 15
Şekil 4. Google Search JSON API örnek request 16
Şekil 5. Robots.txt dosyası içeriği 18
Şekil 6. reppy ile kullanılan örnek algoritma 19
Şekil 7. Requests ile örnek kullanım 20
Şekil 8. Web scraping testleri için geliştirilmiş Fake Jobs web sayfası 21
Şekil 9. Fake Jobs sayfası içerisindeki bir işin HTML ağacı 22
Şekil 10. Sayfanın indirilmesi ve BeatifulSoup objesine dönüştürülmesi 23
Şekil 11. ResultsContainer id'li tüm meslekleri içinde barındıran container elementinin seçilmesini sağlayan kod parçacığı 23
Şekil 12. Tüm işlerin parent elementi olan card-content elementinin ve içeriğinin seçilmesi 23
Şekil 13. İş soup objeleri içerisinden title, company ve location bilgilerini içeren elementlerin kazınması 24
Şekil 14. Elde edilen title, company ve location elementlerinin çıktısı 24
Şekil 15. Elde edilen title, company ve location elementlerinin textlerinin kazınması 24
Şekil 16. Hedeflenen çıktının elde edilmesi; her bir iş için title, company ve location bilgileri 25
Şekil 17. link.springer.com için geliştirilmiş kazıma botu 26
Şekil 18. dl.acm.org için geliştirilmiş kazıma botu 27
Şekil 19. ieee.iexplore.org için geliştirilmiş kazıma botu 28
Şekil 20. researchgate.net için geliştirilmiş kazıma botu 30
Şekil 21. Geliştirilen uygulamanın backend mimarisi 33
Şekil 22. Cloud Firestore ile yeni proje oluşturulması 34
Şekil 23. Cloud Firestore ile veritabanı oluşturulması 35
Şekil 24. Cloud Firestore veritabanı için private key oluşturulması 36
Şekil 25. Cloud Firestore veritabanının uygulamaya entegre edilmesi 37
Şekil 26. Veritabanı Diyagramı 37
Şekil 27. Pydantic ile oluşturulmuş şema örneği 40
Şekil 28. Swagger API dökümantasyonu 41
Şekil 29. Crawler API Get Search Details kullanım örneği 42
Şekil 30. Crawler API Start Searching By Keyword kullanım örneği 44
Şekil 31. Crawler API Get Search Results kullanım örneği 45
Şekil 32. Scraper API Get Scraping Details kullanım örneği 47
Şekil 33. Scraper API Start Scraping By Id kullanım örneği 48
Şekil 34. Scraper API Get Scraping Results kullanım örneği 49
Şekil 35. Uvicorn ile çalışacak proje main dosyası 50
Şekil 36. Arkayüz proje iskeleti 51
Şekil 37. Crawler API için geliştirilen Python şeması 52
Şekil 38. Scraper API için geliştirilen Python şeması 53
Şekil 39. Api kod örneği 54
Şekil 40. Servis kod örneği 55
Şekil 41. Service Reference kod örneği 56
Şekil 42. Uygulama arayüzü 58
Şekil 43. JSX formatında yazılmış kod örneği 60
Şekil 44. React component örneği 60
Şekil 45. React componentinin props ile kullanılması 61
Şekil 46. React componentinin state ile kullanılması 62
Şekil 47. Frontend uygulama iskeleti 64
Şekil 48. Layout container componenti 65
Şekil 49. Layout componentinin kullanım örneği 66
Şekil 50. İç içe component yapısı örneği 67
Şekil 51. Utilization dosyası örneği 68
Şekil 52. Redux store oluşturulması 68
Şekil 53. Redux action örneği 69
Şekil 54. Redux reducer örneği 70

Şekil 55. Redux store tarayıcı görünümü *71*
Şekil 56. Frontend service örneği *72*
Şekil 57. Smart Literature Search Giriş Sayfası *73*
Şekil 58. Smart Literature Search Anasayfa *74*
Şekil 59. Smart Literature Search yeni arama oluşturma *76*
Şekil 60. Smart Literature Search yeni arama başlatma *77*
Şekil 61. Smart Literature Search devam eden arama *78*
Şekil 62. Smart Literature Search tamamlanmış arama *79*
Şekil 63. Smart Literature Search arama sonuçları - I *80*
Şekil 64. Smart Literature Search arama sonuçları – II *81*
Şekil 65. Smart Literature Search makale detayı *82*
Şekil 66. Önyüz uygulama akışı *83*
Şekil 67. Uygulama akış grafiği *84*
Şekil 68. Tarama (crawling) akışı *85*
Şekil 69. Kazıma (scraping) akışı *86*
Şekil 70. Crawler API, tarama *88*
Şekil 71. Crawler API, tarama sonuçları *89*
Şekil 72. Scraper API, kazıma *90*
Şekil 73. Scraper API, kazıma sonuçları *91*

1. GİRİŞ

Proje, makale, tez gibi çalışmaların başarıya ulaşmasında rol oynayan en önemli etkenlerin başında çalışma öncesinde yapılan literatür araştırması gelmektedir. Literatür taraması, var olan kaynaklar içerisinde belirli bir konunun detaylı biçimde araştırılması ve o konuya ait verilerin sistemli biçimde toplanması süreci olarak tanımlanmaktadır [1]. Bir başka ifade ile de araştırma konusunun yol haritasının belirlendiği süreç olarak da tanımlanmaktadır. Literatür araştırması, araştırma yapılacak konunun daha önceden çalışılıp çalışılmadığını, konu üzerine çalışma yapan kişileri, konunun hangi yönlerden ve nasıl bir yöntem ile ele alındığını, yapılacak çalışmanın literatürdeki hangi soruna çözüm olabileceği gibi soruların cevaplarına ulaşmamızda yardımcı olmaktadır.

Literatür taramasına başlamadan önce yapılacak ilk işlem araştırma konusundaki problemin ve anahtar kelimelerin net olarak belirlenmesidir. Daha önce yapılan çalışmalarda konunun hangi açılardan detaylı incelendiğini ve hangi yönlerden eksik kaldığını tespit etmek için oldukça önemlidir. Bir sonraki adımda ise, yapılacak olan literatür taraması için hangi araçların kullanılacağının belirlenmesi gerekmektedir. Yapılacak olan bir başka işlem de konu kapsamında hangi kaynakların kullanılacağının ve bu kaynaklara erişim durumunun belirlenmesidir. Bu sayede tarama için hedef kaynaklar ve metotlar saptanmaktadır. Tarama yaparken karşılaşılan kaynakların güvenilirliği de yapılan oldukça önemlidir. Özellikle, kaynağın yazarı ve eserin yayımlandığı indeks kontrol edilerek herkes tarafından kabul görmüş, bilinen ve güvenilir bir kaynak olduğu tespit edilmelidir. Araştırma konusu ile ilgili literatür taraması gerçekleştirirken, tüm kaynaklardaki tüm çalışmalar yerine bilimsel içeriğe sahip kaynaklara ulaşılması çok daha önemlidir. Konuyla ilgili yazılmış ve web ortamında kolayca ulaşılabilen kitaplar, makaleler, tezler ve ansiklopediler bu amaçla kullanılabilmektedir. Bu noktada genel olarak herkese hitap eden popüler yayınlardan ziyade sadece o konu üzerinde özel olarak çalışılmış, profesyonel bir kitle için yazılmış akademik yayınlar önceliklendirilmelidir. Bu tespit de o kaynağın yayınlandığı ortam ve incelenme sayısına bakılarak yapılabilmektedir. Problem ve bu problemi incelemek için kullanılacak güvenilir araçlar belirlendikten sonra yapılacak taramanın planlanıp alt parçalara bölünmesi ve sınıflandırılması gerekmektedir. Her bir alt başlıktaki ulaşılan kaynaklardaki soruna yaklaşım, üretilen çözüm konularında birbiri ile karşılaştırılması ve

benzerliklerinin tespit edilmesi taramayı bu şekilde sınıflandırma ve alt başlıklara ayırma ile mümkün olacaktır. Tamamlanan literatür taramasına literatürdeki durumu, eksiklikleri ve sonuçları özetleyen bir sonuç bölümü eklenmelidir. Tarama yapılırken elde edilen, faydalanılmış kaynakların bilgilerine yer vererek eleştirel bir yaklaşımla güncel konular ve yöntemler üzerinden değerlendirme yapılmalıdır. Sonuç kısmı, yapılan literatür taraması ile tespit edilen eksiklikleri ve bu eksikliklerin nasıl giderileceğini ifade edeceği için araştırma konusunu destekleyici niteliktedir. Sıralanan bu literatür taraması aşamaları sıra ile doğru bir şekilde izlenirse yapılacak çalışmanın literatürde tam olarak konumu belirlenmiş olacaktır. Çalışma tamamlanana kadar da yapılmış literatür taramasının üzerinden sıkça geçilerek güncelliğini ve konuya bağlılığını koruduğundan emin olunmalıdır.

Son on yılda bilgisayar bilimi alanında yapılan çalışmalar incelendiğinde, makine öğrenmesi, derin öğrenme, yapay zeka, büyük veri gibi anahtar kelime başlıklarında oldukça fazla araştırma yapıldığı gözlemlenmektedir. Bu alanlarda yapılan çalışmaların temelini veri oluşturmaktadır. Yapılacak çalışmayı desteklemesi ve bu çalışmada kullanılabilmesi için verinin doğru şekilde toplanması, işlenmesi ve yapısal olarak düzenli hale getirilmesi önemli noktaların başında gelmektedir. Günümüz teknolojileriyle geliştirilmiş arama motorları sayesinde elde edilmesi istenilen bilgiye ulaşılması ücretsiz hale gelmekle birlikte oldukça hızlanmış ve kolaylaşmıştır. Fiziksel kütüphanelerin yerini alan sanal kütüphaneler arama motorlarının ucunda kullanıcıya hizmet vermektedir. Literatür taraması için de ihtiyaç duyulan kaynaklara web ortamındaki bu arama motorları sayesinde ulaşılabilmektedir. Arama motorları her ne kadar bilgiyi hızlı ve kolay bir şekilde kullanıcıya ulaştırsa da arama sonucunda dönen web sitelerinin tamamını gezmek, her site için içerisindeki bilgiyi elde etmek süre gerektiren zor bir süreçtir. Bu şekilde yapılan literatür taramaları da uzun sürmekte, yapılacak çalışmaya başlama tarihini de uzatmaktadır. Bununla birlikte siteleri gezerken gözden kaçan çalışmaya değer katacak, önemli bilgiler de olabilmektedir.

Kullanıcıların, literatür araştırmalarını daha hızlı ve daha verimli yapabilmeleri için çeşitli yöntemler geliştirilmiştir. Bu yöntemlerin temelinde web crawling (web üzerinde tarama) ve web scraping (web üzerinde veri kazıma) teknikleri yatmaktadır. Web üzerinde arama yapması ve tüm WWW ortamını tarayıp ilişkili site adreslerini elde etmek için kodlanmış botlara web tarama (crawling) botları, elde edilmiş site adreslerini tarayıp

anlamlı bilgileri çıkarmaya yarayan botlara da web kazıma (scraping) botları denmektedir. Geliştirilen bu botların uygun algoritmalarla birleştirilip sistematik ve otomatik hale getirilmesi ve anlamlı verilerin çıkarılması işlemlerinin tamamına web tarama ve kazıma (crawling) denmektedir. Bu arama araçları ilgili veriyi anlamlı bir şekilde çıkararak yapısal veritabanları oluşturmaya yardımcı olmaktadır [2-4].

Web tarama ve kazıma teknikleri, veriyi çıkarmayı ve işlemeyi kolaylaştırsa da bu işlemlerin yasallığı ve etikliği sorgulanmıştır [5]. Özellikle kişisel verilerin ya da finansal bilgilerin web sitelerinden çalınması konusunda çekinceler ortaya çıkmıştır. Ethical Decision-Making ve Internet Research committee tarafından Web kazıma yöntemlerinin kullanım rehberi olması için bir çalışma yapılmıştır [6]. Bu çalışmada da açıkça ortaya koyulduğu gibi web scraping ile finansal ya da kişisel bilgilerin çıkarılması yasal olarak ihlal kabul edilmiş ve tüm ülke mahkemelerinin bunun karşısında olduğunu belirtmişlerdir. Web site sahipleri de sitelerinde "website/robots.txt" yoluna koyduğu dosya ile sitelerinde botlar ile ziyaret edilmesine izin verdiği ve ziyaret edilmesini yasakladığı site kısımlarını belirtebilmektedir. İhlal anında yasalar bu dosyayı da değerlendirmektedir. Bunun dışında kişisel bilgi ihlali olmadığı ve sitenin "robots.txt" dosyası dikkate alınarak yapıldığında web scraping oldukça kullanışlı ve yasal bir yöntemdir.

Literatür çalışmasının kapsamlı ve verimli bir şekilde gerçekleştirilmesi yapılacak çalışmanın sağlam temellere oturmasını ve böylelikle yapılan işin kalitesini artırarak güncel ve yenilikçi bir çalışma ortaya konmasını sağlamaktadır. Literatür taraması, sadece akademik çalışmalarda değil, hemen hemen tüm alanlarda yapılan yenilikçi çalışmalarda gerçekleştirilen ilk adımdır. Yapılan çalışmanın telif, lisans ihlalleri gibi sorunlarla karşılaşmaması için de oldukça önemli bir süreçtir. Bu tez çalışmasında, web tarama ve kazıma yöntemlerini yasal çerçevelere dikkat ederek mühendislik alanında yaygın olarak kullanılan makale web sitelerinden veri çıkarma uygulaması geliştirme amacıyla kullandık. Geliştirilen uygulama ile hızlı ve otomatik bir şekilde basit bir arayüz üzerinden girilen anahtar kelimeyi, arama kriterlerini de dikkate alarak WWW ortamında önceden belirlenmiş popüler makale web sitelerinde tarayıp, bulduğu yayınların özet kısımlarını ve anahtar kelimelerini elde eden bir uygulama geliştirdik. Bu sayede kişi akademik bir çalışma yapmadan önce yapacağı literatür taramasını bu uygulama üzerinden çok daha kısa sürede kapsamlı ve verimli bir şekilde gerçekleştirebilecektir.

Böylelikle, çalışma yapacağı alanla ilgili anahtar kelimeleri içeren çalışmaların özetlerine ve diğer anahtar kelimelerine uygulama üzerinden hızlı bir erişim sağlayabilecektir. Sonuç olarak, bu teknolojiler kullanılarak, literatür taraması için bilgiyi elde etme süresinin minimuma indirilmesi ve verimliliğin en üst düzeye çıkarılması çalışmanın temel amacını oluşturmaktadır.

2. LİTERATÜR ÖZETİ

Günümüzde internet kullanımının giderek yaygınlaşması sebebiyle web siteleri üzerinden gerçekleştirilen kazıma (scraping) işlemlerine yönelik çalışmalar da her geçen gün yaygınlaşmaktadır. Verinin öneminin giderek artması ile birlikte web sitelerinden anlamlı verilen çıkarılması işlemleri de giderek yaygınlaşmakta ve önem kazanmaktadır. Son yıllarda, web kazıma işlemlerine yönelik yapılan akademik çalışmalar da giderek popülerlik kazanmaktadır.

Tropikal hastalıkların ciddiyetine vurgu yapılan Sumatera Utara Üniversitesi'nde çalışılmış bir makalede, Endonezya'da yaşayan insanlarının tropikal hastalıklara karşı ilaç ve tedavi yöntemleri gibi çözümler aramak için arama motorlarına güvendiğini saptamışlardır. Fakat geleneksel arama motorlarının hastalıklara çözüm ararken hastalığın eş anlamlı kelimesini de hesaba katarak sonuç getirdiğini dile getirmişlerdir. Bu sorunun üstesinden gelmek için çözüm olarak anlamsal olarak da doğru sonuçlar içeren bir arama motoru geliştirilmesi gerektiğine değinmişlerdir. Anlamsal web ismini verdikleri bu çözüm verinin eş anlamlarıyla birlikte ilişkisini de Resource Description Framework (RDF) denilen form ile ortaya koymaktadır. Yapılan ilk araştırmalar Endonezya'da sağlık bakanlığı web sitelerinin RDF formatta dosyalar sunamadığı için başarısız olmuştur. Daha sonrasında ilgili çalışma web kazıma teknolojileri kullanarak RDF formatında sağlık bilgileri üreten bir platforma dönüştürmeyi hedeflemiştir. Bu yolda popüler sağlık web sitelerinden kelime toplamak ve çıkarmak, ontoloji ile terimler arasındaki ilişkiyi kurmak ve bunların sonucunda RDF formunda bilgiler üretmek için web kazıma teknolojileri kullanılmıştır [7].

Newsone isminde Hindistan Agni Teknoloji Koleji'nde ulusal ve uluslararası kaynaktan en son haber güncellemelerini toplayıp bunları kısa ve net bir şekilde özetleyen bir platform geliştirilmiştir. Güncel haberleri bulmak için harcanan zaman ve efordan kazanç sağlayan bu uygulama ile hızlı bir şekilde sonuca ulaşılması hedeflenmiştir. Bunu yaparken de web kazıma ve tarama yöntemlerinden faydalanılmıştır. Uygulama yöneticileri RSS adreslerini veritabanına kaydeder. Uygulama içerisindeki web kazıma botları da ilgili RSS sitelerini dinamik ve periyodik bir şekilde gezerek ilgili içerikleri kaydeder. Bundan sonraki aşamada da çıkarılmış veriler kategori ve URL tabanlı ayrıştırarak dökümante edilecek ve bir model oluşturulacaktır. Bu model sayesinde de kullanıcılar ilgi ve alakalarına göre ya da kategori tabanlı arama yaparak haberlere erişim

sağlayabilmektedirler. Bir başka ifade ile geliştirilmiş bu uygulama ile okuyucu ulusal ve dünya çapında 100'den fazla lisanslı ve güvenilir haber sitesindeki içeriklere basit ve hızlı bir şekilde ulaşabilmektedir [8].

Web kazıma teknolojileri günlük hayattaki yeme alışkanlıkları üzerine bir çalışmada da kullanılmıştır. Ramaiah Teknoloji Enstitüsü'nde yapılan çalışmada insanların yoğun yaşam tarzının getirileriyle birlikte düzensiz yeme alışkanlıklarının çoğaldığı belirtilmiştir. Sağlıklı bir yaşam için diyet yemeklerinin tercih edilmesi gerektiği vurgulanmıştır. Yemek tarif sitelerinde tarif yazan şeflerin ilgili yemeğin içeriğindeki malzemelerin yan etkilerini çoğu zaman önemsemeden bu tarifi yazdığından bahsetmişlerdir. Yemek malzemesi tabanlı arama yapabilmek ve güvenilen diyet malzemesi içeren tariflere ulaşılabilmesi için ilgili çalışma yapılarak bir algoritma geliştirilmiştir. Python dilinde web kazıma teknolojileri kullanılarak yemek tarifi web sitelerinden sağlıklı diyet yemekleri tariflerini çıkarıp MongoDB veritabanına kaydeden akıllı şef isminde web ve mobil tabanlı uygulama geliştirmişlerdir [9].

Web sitelerinden web kazıma teknolojileriyle veri çıkarılırken uygulanan yöntemlerin zayıf yönlerine değinen Erdinç Uzun, yaptığı çalışmada bu zayıflıkların üstesinden gelebilmek için geliştirdiği kendi yöntemine değinmiştir. Klasik web kazıma uygulamalarında veri elde edilmeye çalışılırken ilgili web sitesinin HTML ağacının çıkarılıp içerisinden gezilerek sonuca ulaşıldığından ve bunu yaparken de tüm ağacı oluşturma ve gezme süresinin oldukça uzun olup sonucun düşük performanslı bir şekilde üretildiğinden bahsetmiştir. Kendi ürettiği UzunExt yöntemi ile bu şekilde ağaç modeli üretmeye gerek kalmadan String dizileri kullanarak web sitesinden istenilen sonucun çıkarabilmeyi önermiştir. Bu yönteme göre ilk olarak kelime deseni aratılmış, sonrasında bu desene göre kapatılan HTML elementlerinin sayısı hesaplanmış ve son olarak da bu kelime deseni ile alakalı tüm bilgiler çıkarılmıştır. Bu methodun geleneksel kazıma methodlarına göre performans olarak 60 kat daha hızlı olduğu belirtilmiştir [10].

David ve Sandeep'in çalıştığı makalede standart bir bilgi araştırması yapılırken kelimenin kökeni, ilişkilendirilmesi ve kullanıldığı yerlerin öneminden bahsedilmiştir. Web kazıma teknolojileri kullanırken de bu ilişkilerin kaybedilmeden kazıma yapılması gerektiği vurgulanmıştır. Bu teknikten de üç adımda bahsetmişlerdir: Web kazıyıcı ilgili linklere gider ve verileri çeker, veriler anlamlaştırılır ve son olarak da csv dosyasına yazılır. Web kazıma teknolojileri için de kütüphaneler ve araştırma alanı geniş olduğu için Python dili

tercih edilmiştir [11].

Türkiye Fırat Üniversitesi'nde Fatih Ertam yaptığı çalışmada büyük bilgi deposu olan internet sitelerinde bulunan verinin toplanıp işlenmesinin önemine değinmiştir. Bu çalışmada da web kazıma yöntemleri kullanılarak Türkçe bir haber ajansı sitesinde bulunan kategorik haber başlıkları ve özetleri toplanmış ve toplanan veriler üzerinde tensorflow tabanlı derin öğrenme yöntemleri ile vektörel sınıflandırma yaklaşımlarından "one hot encoding" işlemi ile test verileri sınıflandırılmıştır. Sınıflandırma sonucunda %90 üzerinde bir doğruluk oranına ulaşıldığı görülmüştür [12].

Mahidol Üniversitesi'nde Lalita Junjoewong ve arkadaşlarının kaleme aldığı makalede de insanların web sayfalarında kendilerine uygun promosyon ve kampanyaları bulmak için çok zaman harcadığından ve bu uğraşın zorluklarından bahsetmişlerdir. Geliştirdikleri "ProCircle" mobil uygulaması ile promosyon haberlerini tek bir platformda web kazıma teknolojileri ile toplamışlardır. Yaptıkları kitle kaynak çalışması ile de topladıkları promosyon haberlerini desteklemişlerdir. Bu destek sayesinde de kullanıcıların kitlesel olarak promosyonları taramasına olanak sağlamışlardır [13].

Rabiyatou Diouf ve arkadaşları web kazıma hakkında bir veya daha fazla web sitesinden bilgilerin elde edilip excel, database vb. araçlarına çıktı almak için birçok uygulamanın kullanımda olduğundan bahsetmişlerdir. Ortaya koydukları makalede varolan bu otomatize edilmiş farklı web kazıma uygulamalarını ve yaklaşımlarını yeniden gözden geçirmekle birlikte bu uygulamaların kullanım alanlarına da değinmişlerdir [14].

2020 IEEE Avrupa Sempozyumu'nda Junjoewong ve arkadaşları tarafından yayımlanan makalede web kazıma tekniklerine karşın sitelerin savunma mekanizmaları incelenmiştir. Bazı web site yöneticileri savunma tekniği olarak siteden veri toplamayı engellemeye çalışırken bazılarının da veri toplamayı zorlaştırıcı yöntemlere başvurduğuna değinilmiştir. İlgili makalede başvurulan savunma teknikleri incelenip bunlara karşı geliştirilen savunma kırıcı teknikler de incelenmiştir. Bu sayede hangi savunma tekniklerinin ne düzeyde başarılı olduğu tespit edilmeye çalışılmıştır. Yapılan çalışma sonuçlarına göre de bu savunma tekniklerinin her ne kadar kazımayı yavaşlatsa da tam bir koruma sağlamadığı saptanmıştır. Bu güvenlik önlemlerinin aslında bir şekilde aşılabildiği ve siteden kazıma yapılabildiğini kanıtlamışlardır [15].

Günümüz popüler sosyal medya platformlarından olan Instagram'ın ürün tanıtım ve pazarlama özellikleri üzerine Endonezya'da web kazıma teknolojileri ile bir çalışma

yapılmıştır. Ürünü almaya çalışırken güvenmek için Instagram tarafından onaylı hesap ya da ürün olması gerektiğine ve Instagram'da çok fazla sayıda sahte takipçiye sahip hesabın bulunduğuna dikkat çekilmiştir. Kullanıcıların hangi hesabın sahte olduğunu çoğu zaman anlayamadığı için bu tespiti yapan bir sisteme ihtiyaç duyulduğuna vurgu yapılmıştır. Bu çalışmada Instagram hesabı üzerindeki takipçi sayısı, beğeni sayısı, yorum sayısı ve güncel gönderi sayısı gibi parametreler dikkate alınmış ve web kazıma teknolojileri kullanılarak hangi parametrelerin ilgili hesabın gerçek bir pazarlayıcı hesabı olduğunu tespit edebilmek için yeterli olduğuna karar vermesi sağlanmıştır. Çalışmanın başarım oranının % 75 olduğu belirtilmiştir [16].

Hindistan'da yapılan bir araştırmada internette yer alan haberlerin sahte olabileceğine vurgu yapılarak bir çalışma yapılmıştır. İlgili çalışmada Derin Öğrenme ve Doğal Dil İşleme metotları yardımı ile sahte bilgi ve haberleri tespit eden bir model önerilmiştir. Web kazıma yöntemleri ile haber sitelerinden elde edilmiş haber içerikli veri seti Derin Sinir Ağları ile eğitilmiş ve Doğal Dil İşleme kullanılarak ilgili belgelerdeki kelimelerin korelasyonu bulunmuştur. Bu korelasyonlar derin sinir ağı için başlangıç değer görevi görüp haberin sahte olup olmadığını anlamaya yaradığı belirtilmiştir. Recurrent Neural Network (Tekrarlayan Sinir Ağı), Long Short-Term Memories(Uzun Kısa Sürekli Anılar) ve Grated Recurrent Units (Sıralı Tekrarlayan Birimler) yöntemleri kullanarak sınıflandırmanın yapıldığı belirtilmiştir. Sahte haberler eğitim öğrenme modeline göre tespit edilebileceğinden iyi bir eğitim setinin zorunlu olduğunu vurgulayarak bu çalışmadaki çıktıların bu isteği karşıladığını ve başarılı sonuçlar verdiği vurgulanmıştır [17].

Andrew ve arkadaşlarının Trinity Western Universitesi'nde yaptıkları çalışmada kimlik hırsızlığı konusu üzerine yoğunlaşmışlardır. Kimlik avcıları avlama yöntemleri ile web üzerindeki sitenin amacını kötüye kullanıp kullanıcıların bilgilerini çalmak için yöntemler geliştirmişlerdir. Geçmişte kimlik avının yapılırken nelerin güvenlik açığı olduğu ya da nelerin bunları etkilediği konusunda bir çok makalenin de kaleme alındığını belirten Andrew ve arkadaşları bu çalışmalarında güvenlik açığına nelerin sebep olduğunu incelerken web sitelerindeki kimlik avı veri setlerinin yanı sıra sitenin içeriğini de elde edip ikisi arasındaki bağlantıyı bulmayı önermişlerdir. Daha önceki kimlik avı algılama yöntemlerinin çoğunun daha önce tespit edilmiş kimlik avı yöntemlerine bakılarak araştırıldığı için çok doğru sonuç vermediğini belirtmişlerdir. Çalışma

sonucunda “Phishing-Detective” (kimlik avı dedektifi) isminde bir framework (yöntem) geliştirmişlerdir. İlk olarak web kazıma teknolojileri kullanılarak legal ve kimlik avı web sitelerinin tamamının içeriğinin elde edilmesi için bir web tarayıcısı geliştirmesi yapmışlardır. Bahsedilen yöntemler ile kazıyıcı aracılığıyla toplanan veri setini veri madenciliği araçları kullanarak analiz etmişlerdir. Bu çalışma üzerine kurulmuş vaka çalışması henüz tamamlanmamıştır fakat yapılan bu yöntemlerin kimlik avı tespitinde nasıl etkili olacağını çalışma sonucunda açıklanacağını belirtmişlerdir [18].

Piyasada çok fazla konuda çok fazla sayıda bulunan öneri sistemleri konusunda Mumbai, Hindistan’da bir çalışma yapılmıştır. Bir kitap okunduktan sonra hangi kitabın okunacağına karar verilebilmesi için web tabanlı bir öneri sistemi üzerinde çalışılmıştır. Kişiye özel olarak farklı yaş, cinsiyet ve ülkelerdeki kullanıcılar için filtrelenebilir bir şekilde hibrit kitap tavsiye sistemi için bir model oluşturulmaya çalışılmıştır. Web kazıma yöntemleri ile kazınmış veriler ve kişinin daha önce okuduğu kitap verilerini bir arada kullanarak bir model ortaya koyduklarını belirtmişlerdir [19].

Sağlık üzerine Karunya Teknoloji Enstitüsü’nde 2019 yılında bir çalışma yapılmıştır. Çalışmaya göre Dünya Sağlık Örgütü Dünya’da en çok ikinci sıradaki ölüm sebebinin kanser olduğunu raporladığı vurgulanmıştır. Kanserle boğuşan insanların öfke, korku, depresyon gibi negatif duyguları çokça yaşayıp hastalıklarına olumsuz etki oluşturduklarını vurgulamışlardır. Etraftaki insanlar da kanser olan insan ile doğru iletişim kuracağından çoğu zaman şüphe duyduğundan ilgili çalışmayı yapan Karunya Üniversitesi bu insanların rahatlıkla konuşup kafasındaki tüm soruları sorabileceği ve aynı duyguları yaşayan insanlarla iletişim ihtiyacını giderebileceği bir chatbot (mesajlaşma botu) tasarımı üzerine çalışmışlardır. Sadece kanser olan insanların kullanabileceği bu chatbot kanser hakkında tedavi, semptomlar, yaşanılan psikolojik rahatsızlıklar vb. gibi birçok konuda sorulara cevap verebilecek şekilde tasarlanmıştır. Mesajlaşma botu geliştirilirken web üzerinde bu konu hakkında geniş bilgi kaynağı olan birçok kanser hastalığı forumu web kazıma yöntemleri ile kazınıp elde edilen veriler ile eğitilmiştir. Sentiment analizi yöntemleri kullanılarak kullanıcının o anki duygu durumunu tespit edip botun insan gibi davranması sağlanıp kullanıcıların ruhsal olarak da rahatlatılması hedeflenmiştir [20].

Web üzerinde kullanıcılara özel ihtiyaçları karşılamak için birçok uygulama geliştirildiğine vurgu yapılan bir çalışmada dağıtık ve paralel çalışan sistemlerin

değerlendirmesi yapılmıştır. Farklı farklı yöntemlerle çalışan kişiye özel ihtiyaçları tespit etmeye yarayan bu tarz bir sistem için web kazıma yöntemleriyle sosyal ağları taramaya odaklanmışlardır. Bunu yaparken de kişinin daha önce ziyaret ettiği web ortam ve uygulamaları taranarak kişiselleştirmeyi daha yüksek seviyede yapan bir değerlendirme aracı önermişlerdir [21].

Dünya'da popüler olarak kullanılan Instagram sosyal medya uygulamasının Endonezya'da en çok kullanılan sosyal medya uygulaması olduğu vurgulanan çalışmada Instagram'a yüklenmiş videolarda görünen "video görüntülenme sayısı" ile beğeni sayısı arasındaki ilişki saptanmaya çalışılmıştır. Daha çok izlenme daha çok beğeni sayısını beraberinde mi getirir sorusunun üzerine yoğunlaşılmıştır. Bunun için de web kazıma teknolojileri kullanılarak Instagram üzerine yüklenmiş İslami içerikli videolardaki video görüntülenme sayısı, beğeni sayısı elde edilmiş ve aralarındaki ilişkinin güçlü olup olmadığını anlamak için de Pearson korelasyon yöntemi kullanarak hesaplamalar yapılmıştır. Bu analizin sonucuna göre de vaizlerin günümüz çağındaki insanların ilgisini çeken İslami konular üzerinde daha çok yoğunlaşabileceği ve dini olarak daha çok ilgi toplayabileceği belirtilmiştir [22].

İnsanların dünya üzerinde katılabileceği etkinliklerin üzerinde Türkiye'de yapılan bir çalışmada bu etkinliklerin akıllı bir öneri sistemi ile sosyal medyada insanlara duyurular yapılırsa daha çok katılım sağlanabileceği belirtilmiştir. Bu çalışmada da sosyal medya ile insanlara öneriler yapmaya yarayan ve birçok sosyal medya sitesine entegre edilebilen bir sistem geliştirilmiştir. Sistem çalışma mantığı olarak web kazıma teknolojileri ile web üzerinde etkinlikleri kişinin daha önce katıldığı veya katılmayı planladığı etkinlikleri de baz alarak elde eder. Kişinin coğrafik lokasyonu ve sosyal çevresi de dikkate alınarak çalışan bu sistem uygun olan etkinlikleri kişiye önerecek şekilde çalışmaktadır [23].

Yeni teknolojilerin çoğalmasıyla internet kullanıcılarının ve ürettikleri verilerin günden güne artmasına değinilen bir çalışmada yapısal olarak düzensiz verinin web kazıma yöntemleri ile basit bir şekilde elde edilebileceği gibi elde ederken verinin aşırı büyük olması sebebiyle depolama kısıtı, hesaplama zorluğu gibi sorunlarla karşılaşabileceğine vurgu yapılmıştır. Bu çalışmada ilgili sıkıntıları gidermek için bulut tabanlı çözümlerin kullanılabileceği önerilmiştir. Amazon'un sağladığı "Elastic Compute Cloud" ve "DynamoDB" hizmetleriyle bulut tabanlı mimariyle web kazıma teknolojisi geliştirilmiştir. Çalışma içerisinde geleneksel kazıma yöntemleri, Amazon mimarisi ile

kazıma yöntemi ve diğer bulut tabanlı kazıma yöntemleri kıyaslanmıştır [24].

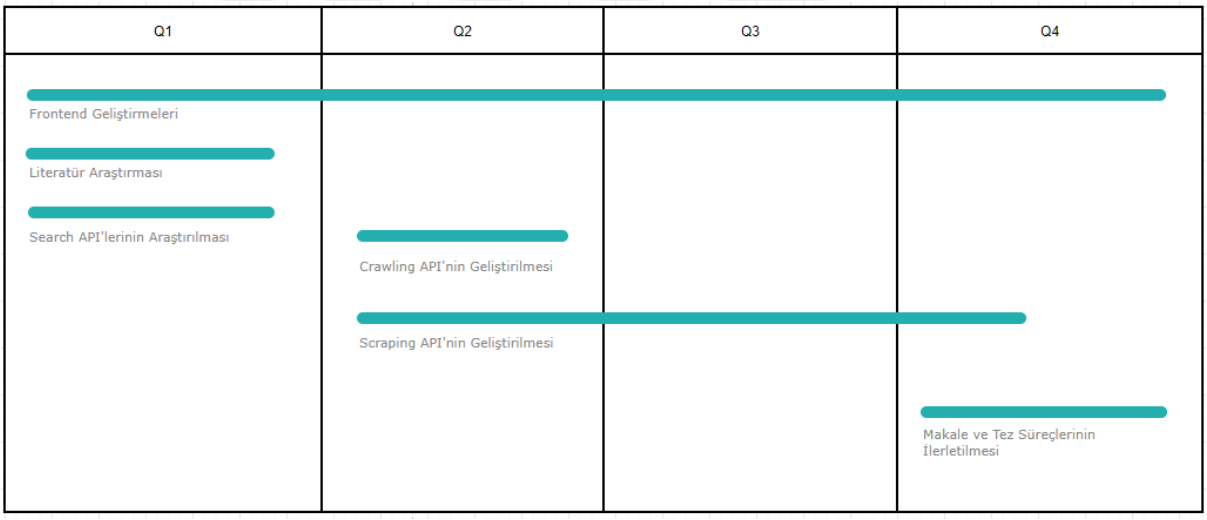
Lunn ve arkadaşlarının yaptığı araştırmada web kazıma ve doğal dil işleme yöntemlerinin bilgisayar mühendisliği eğitiminde karmaşık problemlere nasıl çözüm üretebileceği üzerine yoğunlaşılmıştır. Kamuya açık web sayfalarından büyük miktarda verinin veri kazıma teknikleri ile çıkarıp bu bilgiye benzer daha fazla veriyi tahminlemenin yapılabileceği önerilmiştir. Bu işleme ek olarak da doğal dil işleme yöntemleri kullanarak bu büyük miktardaki verilerden göze çarpan bilgileri elde etmek için nasıl kullanılabileceği üzerine çalışılmıştır. Bu teknikleri pratiğe dökerken de bilgisayar mühendisliği öğrencileri için iş piyasasındaki mevcut eğilimlerin incelendiği bir uygulama ortaya koyulmuştur [25].

Günümüz rekabet ortamında bakkalların rakiplere göre daha ucuz, kaliteli ürün satabilmesinin ve tedarik süresinin hızlı olmasının müşterisini artırdığına vurgu yapılan bir çalışmada Japonya'daki 54000 adet bakkal üzerinde inceleme yapılmıştır. Çok fazla sayıda market olduğu ve bu büyümenin de devamlı artmasından dolayı ürün tedarik ve pazarlama sürelerinin de uzayabileceğine değinilmiştir. Bu çalışmada birden fazla markete mal ve hizmet teslimi yapan toptancılar için en uygun araç rotasını bulmak üzere web kazıma teknolojileri ve excel visual basic kullanan bir yaklaşım belirlenmiştir. Problemi incelemek için ilk olarak network flow model (ağ akış modeli) çizilmiş ve web kazıma teknikleri ile tüm bakkallar, bunlara bağlı depolar, benzin istasyonlarının coğrafi verileri çekilmiştir. Elde edilen bu coğrafi veriler ile Google API hizmetini entegre ederek en optimum ve en hızlı sonucu bulma yolunda yöntem geliştirmişlerdir [26].

Yapılan literatür taraması sonucunda elde edilen bu araştırmalara bakıldığında dünya genelinde popüler olarak kullanılan web tarama ve web kazıma tekniklerinden bir konu hakkında web ortamında ilgili sayfalardan istenilen verilerin çıkarılması ve bu verilerin analizinin yapılması amacıyla kullanıldığı farkedilmiştir. Yapılan çalışma sonucunda geliştirilen uygulama; kullanıcının istediği verilerin web ortamından dinamik bir şekilde, girdiği arama kelimesini kullanarak web tarama ve kazıma işlemleriyle gerçekleştirmektedir. Bu sayede web tarama ve kazıma yöntemleri alanında yapılan çalışmamızda istenilen verinin tek seferde çıkarıp analizini yapmaktan ziyade sürekli olarak kullanıcının isteğine göre çalışan bir uygulama ortaya koyulmuştur.

3. MATERYAL METOD

Günümüz modern dünyasında internet üzerinden araştırma yapmak oldukça kolaydır. Fakat araştırma yaparken aramayı belirli başlı kriterlere göre daraltabilmek, üstelik bunu yaparken kolaylıkla kullanabilecek bir arayüz ile yapabilmek oldukça önemlidir. Geliştirilen uygulama, ilerleyen zamanlarda farklı özelliklerin eklenebilmesini veya çıkarılabilmesini sağlamak amacıyla modüler bir yapıda geliştirildi. Uygulama geliştirilirken, kullanıcıların farklı kriterlere göre kolay bir şekilde arama yapabilmesini sağlamak amacıyla kullanıcı dostu bir arayüz oluşturuldu. Yapılan aramalara hızlı bir şekilde cevap alabilmesi için bir servis ve bu servisin içerisinde asıl işi yapacak olan tarama ve veri toplama botları geliştirildi. Projenin geliştirme adımları Şekil 1' de gösterilmiştir. Çalışma dört aşamalı olarak geliştirildi. Web crawler, web scraper, backend ve frontend geliştirmeleri bu dört aşamayı oluşturmaktadır.



Şekil 1. Uygulama geliştirme adımları

3.1. Tarama (Web Crawling)

Web ortamında hazırlanmış bir sorgu(query) ile geliştirme araçları kullanarak arama yapmaya ve ilgili sayfaları bulma işlemi tarama olarak adlandırılmaktadır [27]. Tarayıcı ile bir arama motorunda ya da arama hizmeti veren servislerle yapılan aramalar web tarama işlemine örnek olarak gösterilebilir. Google, Yandex, Bing gibi arama motorlarının sağladığı Application Programming Interface'ler (API) belli başlı limitasyonlarla web tarama hizmetini sunmaktadırlar. İlgili API'ler istek tabanlı cevap

döndürmektedirler. İstekler, kullanıcının arama yaptığı sorgu ve kriterlerle WWW ortamında aramaları gerçekleştirip cevap olarak web sayfa URL'lerini döndürmektedir. Yapılacak arama sayısı ya da arama sonucu dönen sayfa sayısı bazında limitasyonları vardır. Bu sayılara ulaşıldığında servis ücretsiz hizmetini sonlandırmaktadır. Limitlendirme yetmediği takdirde ekstra ücretlendirme ile bu limitasyonlar ortadan kaldırılabilmektedir.

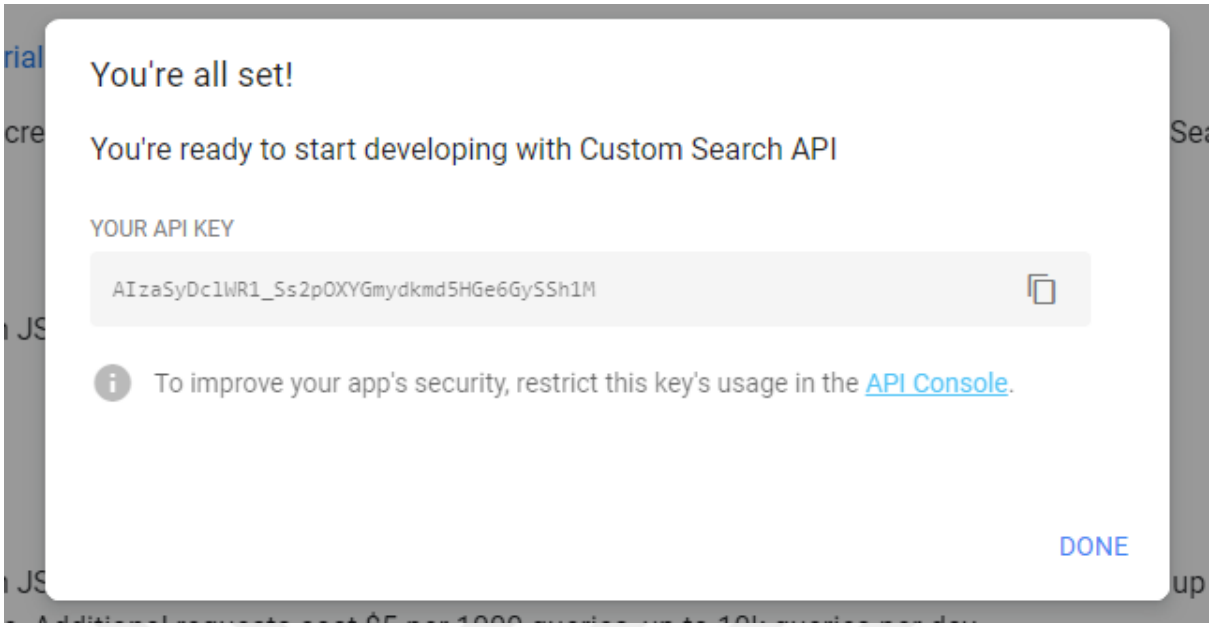
Girilen bir anahtar kelime ile ilişkili makale verisini elde etmenin ilk adımı olarak WWW ortamında arama yapıp istenen kriterlere göre sonuçlar getirmesi için crawling işlemlerini yapacak tarama botu geliştirilmiştir. Bu bot geliştirilirken günümüzün en yaygın kullanılan arama motoru olan Google'ın belirli limitler ile ücretsiz olarak sunduğu Google Search JSON API hizmetinden faydalanılmıştır.

3.1.1. Google Search JSON API

Google'ın kullanıcılara sunduğu Search JSON API, kullanıcıya tahsis edilmiş ve özelleştirilebilir bir arama motoru yardımıyla WWW üzerinde istek tabanlı arama yapmaya yarayan bir servistir.

3.1.1.1. API anahtarı oluşturulması

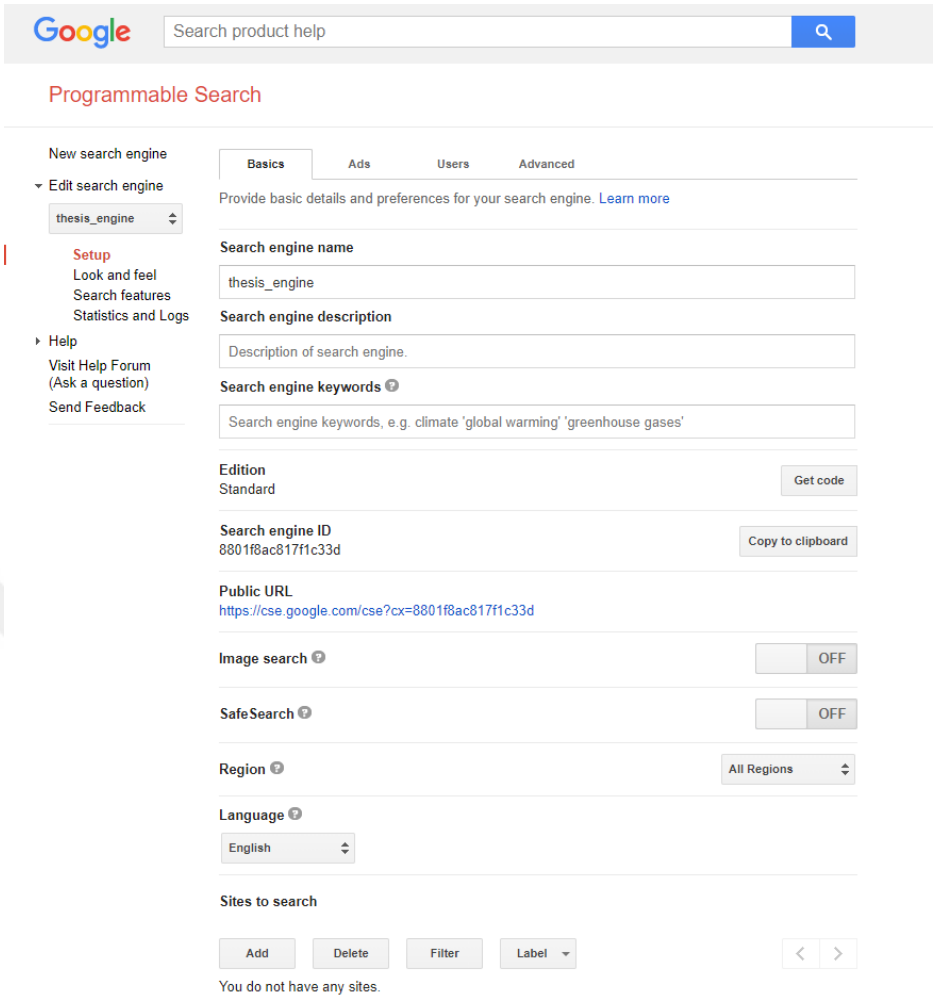
Search JSON API servisine üye olduktan sonra kullanıcıya özel bir API anahtarı üretilmelidir. https://console.cloud.google.com/ adresinde credentials menüsünden Get a Key butonuna basılarak ilgili google account' u için API' lerde kullanılmak üzere bir API anahtarı üretilir. API anahtarı üretildikten sonra Şekil 2' de gösterildiği gibi bir pop-up (açılır pencere) çıkmaktadır. Bu anahtara istenildiği zaman Google API yönetim ekranlarından ulaşılabilmekte ve istenirse anahtar yenilenebilmektedir. Atılan her istekte bu API anahtarı kullanıcı üyeliğinin doğrulaması için gereklidir.



Şekil 2. Google Search JSON API için API anahtarının oluşturulması.

3.1.1.2. Arama motoru oluşturulması

Search JSON API servisine üye olup API anahtarı elde edildikten sonra bir Programlanabilir Arama Motoru (Programmable Search Engine oluşturulması gerekmektedir. https://programmablesearchengine.google.com/ adresine giderek Google hesabı altından yeni bir arama motoru oluşturulur. Şekil 3'te oluşturulmuş bir arama motoru gösterilmiştir.

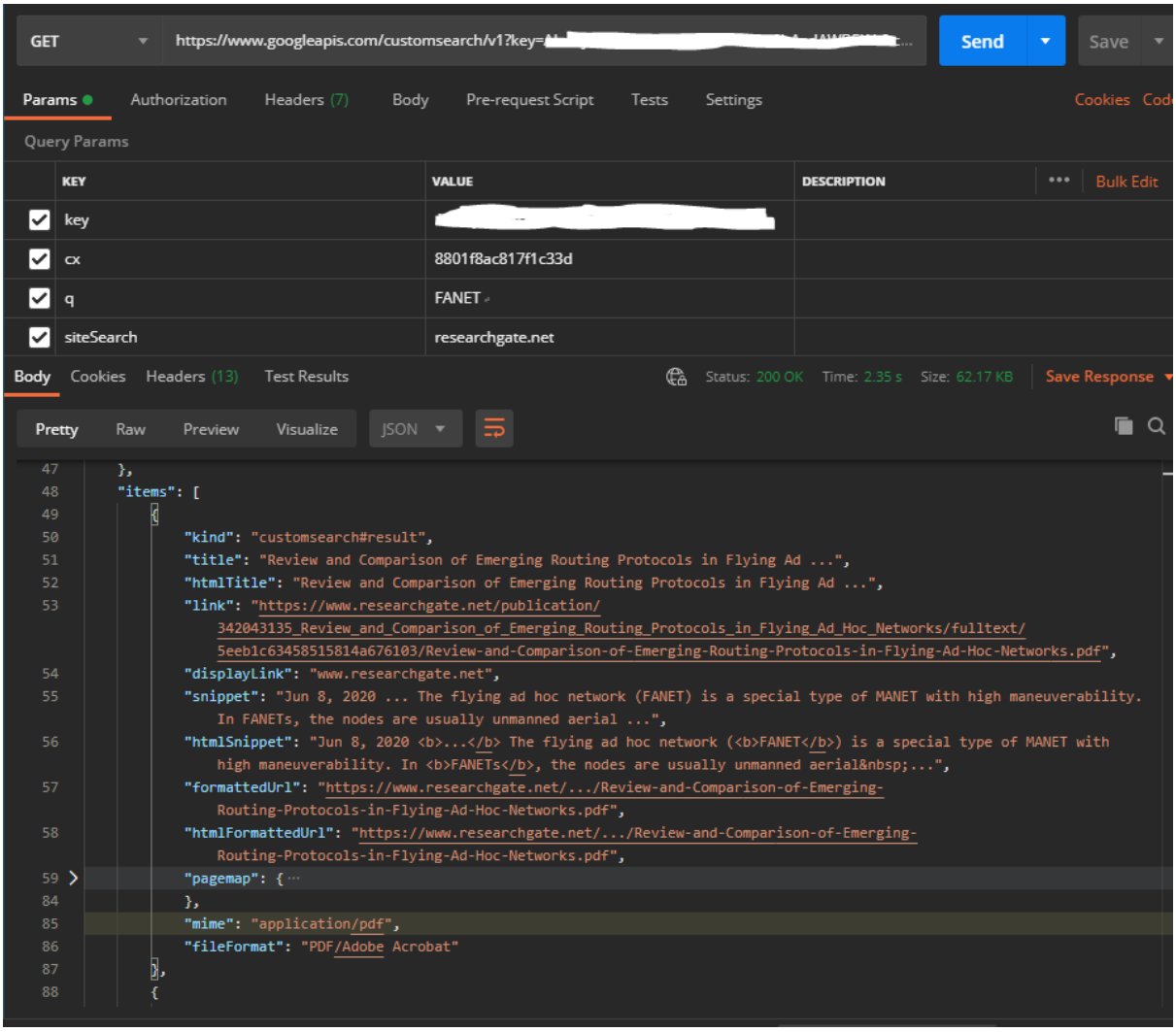


Şekil 3. Google ile kişiselleştirilmiş arama motoru oluşturulması

Hesaba özel oluşturulan arama motorunda çeşitli konfigürasyonlar yapılabilmektedir. Aramanın sadece belirlenen dillerde, belirlenen bölgelerde, belirlenen sitelerde yapılması ya da bu belirlenenler hariç tutularak yapılması gibi ayarlar yapılabilmektedir. Oluşturulan bu kişisel arama motorunun kullanıcıya özgü API anahtarı ile eşleşen bir ID değeri bulunmaktadır. Yine aynı şekilde servise atılan her istekte aramanın hangi arama motoru üzerinden yapılacağını belirtmek için bu ID bulunmalıdır. Google'ın sunduğu bu arama motoruna aranacak kelime ile atılan her istek için bir sayfada 10 adet URL bulunmakta ve maksimum olarak da 10 sayfa sonuç getirmektedir. Bir başka ifade ile, 10 sayfa ve 10 adet URL olmak üzere aranan bir kelime için toplam 100 adet URL dönmektedir. Ek olarak da API anahtarı özelinde günlük 100 adet istek atılabilmektedir. Sınırlı olarak sunulan istek sayısı kullanıcı için yeterli olmadığı durumda 1000 adet istek için $5 ücret talep edilmektedir [28]. Bu ödenen ücret ise 10000 istek ile sınırlıdır. İhtiyaç halinde 10000 istek üzeri için Google'ın ücretli olarak sunduğu Custom Search Site

Restricted JSON API kullanılabilmektedir. [29]

https://www.googleapis.com/customsearch/v1 endpointine "key" parametresi ile API anahtarı, "cx" parametresi ile kişiselleştirilmiş arama motorunun id'si, "q" parametresi ile arama sorgu cümlesi ve "siteSearch" parametresi ile GET isteği atılarak arama başlatılır. Arama tamamlandığında Google döndüğü json cevabı içerisinde birçok metadata ile birlikte bulduğu web sayfalarını "items" dizisi altında iletir. Bu dizi içerisinde her bir site için url, title, fileFormat gibi bilgiler dönmektedir. Şekil 4'de Google Search JSON API ile yapılmış örnek bir arama isteğine yer verilmiştir.



Şekil 4. Google Search JSON API örnek request

3.1.2. Tarama (Crawler) botları

Kişiselleştirilmiş arama motoru oluşturulduktan ve konfigürasyonu yapıldıktan sonra Python dilinde geliştirdiğimiz serviste görev yapmak üzere crawler botu geliştirilmiştir. Crawler botu aratılan anahtar kelimeye, filtrelere ve arama yapılması istenen web sayfalarına göre arama motoruna istek göndermekte ve sonuçları döndürmektedir. Dönen sonuçlar içerisinden "items" JSON nesnesi içerisinden tüm nesneler ayıklanıp elde edilmiştir. Bu nesneler içerisinden site url'ini elde etmek için "link", dosya formatını elde etmek için "fileFormat" nesnesi parse edilerek bilgilere ulaşılmıştır.

Anahtar kelime bazlı literatür taraması yapabilmek için aramalar mühendislik alanlarındaki akademik çalışmalarda yaygın olarak kullanılan Researchgate, IEEE, Springer, ACM siteleri ile sınırlandırılmıştır. Sitelerin robots.txt dosyaları detaylıca incelenmiş ve herhangi bir illegal durum olmadığı teyit edilmiştir. Bu sitelere ek olarak taranmak istenen Scopus, ScienceDirect, Academia gibi siteler robot engeline veya ücretli erişime ihtiyaç duyduğu için dahil edilmemiştir.

İstenilen anahtar kelime için yapılan aramayla birlikte crawler botu çalışacak ve bunun sonucunda Google Search JSON API ile belirlenmiş makale sitelerindeki ilişkili sonuçlar listelenecektir.

3.2. Veri Kazıma (Web Scraping)

Tarama (crawling) işleminden sonra ikinci adım olarak elde edilen url listesindeki makale özetlerini çıkarabilmek için web kazıma botu geliştirilmiştir. Bu bot, web kazıma teknolojileri konusunda dil havuzunda oldukça fazla kütüphaneye sahip olduğu ve geliştirilecek servise entegre edileceği için Python dili ile ilerlenmiştir.

3.2.1. Yasallık kontrolü

Sitelerin ana domaini üzerinde "domain/robots.txt" yoluyla sitesinde kazıma ve tarama botlarının nerelerde ve nasıl gezebileceğine dair bir dosya bulunmaktadır. Yasal bir tarama ve kazıma çalışması için bu dosyaya dikkat ederek sitede işlem yapılması gerekmektedir. Aksi takdirde site sahibi her türlü hukuki yaptırım hakkına sahiptir.

Researchgate.net sitesinden alınmış robots.txt dosyası örneği Şekil 5'de verilmiştir. Buna göre User-agent karşısında belirlenen tarayıcı motoru için alt satırlarında kuralları

verilmektedir. Herhangi bir kazıyıcı ya da tarayıcı bot ile “Allow” komutunun karşısına yazılan web sayfasının yolları ziyaret edilebilirken “Disallow” karşısına yazılan yollar ziyaret edilmemelidir. Şekil 5’e baktığımızda * olarak belirtilen “herhangi bir tarayıcı motoru” ile yapılabilecekler ve yapılamayacaklar listelenmiştir. Buna göre tarayıcı motorunun altında “Disallow” karşısında verilen “/connector/”, “/plugins” gibi yolları bot ile ziyaret etmek yasaklanmışken “Allow” komutu karşısında “/” ile belirtilmiş diğer tüm yollar için ziyarete açıktır.

```
User-agent: *
Allow: /
Disallow: /connector/
Disallow: /plugins.
Disallow: /firststeps.
Disallow: /publicliterature.PublicLiterature.search.html
Disallow: /lite.publication.PublicationRequestFulltextPromo.requestFulltext.html
Disallow: /amp/authorize
Allow: /signup.SignUp.html
Disallow: /signup.
```

Şekil 5. Robots.txt dosyası içeriği

Yukarıda bahsedildiği gibi kazımanın yasallık çerçevesinde olması için “robots.txt” dosyası gözetilmiştir.

3.2.1.1. Yardımcı kütüphaneler

Web kazıma işlemlerinde yasallık kontrolü yapılabilmesi için “reppy” kütüphanesinden faydalanılmıştır.

Reppy kütüphanesi

Elde edilen sayfaları gezmeden önce “reppy” kütüphanesi ile ilgili sayfadaki makalenin web scraping yöntemleri ile kazınmasında bir problem olup olmadığını kontrol edecek algoritma geliştirilmiştir. Şekil 6’ da bu işlemi yapan örnek bir kod parçacığına yer verilmiştir.

```
from reppy.robots import Robots

# This utility uses `requests` to fetch the content
robots = Robots.fetch('http://example.com/robots.txt')
robots.allowed('http://example.com/some/path/', 'my-user-agent')
```

Şekil 6. reppy ile kullanılan örnek algoritma

Reppy içerisindeki robots sınıfı ve metotları sayesinde "robots.txt" dosya kontrolü kolaylıkla yapılabilmektedir. İlk olarak robots class'ına ait "fetch" methodu ile robots.txt dosyasının olduğu website yolunu parametre geçerek tanıtıyoruz. Daha sonra bool değer döndüren "allowed" metoduna ziyaret etmek istediğimiz sayfayı ve ziyaret gerçekleştirirken kullandığımız tarayıcı motorunu parametre olarak gönderiyoruz. Buradan alınan "true" değeri ilgili sayfanın kazımak için legal olduğunu, alınan "false" değeri ise illegal olduğunu belirtmektedir.

Reppy kütüphanesi kullanılarak robots.txt algoritmasına göre bir engel varsa o sayfa ziyaret edilmekten kaçınılmıştır. Bir engel yoksa "başarılı" mesajı ile birlikte botlarımız işlemlere hazır hale getirilmiştir. Kazıma algoritmalarına başlamadan önce her zaman bu algoritma kullanılmaktadır. [30]

3.2.2. Kazıyıcı (Scraper) botları

Robots algoritmasından başarılı sonuç döndüyse bu durum, sayfa indirilerek kazıma işlemlerine başlanabileceğini göstermektedir. Sayfayı indirebilmek ve kazıma yapabilmek için uygulamanın çalışacağı makale sitelerinin her biri için özgün algoritmalar geliştirilmiştir. Sitelerin geliştirme metodolojilerinin farklı olması, sayfa ziyaretlerindeki kullanıcıya verdikleri izinlerin farklı olması bu konuda etken sebep olmuştur.

Bu algoritmalar içerisinde sayfayı indirebilmek, oturum (session) kontrolü yapabilmek için "Requests" kütüphanesi kullanılmıştır.

3.2.2.1. Yardımcı kütüphaneler

Web scraper botları geliştirilirken çeşitli yardımcı kütüphanelerden faydalanılmıştır. Bu yardımcı kütüphaneler "Requests" ve "BeautifulSoup" kütüphaneleridir.

Requests kütüphanesi

HTTP requestleri göndermeye yarayan kütüphane, sayfayı ziyaret ederken gerektiği yerde çerezleri yöneterek, yetkilendirme (authentication) sağlayarak sayfada işlemler yapabilme imkânı vermektedir [31].

Şekil 7' de requests kütüphanesinin kullanımı ile ilgili örneklere yer verilmiştir.

```
>>> import requests
>>> r = requests.get('https://api.github.com/user', auth=('user', 'pass'))
>>> r.status_code
200
>>> r.headers['content-type']
'application/json; charset=utf8'
>>> r.encoding
'utf-8'
>>> r.text
'{"type":"User"...'
>>> r.json()
{'disk_usage': 368627, 'private_gists': 484, ...}
```

Şekil 7. Requests ile örnek kullanım

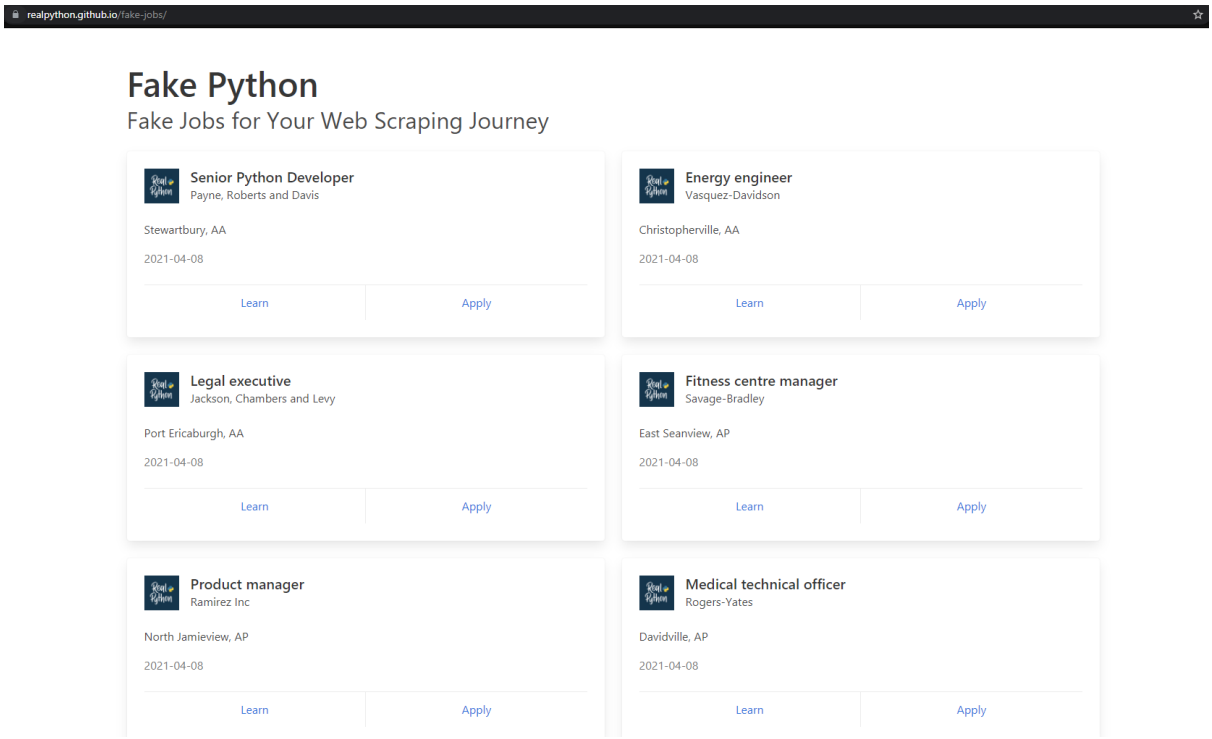
Requests sınıfı içerisinde sayfaya giriş yapmak için GET isteğinin gönderildiği örnekte isteğin tamamlanmasından sonra ihtiyaç halinde istek için cevapta HTTP response statüsünü veren "status_code", header'ları veren "headers", sayfanın içeriğini text olarak veren "text" gibi metotları bulunmaktadır. İhtiyaç halinde bu cevap mettoları kullanılabilir.

Bir sayfaya HTTP isteği metotlarıyla (POST, GET, DELETE, PUT vb.) istek atmaya, istek gönderirken headers, body, authentication ekleyerek gönderebilmeye izin veren kütüphane, uygulamamız içerisinde ziyaret edilecek akademik web sayfalarında ilk adımda kullanılmıştır. Genel olarak sayfayı algoritma çalışma esnasında sayfayı indirme amacıyla kullanılmıştır. Akademik web sayfası indirme işlemi için bir yetkilendirme istiyorsa, ilk olarak bu kütüphane ile sayfaya giriş yapılır ve indirme işlemine başlanır. Detaylarına ziyaret edilen sayfa için geliştirilen algoritma özelinde, aşağıdaki kısımlarda yer verilmiştir.

BeautifulSoup kütüphanesi

Botun kazıma işlemini yapabilmesi için yaygın olarak kullanılan "BeautifulSoup" kütüphanesinden faydalanılmıştır. Bu kütüphane, Requests kütüphanesi ile indirilmiş sayfayı kaynak olarak alıp HTML ağacına dönüştürmektedir. Bu HTML ağacında element tipine göre bulma, sınıfa göre bulma, xpath' e göre bulma gibi yöntemler kullanarak sayfanın ilgili elementi elde edilebilmektedir [32].

Şekil 8' te web kazıma konusunda pratik yapma amaçlı geliştirilmiş "Fake Jobs" sitesinin ekran görüntüsü verilmiştir.



Şekil 8. Web scraping testleri için geliştirilmiş Fake Jobs web sayfası

Bu sitenin BeatifulSoup kullanılarak ilgili mesleklerin ve detaylarının çıkarılması aşağıdaki örneklerde adım adım verilmiştir. İlk olarak sayfa "requests" kütüphanesi ile imdirilmiş ve HTML ağacı elde edilmiştir. Burada işlerin her biri için HTML içerisindeki parent div elementi Şekil 9' da gösterildiği gibi elde edilmiştir.

```HTML
<div class="card">
  <div class="card-content">
    <div class="media">
      <div class="media-left">
        <figure class="image is-48x48">
          <img
            src="https://files.realpython.com/media/real-python-logo-thumbnail.7f0db7
            alt="Real Python Logo"
          />
        </figure>
      </div>
      <div class="media-content">
        <h2 class="title is-5">Senior Python Developer</h2>
        <h3 class="subtitle is-6 company">Payne, Roberts and Davis</h3>
      </div>
    </div>

    <div class="content">
      <p class="location">Stewartbury, AA</p>
      <p class="is-small has-text-grey">
        <time datetime="2021-04-08">2021-04-08</time>
      </p>
    </div>
    <footer class="card-footer">
      <a
        href="https://www.realpython.com"
        target="_blank"
        class="card-footer-item"
        >Learn</a
      >
      <a
        href="https://realpython.github.io/fake-jobs/jobs/senior-python-developer-0.ht
        target="_blank"
        class="card-footer-item"
        >Apply</a
      >
    </footer>
  </div>
</div>
```

Şekil 9. Fake Jobs sayfası içerisindeki bir işin HTML ağacı

Bu HTML ağacında görüldüğü üzere işin başlığı ve şirketin ismi "media-content" sınıfına sahip div'e ait "title is-5" ve "subtitle is-6 company" sınıflarına sahip elementlerin içerisinde metin olarak verilmiştir. İşin lokasyonu ve tarihi de "content" sınıfına sahip

div'e ait "location" ve "is-small has-text-grey" sınıflarına sahip elementlerin içerisinde metin olarak verilmiştir. Bu işlerin herbiri de "ResultsContainer" id'sine sahip bir container div içerisindedir. Kazıma işlemi bu div elementi içerisinde yapılacaktır. Şekil 10' da "requests" ile indirilmiş sayfanın içeriği "BeatifulSoup" objesine aktarılmıştır.

```
import requests
from bs4 import BeautifulSoup

URL = "https://realpython.github.io/fake-jobs/"
page = requests.get(URL)

soup = BeautifulSoup(page.content, "html.parser")
```

Şekil 10. Sayfanın indirilmesi ve BeatifulSoup objesine dönüştürülmesi

Artık, "soup" isimli nesnede sayfadaki tüm HTML ağacı bulunmaktadır. İlk olarak yapılan işlem ile "find" metodu kullanılarak sayfada bulunan mesleklerin parent elementi olan " " isimli element ve içerisindeki metin seçilmiştir. Şekil 11' de bu işlem için kod parçacığına yer verilmiştir.

```
results = soup.find(id="ResultsContainer")
```

Şekil 11. ResultsContainer id'li tüm meslekleri içinde barındıran container elementinin seçilmesini sağlayan kod parçacığı

Şekil 12' de verilen kod parçacığında görüldüğü üzere elde edilen bu results isimli soup nesnesi içerisinde "find_all" isimli metot ile her biri Şekil 9' da verilen "card-content" isimli div' ler elde edilmiştir.

```
job_elements = results.find_all("div", class_="card-content")
```

Şekil 12. Tüm işlerin parent elementi olan card-content elementinin ve içeriğinin seçilmesi

Elde edilen "job_elements" nesnesi artık tüm işleri HTML olarak içerisinde barındırmaktadır. For döngüsü ile içerisinde gezilip her bir iş için title, company ve location bilgileri çıkarılmaya hazırdır. Yine "find" metodu ile Şekil 10' da yer alan kod bloğu ile mesleklerin bilgileri kazınmıştır. Şekil 13' de bu işlemleri gerçekleştiren kod bloğu verilmiştir.

```
for job_element in job_elements:
    title_element = job_element.find("h2", class_="title")
    company_element = job_element.find("h3", class_="company")
    location_element = job_element.find("p", class_="location")
    print(title_element)
    print(company_element)
    print(location_element)
```

Şekil 13. İş soup objeleri içerisinden title, company ve location bilgilerini içeren elementlerin kazınması

Elde edilen bilgilerin yazdırılmış hali Şekil 14' de yer alan kod bloğunda verilmiştir.

```
<h2 class="title is-5">Senior Python Developer</h2>
<h3 class="subtitle is-6 company">Payne, Roberts and Davis</h3>
<p class="location">Stewartbury, AA</p>
```

Şekil 14. Elde edilen title, company ve location elementlerinin çıktısı

Doğru sonuca ulaşıldığı tespit edildikten sonra bu elementlerin içerisindeki metinler "text" metodu ile Şekil 15' deki gibi çıkarılmıştır.

```
for job_element in job_elements:
    title_element = job_element.find("h2", class_="title")
    company_element = job_element.find("h3", class_="company")
    location_element = job_element.find("p", class_="location")
    print(title_element.text)
    print(company_element.text)
    print(location_element.text)
    print()
```

Şekil 15. Elde edilen title, company ve location elementlerinin textlerinin kazınması

Metinler kazındıktan sonra da "Senior Python Developer", "Energy Engineer", "Legal Executive" mesleklerinin yukarıdaki koda göre çıktısı Şekil 16' da verilmiştir.

```
Senior Python Developer
Payne, Roberts and Davis
Stewartbury, AA

Energy engineer
Vasquez-Davidson
Christopherville, AA

Legal executive
Jackson, Chambers and Levy
Port Ericaburgh, AA
```

Şekil 16. Hedeflenen çıktının elde edilmesi; her bir iş için title, company ve location bilgileri

BeatifulSoup kütüphanesi oldukça başarılı ve yaygın olarak kullanılmasının yanında her bir web sitesinin içeriği ve HTML ağacı farklı olduğu için farklı algoritmalar kullanılmasını gerektirmektedir. Bunun yanısıra bazı sayfalarda da yardımcı kütüphaneler gerektirmektedir. Teknik olarak geliştirme metodolojisi farklı olan, HTML ağacının çıkarılamadığı sayfalar için de sayfa metadatasına ulaşarak makale özetini elde etme yoluna gidilmiştir. Bu amaçla da indirilmiş sayfa metadatasını elde etmeye yarayan "extraction" kütüphanesi kullanılmıştır [33].

3.2.2.2. Springer veritabanı için geliştirilen algoritma

Şekil 17' de Springer web sayfasından istenilen bilgilerin elde edilmesini sağlayan algoritma görülebilmektedir [34].

```python
import requests
from bs4 import BeautifulSoup

class SpringerBot:

    def __init__(self):
        # constants
        self.session = requests.Session()

    def siteToSoup(self,siteUrl, headers, cookies):
        try:
            page = self.session.get(siteUrl,headers=headers,cookies=cookies)
            soup = BeautifulSoup(page.content, 'html.parser')
            return soup
        except Exception as ex:
            print(ex)

    def scrapeSite(self, siteUrl, headers, cookies):
        try:
            soup = self.siteToSoup(siteUrl, headers, cookies)

            if(soup != None):
                abstractElement = soup.find('div', class_="c-article-section__content")
                if abstractElement != None:
                    children = abstractElement.findChildren("p" , recursive=False)

                    abstract = ""
                    for child in children:
                        abstract += child.get_text()+" "

                    return abstract
                else:
                    return ""
            else:
                return ""
        except Exception as ex:
            print(ex)
```

Şekil 17. link.springer.com için geliştirilmiş kazıma botu

Springer sitesinde kazıma işlemlerinin yapılabilmesi için geliştirilmiş algoritmada "Requests" kütüphanesi içerisindeki "Session" metodu kullanılarak bir oturum oluşturulmuştur. Oluşturulan bu oturum ile sayfanın çerezlerini elde edip sayfayı indirdikten sonra BeautifulSoup nesnesi oluşturularak sayfanın HTML ağacı çıkarılmıştır. Çıkarılan bu ağaca göre de daha önceden tespit edilen makale özet kısmının bulunduğu HTML elementi olan "c-article-section__content" sınıf isimli parent div elde edilip içerisindeki tüm "p" elementlerinin textleri kazınıp birleştirilerek makale özetine ulaşılmaktadır.

3.2.2.3. ACM veritabanı için geliştirilen algoritma

Şekil 18' de ACM üzerinden özet(abstract) elde edilmesini sağlayan kod bloğu görülebilmektedir [35].

```
import requests
from bs4 import BeautifulSoup

class ACMBot:

    def __init__(self):
        # constants
        self.session = requests.Session()

    def siteToSoup(self,siteUrl, headers, cookies):
        try:
            page = self.session.get(siteUrl,headers=headers,cookies=cookies)
            soup = BeautifulSoup(page.content, 'html.parser')
            return soup
        except Exception as ex:
            print(ex)

    def scrapeSite(self, siteUrl, headers, cookies):
        try:
            soup = self.siteToSoup(siteUrl, headers, cookies)

            if(soup != None):
                abstractElement = soup.find('div',class_="abstractSection").find("p",recursive=True)

                if abstractElement != None:
                    abstract = abstractElement.get_text()
                    return abstract
                else:
                    return ""
            else:
                return ""
        except Exception as ex:
            print(ex)
```

Şekil 18. dl.acm.org için geliştirilmiş kazıma botu

ACM sitesinde kazıma işlemlerinin yapılabilmesi için geliştirilmiş algoritmada "Requests" kütüphanesi içerisindeki "Session" metodu kullanılarak bir oturum oluşturulmuştur. Oluşturulan bu oturum ile sayfanın çerezlerini elde edip sayfayı indirdikten sonra BeautifulSoup nesnesi oluşturularak sayfanın HTML ağacı çıkarılmıştır. Çıkarılan bu ağaca göre de daha önceden tespit edilen makale özet kısmının bulunduğu HTML elementi olan "abstractSection" class isimli parent div elde edilip içerisindeki tüm "p" elementlerinin textleri ve bu elementlerin de içerisindeki "p" elementlerinin "text" nesneleri kendini çağıran fonksiyonlarla kazınıp birleştirilerek makale özetine ulaşılmaktadır.

3.2.2.4. IEEE veritabanı için geliştirilen algoritma

Şekil 19' da IEEE ile yapılan kazıma işleminin algoritmasına yer verilmiştir [36].

```
import requests, extraction
from bs4 import BeautifulSoup

class IEEEBot:

    def __init__(self):
        # constants
        self.session = requests.Session()

    def siteToSoup(self,siteUrl, headers, cookies):
        try:
            page = self.session.get(siteUrl,headers=headers,cookies=cookies)
            soup = BeautifulSoup(page.content, 'html.parser')
            return soup
        except Exception as ex:
            print(ex)

    def scrapSite(self, siteUrl, headers, cookies):
        try:
            soup = self.siteToSoup(siteUrl, headers, cookies)

            if(soup != None):
                ext = extraction.Extractor()

                metadata = ext.extract(soup.prettify(), source_url=siteUrl)

                if metadata != None:
                    abstract = metadata.description
                    return abstract
                else:
                    return ""
            else:
                return ""
        except Exception as ex:
            print(ex)
```

Şekil 19. ieee.iexplore.org için geliştirilmiş kazıma botu

IEEE akademik araştırma web sitesi farklı bir metodolojiyle geliştirildiği için "Requests" kütüphanesi ile indirildiğinde sayfa HTML ağacının edilemediği görülmüştür. Dünyada popüler olarak kullanılan bu sitenin geliştirilen literatür tarama uygulamasında

kullanıldığında oldukça faydalı olacağı düşünüldüğü için makale özetine ulaşmanın farklı yolları araştırılmıştır. Siteye ait makale sayfalarını incelerken makale özetinin sayfa metadatasının içerisinde de bulunduğu saptanmıştır. Ve kazıma işlemi için metadata hedef alınmıştır. Sayfa metadatasının çıkarılması için "extraction" kütüphanesinden faydalanıp description içerisindeki makale özetine erişim sağlanmıştır.

3.2.2.5. Researchgate veritabanı için geliştirilen algoritma

Şekil 20' de akademik araştırmalarda oldukça yaygın kullanılan Researchgate.net için veri kazıma işlemlerinde kullanılan kod bloğuna yer verilmiştir.



```python
import requests
from bs4 import BeautifulSoup

from .Config import settings

class ResearchgateBot:

    def __init__(self):
        # constants
        self.username = settings.LOGIN
        self.password = settings.PASSWORD
        self.headers = settings.HEADERS
        self.token = settings.REQUEST_TOKEN
        self.loginEndpoint = settings.ENDPOINT
        self.authenticated = False
        self.session = requests.Session()

    def login(self):
        if(self.authenticated == False):
            data = {
                "request_token": self.token,
                "invalidPasswordCount": "0",
                "login": self.username,
                "password": self.password,
                "setLoginCookie": "on"
            }

            response = self.session.post(self.loginEndpoint, headers=self.headers, data=data)

            if(response.status_code == 200):
                self.authenticated = True

    def siteToSoup(self,siteUrl, headers, cookies):
        try:
            page = self.session.get(siteUrl,headers=headers,cookies=cookies)
            soup = BeautifulSoup(page.content, 'html.parser')
            return soup
        except Exception as ex:
            print(ex)

    def scrapSite(self, siteUrl, headers, cookies):
        try:
            soup = self.siteToSoup(siteUrl, headers, cookies)

            if(soup != None):

                abstractElement = soup.find("div", class_="nova-e-expandable-text__container").find("div")

                if abstractElement != None:
                    abstract = abstractElement.get_text()
                    return abstract
                else:
                    return ""
            else:
                return ""
        except Exception as ex:
            print(ex)
```

Şekil 20. researchgate.net için geliştirilmiş kazıma botu

Springer ve ACM sitelerindeki algoritmanın benzeri Researchgate sitesinde de çalışmaktadır. Fakat sayfa sayısının 100’ den fazla olduğu testlerde web sitesi belirli bir süre sonra “sayfa isteklerini bu kadar sık yapabilmeniz için üye girişi yapınız.” şeklinde uyarı vermekte ve kazıma botunu durdurmaktaydı. Bu sorunun üstesinden gelebilmek için sayfanın tarayıcı üzerinde üye girişi yapılma aşamasındaki network alışverişi incelenmiş ve bu alışveriş “Requests” kütüphanesindeki “Session” sınıfı ile algoritma içerisinde kurgulanmıştır. Buna göre algoritma sitenin de uyardığı şekliyle isteği atmadan önce login olmakta, sonrasında ise BeautifulSoup kütüphanesi ile yine diğer algoritmalarda olduğu gibi HTML ağacından makale özeti elde edilmektedir. Burada da

“nova-e-expandable-text__container” sınıf isimli parent div bulunmuş ve texti kazınarak makale özetine ulaşılmıştır [37].

3.2.2.6. Marmara vetis sistemi için geliştirilen algoritma

Marmara Üniversitesi ve birçok üniversitenin de kendi içerisinde kullandığı Vetis Sistemi (Veritabanı Erişim ve İstatistik Sistemi) sanal bir kütüphanedir. Elektronik ortamdaki veritabanlarına ve kaynaklara istenilen ortamdan üniversite mail adresi ile erişip ücretli/ücretsiz birçok kaynağa erişim olanağı sağlamaktadır.

Ücretsiz erişim sayesinde birçok kaynağa tek bir ortamdan ulaşabilmeyi sağladığı için geliştirilecek olan literatür taraması için iyi bir ortam olduğu düşünülmüştür. Erişim için network alışverişi incelenip kod ile simüle edilmiştir. Fakat araştırmalar sonucunda bu işlemin yasal olmadığı tespit edilmiş ve uygulama içerisinde kullanılmamıştır [38].

3.2.2.7. ScienceDirect için geliştirilen algoritma

Akademisyenler tarafından sıkça kullanılan bir diğer site olan ScienceDirect üzerinde yapılan web scraping çalışması benzer şekilde HTML ağacından uygun elementleri bulunarak kolayca elde edilebilmiştir. Fakat uygulamada kullanılan robot algoritmasıyla yapılan testler sonucunda makalelerin bulunduğu “/science/article” yolunun kazınmasının illegal olduğu tespit edilmiştir. Bundan dolayı uygulamada bu akademik site kullanılmamıştır [39].

3.2.2.8. Scopus için geliştirilen algoritma

Akademik bir web sayfası olmasının yanı sıra çok kapsamlı bir sanal veritabanı olan Scopus, araştırmalar için ücretli üyelik istediği için uygulamaya dâhil edilmemiştir [40].

3.2.2.9. Academia için geliştirilen algoritma

Genel anlamda ücretsiz olan, bazı özellikler ve yükseltmelerin ücretli olduğu Academia, uygulama kapsamında elde edilmek istenen abstract kısmının görüntülenmesini ücretli kıldığı için uygulamaya dâhil edilmemiştir [41].

3.3. Arkayüz (Backend) Geliştirmeleri

Bilgisayar dünyasının gelişmeye başladığı yıllarda yazılım geliştirme yöntemi olarak tüm projenin tek ortamda geliştirildiği tümleşik geliştirme yöntemi kullanılmıştır. Bu da tüm yükün kullanıcının kullandığı ekranlarda olmasına ve dolayısıyla performans düşüklüğüne yol açmıştır. Günümüz modern yazılım dünyasında yazılımın kullanıcı ile erişiminin olmasını gerektirmediği kısımlar arkayüz (backend) adı altında, kullanıcı ile etkileşimi olan kısımlar da önyüz (frontend) adı altında geliştirilmektedir. Tüm yükün kullanıcının kullandığı web uygulamasına, dolayısıyla çalışan bilgisayarın belleğine yüklenmemesi için tüm veritabanı ve fonksiyonel işlemler arkaplanda yapılmaktadır. Bu şekilde arkaplanda yapılan işlerin tamamına, yani kullanıcıların görmediği kısımların tümü arkayüz olarak ifade edilmektedir [42]. Arkayüz, önyüzden gelen talepleri yakalama, veritabanı işlemlerini gerçekleştirme ve önyüze istenilen talep doğrultusunda çıktılar verme gibi işlemlerden sorumludur.

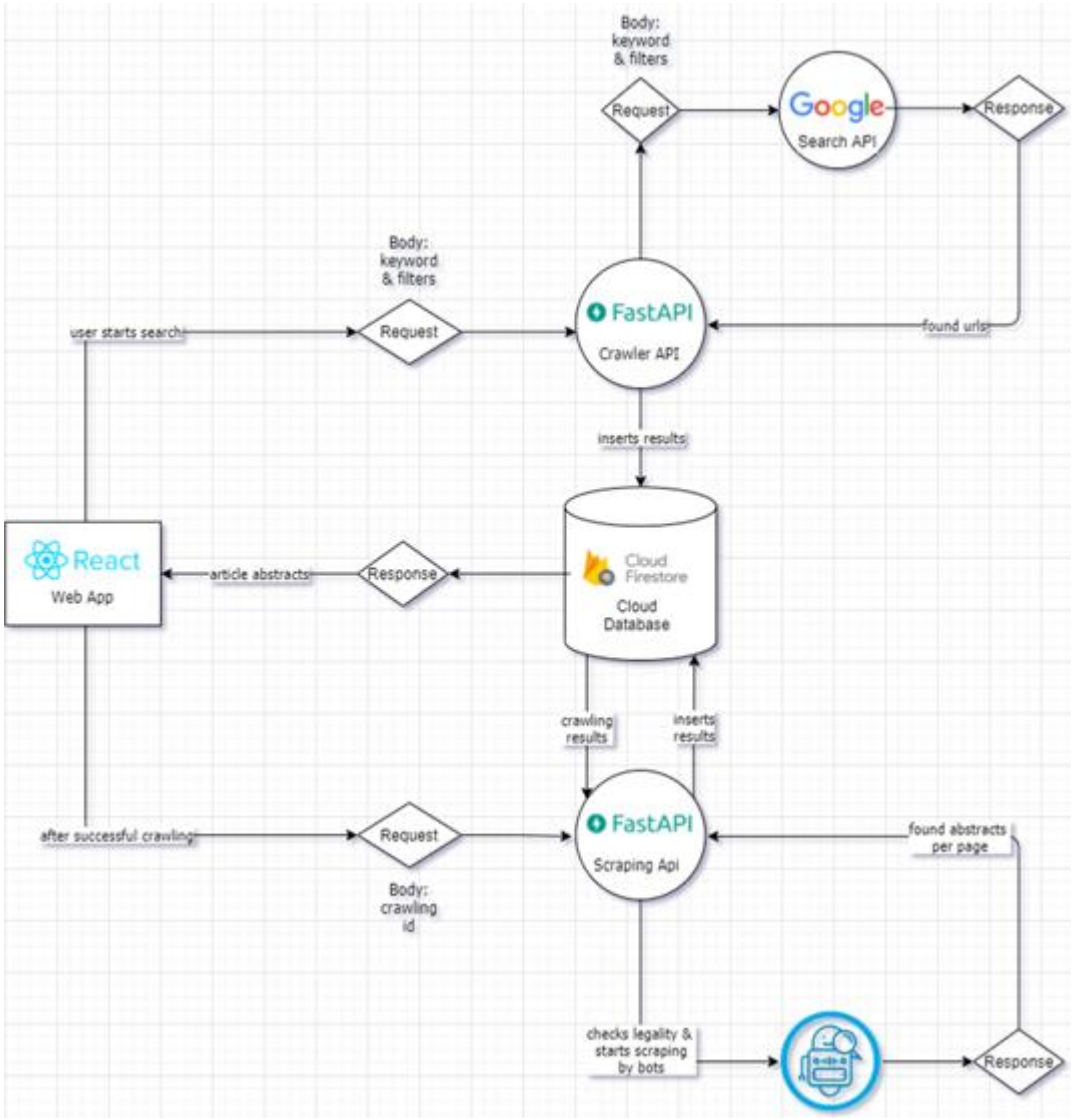
Çalışmanın ana fikri olan araştırılan anahtar kelimeyi makale sitelerinde arayıp, bulunan URL'ler üzerinden makale özetlerini çıkaran algoritmaların (Bölüm 3.1 & 3.2) geliştirmesini tamamladıktan sonra bu algoritmaları yapısal olarak düzenlemek için backend çalışmaları yapılmıştır. Bu kapsamda aratılan anahtar kelimelerin, bulunun URL'lerin ve bu URL üzerinde yapılan çalışmaların, ek olarak tüm uygulama yaşam döngüsündeki verilerin tutulması için bulut (cloud) tabanlı bir veritabanı oluşturulmuştur. Bu veritabanı ile algoritmaları bir araya getirecek ve arayüz üzerinden gelecek tüm isteklere cevap verecek bir servis geliştirilmiştir.

3.3.1. Mimari

Geliştirilen uygulamanın arkayüz mimarisi akışıyla birlikte Şekil 21' de görülebilmektedir. Buna göre ilk olarak uygulama önyüzünden kullanıcı bir arama isteği gönderir. Bu isteği arkayüz servislerinden Crawler API karşılar. Aldığı bu istek detaylarındaki filtrelemelere göre Google Search API'yi kullanarak tarama işlemini başlatır. Google'in dönüş yaptığı site URL'leri veritabanına kaydedilerek tarama işlemi tamamlanmış olur. Başarılı bir şekilde tamamlanmış tarama isteği önyüze iletildikten sonra sistem otomatik olarak kazıma işlemlerini başlatır. Kazıma işlemlerinden sorumlu Scraper API servisi hazır hale gelir. Bu servise tarama işleminin sonucunda veritabanına

kaydedilen “crawling id” ile kazıma başlatma isteği gönderilir. Bu id’ye ait tarama sonucunda kaydedilmiş site URL’leri veritabanından çekilir. İlk olarak robot algoritması kullanılarak yasallık kontrolü yapılır. Kazıma işleminin legal olduğu URL’ler geliştirilen kazıma botları ile kazınmaya başlanır. Bulunan makale özetleri bir “scraping id” ile veritabanına kaydedilerek kazıma işlemleri tamamlanmış olur.

Bu aşamadan sonra da arkayüz mimarisinde tüm işlemler başarıyla tamamlanmış olup önyüze arama anahtar kelimesi ve filtresine göre bulunan makale özetleri iletilir.



Şekil 21. Geliştirilen uygulamanın backend mimarisi

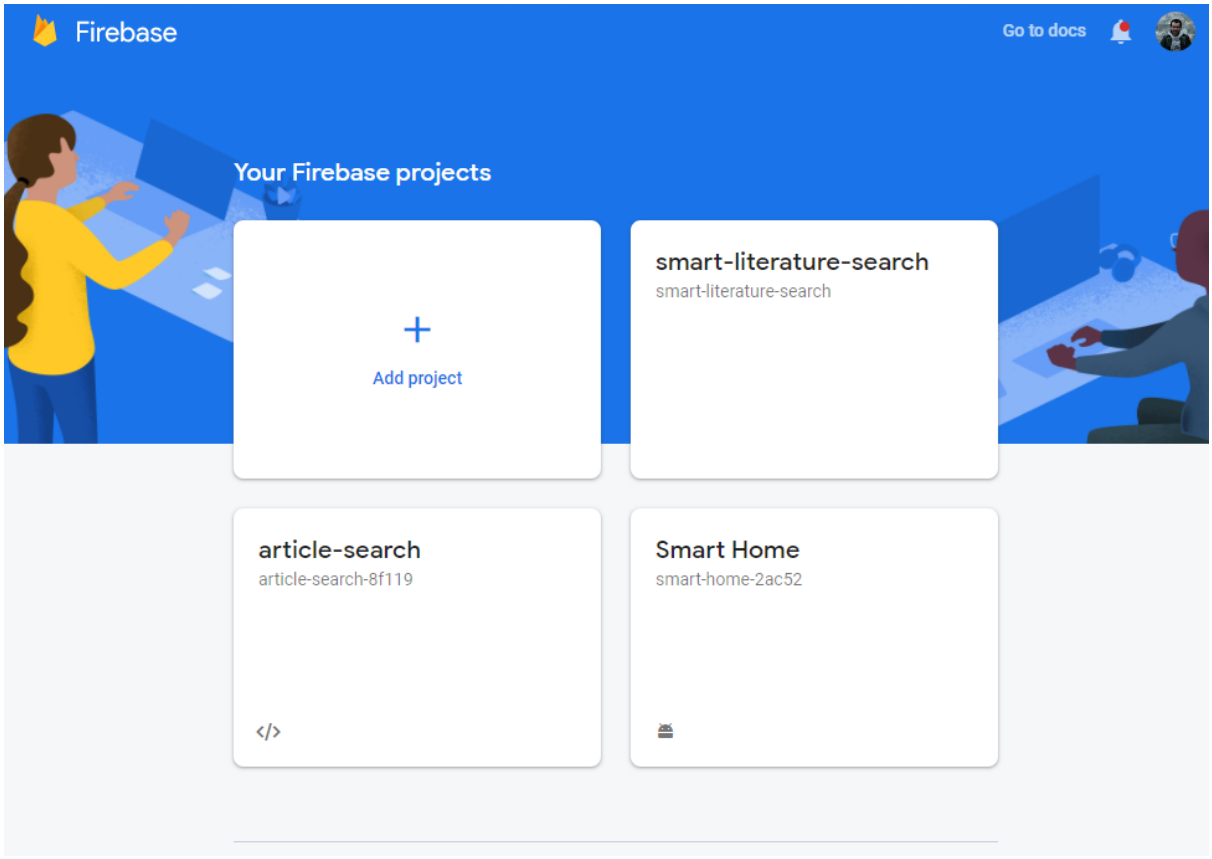
3.3.2. Veritabanı katmanı

Uygulama üzerinde gerçekleştirilen tüm işlemlerin hareket loglarının, yapılan aramanın detaylarının, arama sonucu bulunan URL’lerin, bulunan URL’ler üzerinden çıkarılan

makale hakkındaki bilgilerin tutulması için bir veritabanı oluşturulmuştur. Veri akışının yoğun olduğu uygulamamız için Google'ın Firebase ürünü içerisinde sunduğu yapısal olmayan ve bulut tabanlı çalışan Cloud Firestore veritabanı kullanılmıştır [43].

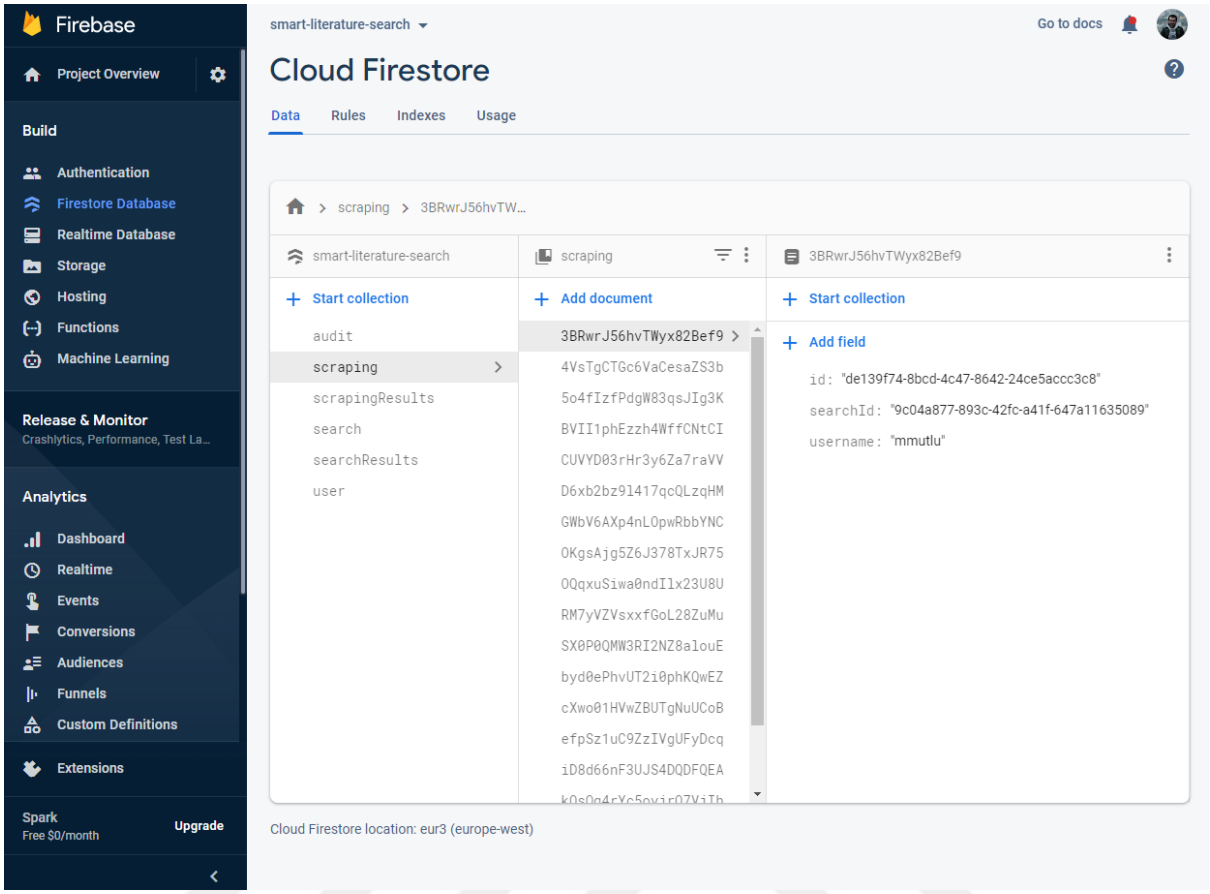
3.3.2.1. Google Firebase Cloud Firestore

Firestore istemci ya da sunucu uygulamaları için NoSQL bir bulut (cloud) veritabanıdır. Bulut ortamında olup tüm dillere desteği, erişiminin ücretsiz ve esnek olması sebebiyle tercih edilmiştir. https://console.firebase.google.com/u/0/ adresine Google hesabı ile üye olduktan sonra yeni bir proje açılır. Şekil 21' de gösterilen yönetim ekranından oluşturulmuş projeler görüntülenebilmektedir.



Şekil 22. Cloud Firestore ile yeni proje oluşturulması

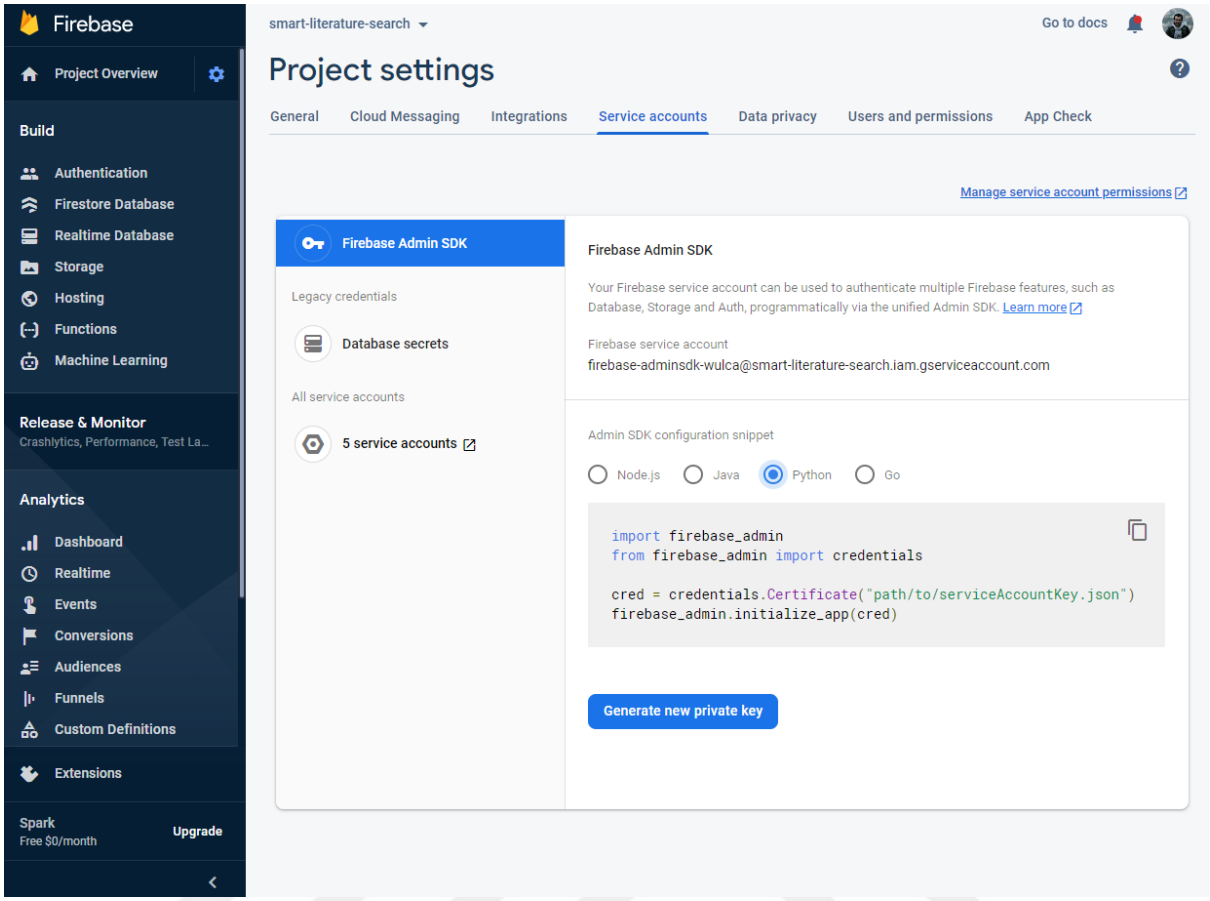
Yönetim ekranında proje oluşturulduktan sonra projeye girilerek Firestore Database menüsünden veritabanı oluşturma işlemlerine başlanır. Şekil 23' de projeye ait Firestore Database ekranları görüntülenmektedir.



Şekil 23. Cloud Firestore ile veritabanı oluşturulması

Standart veritabanı mantığındaki her bir tablo bu NoSQL tabanlı veritabanında "collection" olarak isimlendirilmekte ve içerisinde üretilmiş her bir satır veri de "document" olarak adlandırılmaktadır. Collection'lar oluşturulduktan sonra içerisine JSON tipinde veriler atmaya hazır hale gelmiştir.

Oluşturulan veritabanını uygulamaya entegre etmek için proje ve veritabanına özgü "private key" dosyasının üretilmesi gerekmektedir. Şekil 24' te görüldüğü üzere projeye ait ayarlar ekranından "Generate new private key" butonu tıklanarak üretilebilmektedir.



Şekil 24. Cloud Firestore veritabanı için private key oluşturulması

"Private key" oluşturulduktan sonra uygulamada kullanmak üzere "config.json" isminde bir dosya üretilmektedir. Bu dosya uygulama içerisinde Şekil 25'te gösterildiği gibi Firebase kütüphaneleri ile kullanıldığında uygulama ile veritabanı bağlantısı sağlanmış olur.

config.json

app > serviceReferences > FirebaseServiceReference > config.json > ...

```
{
  "type": "service_account",
  "project_id": "smart-literature-search",
  "private_key_id": "c9d0e26670a8d30c0db48a61d4e8e6fdc1433481",
  "private_key": "-----BEGIN PRIVATE KEY-----\nMIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQChYoD2t
  "client_email": "firebase-adminsdk-wulca@smart-literature-search.iam.gserviceaccount.com",
  "client_id": "111778198829604046843",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://oauth2.googleapis.com/token",
  "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
  "client_x509_cert_url": "https://www.googleapis.com/robot/v1/metadata/x509/firebase-adminsdk-wulca%40s
}

```

FirebaseServiceReference.py

app > serviceReferences > FirebaseServiceReference > FirebaseServiceReference.py > ...

```python
import uuid
import firebase_admin
from firebase_admin import firestore
from firebase_admin import credentials

def init():
    try:
        if not firebase_admin._apps:
            cred = credentials.Certificate("./serviceReferences/FirebaseServiceReference/config.json")

            global app

            app = firebase_admin.initialize_app(cred, {
                'projectId': 'smart-literature-search'
            })

        return firestore.client()
    except Exception as ex:
        print(ex)
```

Şekil 25. Cloud Firestore veritabanının uygulamaya entegre edilmesi

Bu adımdan sonra artık veritabanı alışverişi yapılabilecektir. Projede kullanılan veritabanı yapısı Şekil 26' da gösterilmiştir.



Şekil 26. Veritabanı Diyagramı

3.3.3. Servis katmanı

Geliştirilecek olan arkayüz mimarisinde arayüzün veritabanı erişimini ve kullanıcıdan gelen isteklere cevap vermesini sağlamak amacıyla uygulama için bir servis geliştirilmiştir.

Geliştirilen sistemde, kullanıcı taleplerini hızlı bir şekilde gerçekleştirilebilmek ve performansı üst seviyeye çıkarmak amacıyla tüm işlemler multithread yöntemleriyle asenkron olarak çalışmaktadır. Uzun sürede gerçekleşen işlemlerde kullanıcının diğer işlemlerini etkilememesi için arkaplanda çalışan işler (background tasks) koşturulmaya başlatılır.

Tarama ve kazıma işlemlerini tamamlayıp hızlı bir şekilde kullanıcıya sonuçları gönderme görevini üstlenen servisimiz daha önce hazırlanan tarama ve kazıma algoritmalarıyla da uyumlu olması için Python dilinde nesne yönelimli (object-oriented) programlama temellerine uygun ve geliştirmelere açık olacak şekilde geliştirilmiştir.

3.3.3.1. Uygulama Programlama Arayüzü (API) nedir?

Application Programming Interface (API) yani "Uygulama Programlama Arayüzü", bir uygulamanın işlevlerine dışarıdan veya uzaktan erişilip bu işlevlerin kullanılmasını sağlayan arayüzdür. API, bir sunucunun üzerindeki uygulamaya farklı platformlardan ulaşılmasını ve cevap dönmesine olanak sağlar. Web API' lerinin tamamı Representational State Transfer (REST) mimarisi üzerinde tasarlanır. Bundan dolayı platform bağımsız çalışır. Bu mimari GET, POST, PUT, DELETE metotlarının hepsini desteklemektedir. API' lerin cevap verdiği veri türleri talebe göre JSON, XML veya başka bir veri formatı olabilmektedir [44].

API' nin en üst katmanını oluşturan "endpoint" olarak da adlandırılan uygulamada istenilen kısma istek aratmaya yarayan yollara da "routers" denilmektedir. Örneğin API "http://localhost:8080" ile hizmet veriyorsa bunun altında da kullanıcı işlemlerini yapmaya yarayan "/users" endpoint ya da route'una http://localhost:8080/users şeklinde istek gönderilerek ulaşılabilir. Uygulama kurgusunda buraya gönderilen her istek için uygulamanın nasıl cevap vereceği geliştirilir. Genellikle eğer bir GET ile buraya istek gönderiliyorsa API uygulama kullanıcılarını cevap olarak döner, eğer kullanıcı bilgilerini içeren bir POST isteği atılıyorsa uygulamada yeni bir kullanıcı oluşturulur, PUT isteği

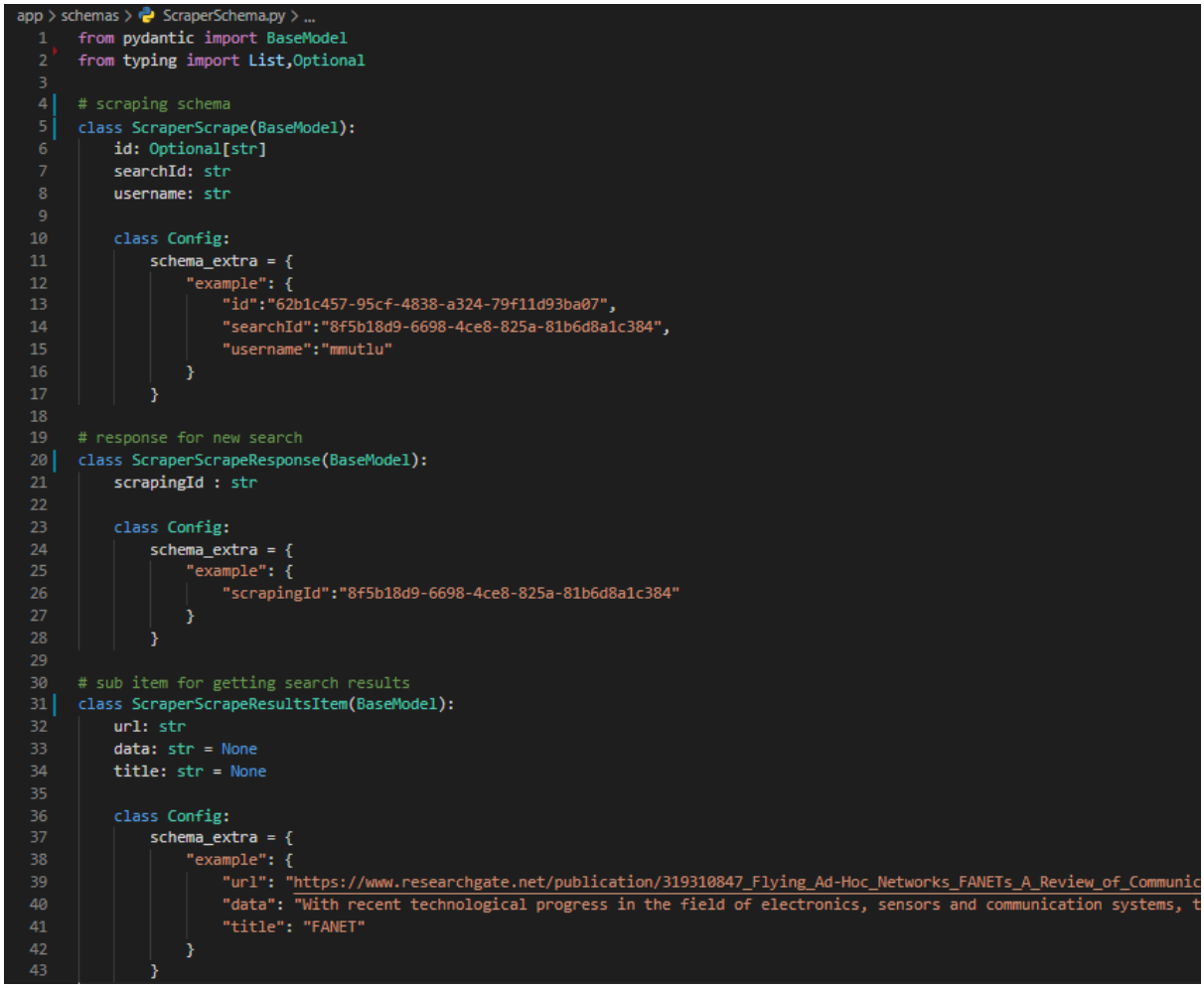
iletiliyorsa güncelleme; DELETE isteği gönderiliyorsa kullanıcı silme işlemleri gerçekleştirilir.
Proje geliştirilirken Python programlama dilinde API geliştirmelerinde popüler olarak kullanılan FAST API framework' ünden faydalanılmıştır.

3.3.3.2. FAST API

FAST API Framework Python versiyon 3.6'dan sonra kullanılabilen; geliştirmesi basit, modern ve performansı oldukça gelişmiş, dünyada yaygın olarak kullanılan bir yazılım geliştirme kütüphanesidir. Son yıllarda popüler olmasıyla birlikte dökümantasyonunun kuvvetli olması ve geliştirme eforunun düşük olması sebebiyle açık kaynak kodlu FAST API kütüphanesinden faydalanılmıştır. İlk olarak 2018 yılında geliştirilmiş açık kaynak kodlu ücretsiz backend servis geliştirmelerine yarayan framework, son yıllarda aynı amaçla kullanılan Flask'ın yerini almıştır. Bu gelişmede, FAST API' nin hem hızlı oluşu hem de uygulama geliştirme iskeletinin kolayca ayağa kaldırılabilmesi rol oynamıştır [45-46].

3.3.3.3. Pydantic şema kütüphanesi

Pydantic, nesne yönelimli programlamanın temelinde yatan sınıf nesnelerinin yapısını tanımlama, konfigürasyonlarını yapma ve kullanılabilir python schema'ları oluşturabilmeye yarayan açık kaynaklı kütüphanedir. Hem swagger hem de FAST API ile uyumlu çalışması, Python schema nesnelerini daha okunabilir ve sade hale getirmesi sebebiyle projede sınıf tanımlamaları yapmak için kullanılmıştır. Pydantic kullanılarak servisimizin içerisindeki istek ve cevap nesneleri tanımlanmıştır. Bu sayede hem swagger dokümantasyonunda hem de kod okunabilirliği anlamında kolaylık sağlanmıştır. Şekil 27' de Pydantic ile oluşturulmuş Scraping işlemlerinde kullanılan şemaların kod örneği verilmiştir. Pydantic altındaki BaseModel kullanılarak Scraping API' de kullanılan şemalar oluşturulmuştur. BaseModel diye adlandırılan ana model içine tanımlanan her bir Config ile de Swagger dokümantasyonu içerisinde şemaya ait bir örnek tanımlanabilmektedir [47].



Şekil 27. Pydantic ile oluşturulmuş şema örneği

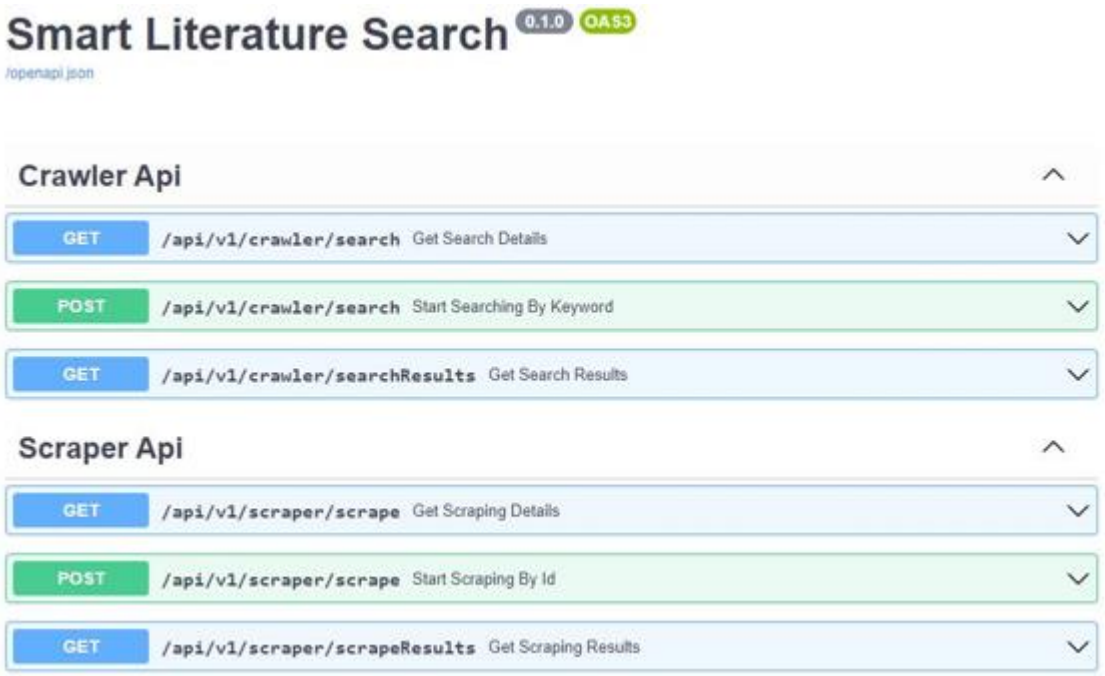
3.3.3.4. Swagger

Swagger, bir arkayüz servisinin dokümantasyonu sayılabilecek, servis içerisindeki endpointlere isteklerin nasıl atılması gerektiğiyle ilgili bilgiler verip; testler yapmaya izin veren açık kaynak kodlu dil bağımsız bir kütüphanedir. Günümüz uygulamalarında geliştirilmiş servis için bir kullanım dokümanı yazmak yerine hem testlere izin verebilen hem kullanım detaylarını veren swagger dokümantasyonunu oluşturmak tercih edilmektedir. Dil bağımsız olarak tüm ortamlarda çalışabilmektedir [48].

Swagger dokümantasyonu

Çalışmada kullanılan FAST API Framework'ü otomatik olarak swagger dokümantasyonu oluşturmaktadır. Oluşturulmuş api endpointlerinde kullanılmış Python

sınıflarını otomatik olarak algılayıp ve “api_address:port/docs” yoluyla servis etmektedir. Uygulamanın bir kısmına ait swagger dokümantasyonuna Şekil 28’ de yer verilmiştir.



Şekil 28. Swagger API dökümantasyonu

Crawler API dokümantasyonu

Geliştirilen uygulamada crawling işlemlerinden sorumlu Crawler API için oluşturulmuş swagger dokümantasyonudur.

Get Search Details

Açıklama: İstek içerisinde parametre gönderildiği takdirde daha önce yapılmış bir crawling işleminin detaylarını parametre gönderilmediği takdirde de daha önce yapılmış tüm crawling işlemlerinin detayını veren endpointtir. Cevap olarak CrawlerSearch şemasını kullanır.

Endpoint: /api/v1/crawler/search

Method: GET

İstek Şeması: -

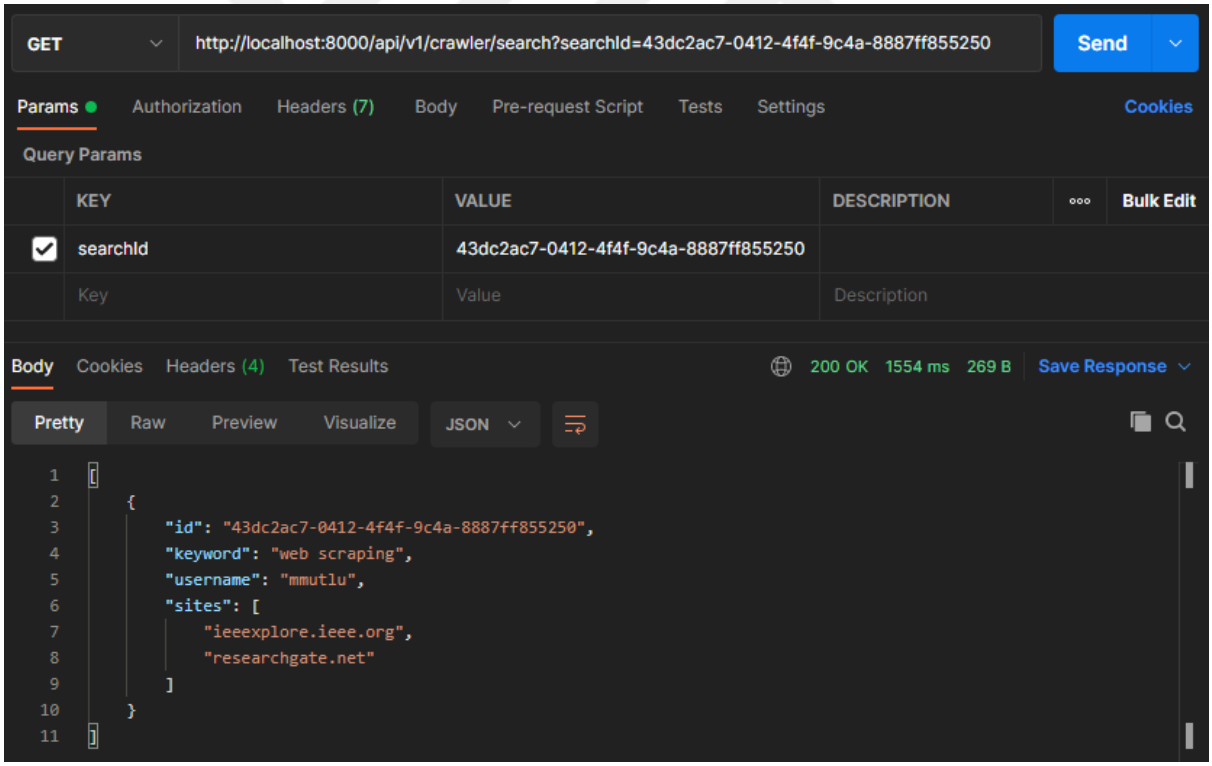
Parametreler:

Type	Name	Description	Schema
query	**searchId** *optional*		string

Cevap Şeması: CrawlerSearch

Name	Description	Schema
id *optional*	**Title**: Id	string
keyword *required*	**Title**: Keyword	string
username *required*	**Title**: Username	string
sites *required*	**Title**: Sites	< string > array

Kullanım Örneği:



Şekil 29. Crawler API Get Search Details kullanım örneği

Start Searching By Keyword

Açıklama: Gönderilen istekte body ile girilen keyword ve filtrelere göre crawling işlemlerini başlatan ve başlatılan arama sonucunda veritabanına kaydedilmiş search id'yi

cevap olarak döndüren endpointtir. Cevap olarak CrawlerSearchResponse şemasını kullanır.

Endpoint: /api/v1/crawler/search

Method: POST

İstek Şeması: CrawlerSearch

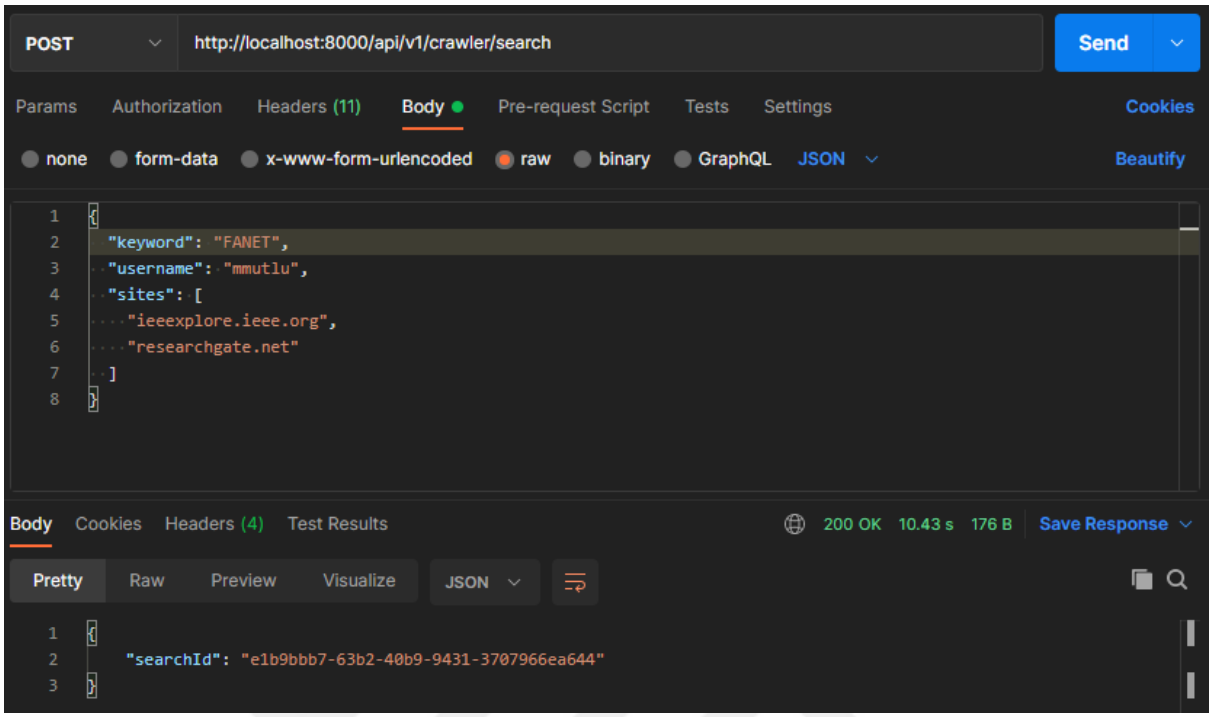
Name	Description	Schema
id *optional*	**Title**: Id	string
keyword *required*	**Title**: Keyword	string
username *required*	**Title**: Username	string
sites *required*	**Title**: Sites	< string > array

Parametreler: -

Cevap Şeması: CrawlerSearchResponse

Name	Description	Schema
searchId *required*	**Title**: Searchid	string

Kullanım Örneği:



Şekil 30. Crawler API Start Searching By Keyword kullanım örneği

Get Search Results

Açıklama: İstek içerisinde parametre gönderildiği takdirde daha önce yapılmış bir crawling işleminin sonucunu, parametre gönderilmediği takdirde tüm crawling işlemlerinin sonucunu veren endpointtir. Cevap olarak CrawlerSearchResultsResponse şemasını kullanır.

Endpoint: /api/v1/crawler/searchResults

Method: GET

İstek Şeması: -

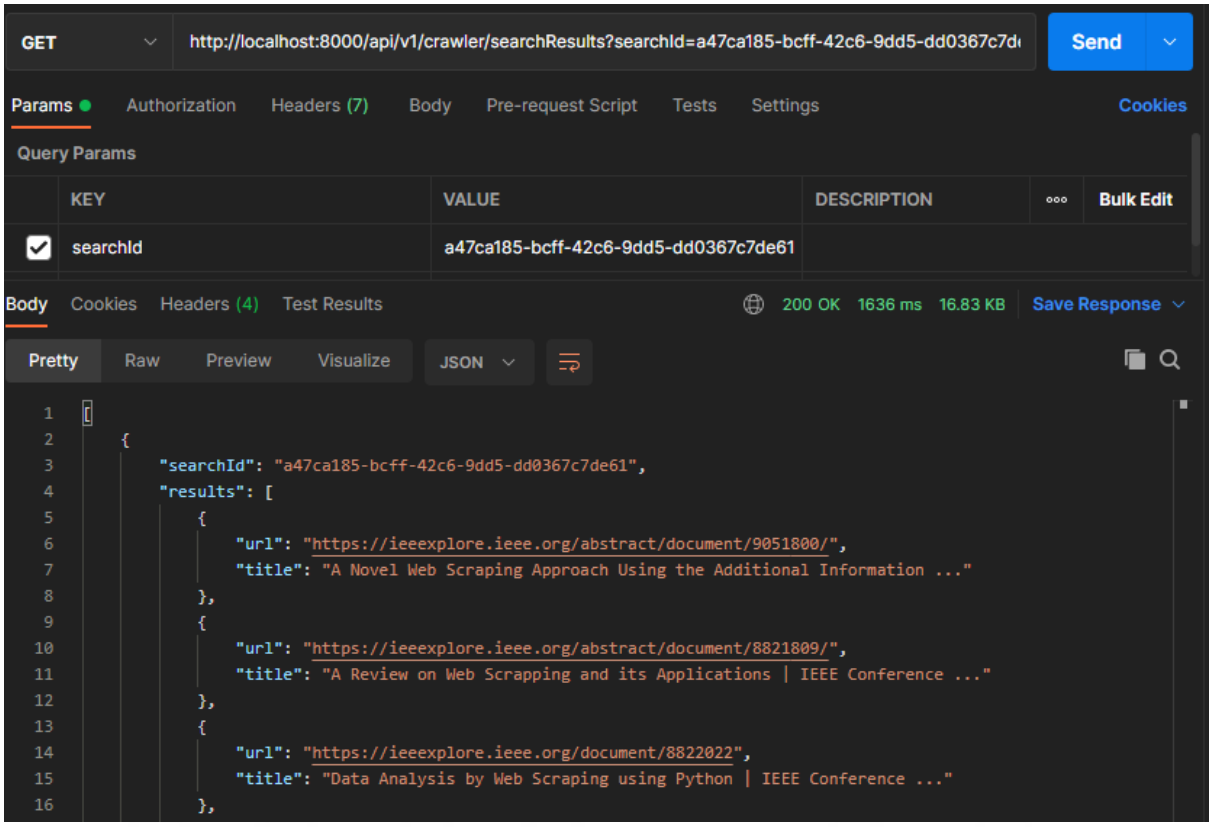
Parametreler:

Type	Name	Description	Schema
query	**searchId** *optional*		string

Cevap Şeması: CrawlerSearchResponse

Name	Description	Schema
searchId *required*	**Title**: Searchid	string

Kullanım Örneği:



Şekil 31. Crawler API Get Search Results kullanım örneği

Scraper API dokümantasyonu

Servis içerisindeki scraping işlemlerinden sorumlu Scraper API için oluşturulmuş swagger dokümantasyonudur.

Get Scraping Details

Açıklama: İstek içerisinde parametre gönderildiği takdirde daha önce yapılmış bir scraping işleminin detaylarını parametre gönderilmediği takdirde de daha önce yapılmış tüm scraping işlemlerinin detayını veren endpointtir. Cevap olarak ScraperScrape şemasını kullanır.

Endpoint: /api/v1/scraper/scrape

Method: GET

İstek Şeması: -

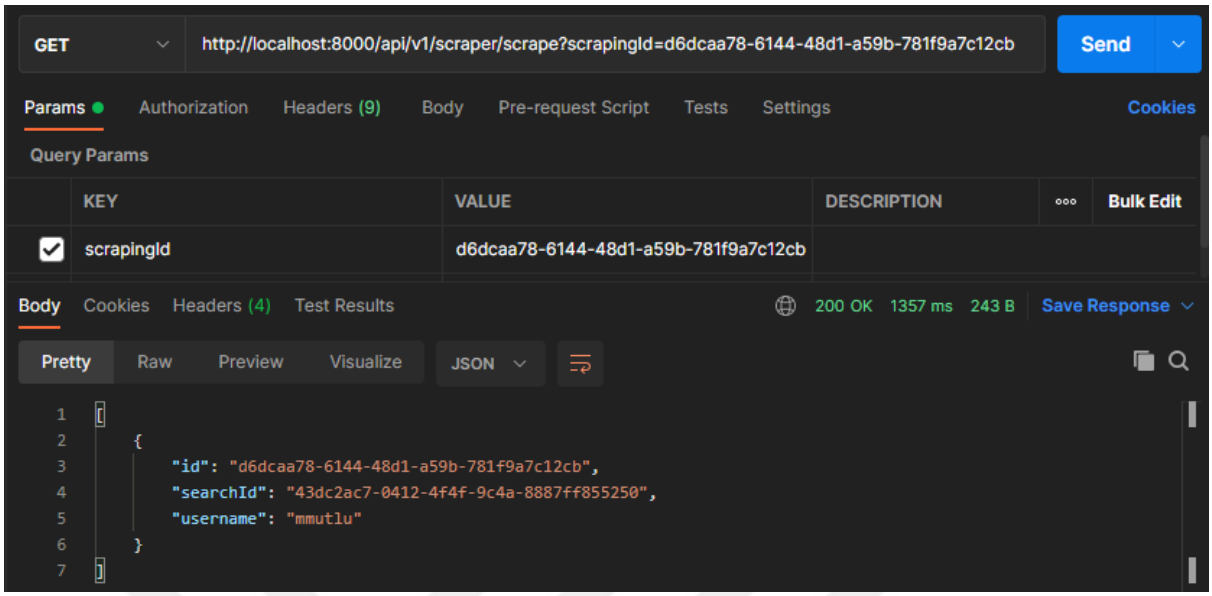
Parametreler:

Type	Name	Description	Schema
query	**scrapingId** *optional*		string

Cevap Şeması: ScraperScrape

Name	Description	Schema
id *optional*	**Title**: Id	string
searchId *required*	**Title**: Searchid	string
username *required*	**Title**: Username	string

Kullanım Örneği:



Şekil 32. Scraper API Get Scraping Details kullanım örneği

Start Scraping By Id

Açıklama: Tamamlanmış crawling id'sine göre scraping işlemlerini başlatan ve tamamlandığında scraping id'yi döndüren endpointtir. Cevap olarak ScraperScrapeResponse şemasını kullanır.

Endpoint: /api/v1/scraper/scrape

Method: POST

İstek Şeması: ScraperScrape

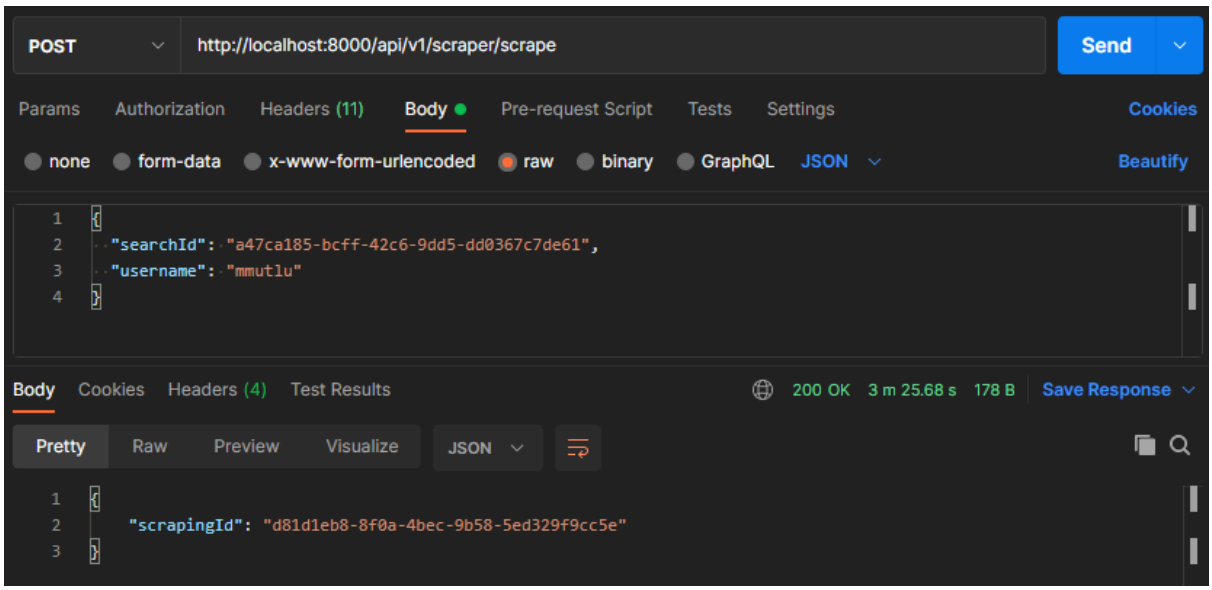
Name	Description	Schema
id *optional*	**Title**: Id	string
searchId *required*	**Title**: Searchid	string
username *required*	**Title**: Username	string

Parametreler: -

Cevap Şeması: ScraperScrapeResponse

Name	Description	Schema
scrapingId *required*	**Title**: Scrapingid	string

Kullanım Örneği:



Şekil 33. Scraper API Start Scraping By Id kullanım örneği

Get Scraping Results

Açıklama: Parametre girildiği takdirde daha önce yapılmış bir scraping işleminin sonucunu, parametre girilmediği takdirde tüm scraping işlemlerinin sonucunu veren endpointtir. Cevap olarak ScraperScrapeResultsResponse şemasını kullanır.

Endpoint: /api/v1/scraper/scrapingResults

Method: GET

İstek Şeması: -

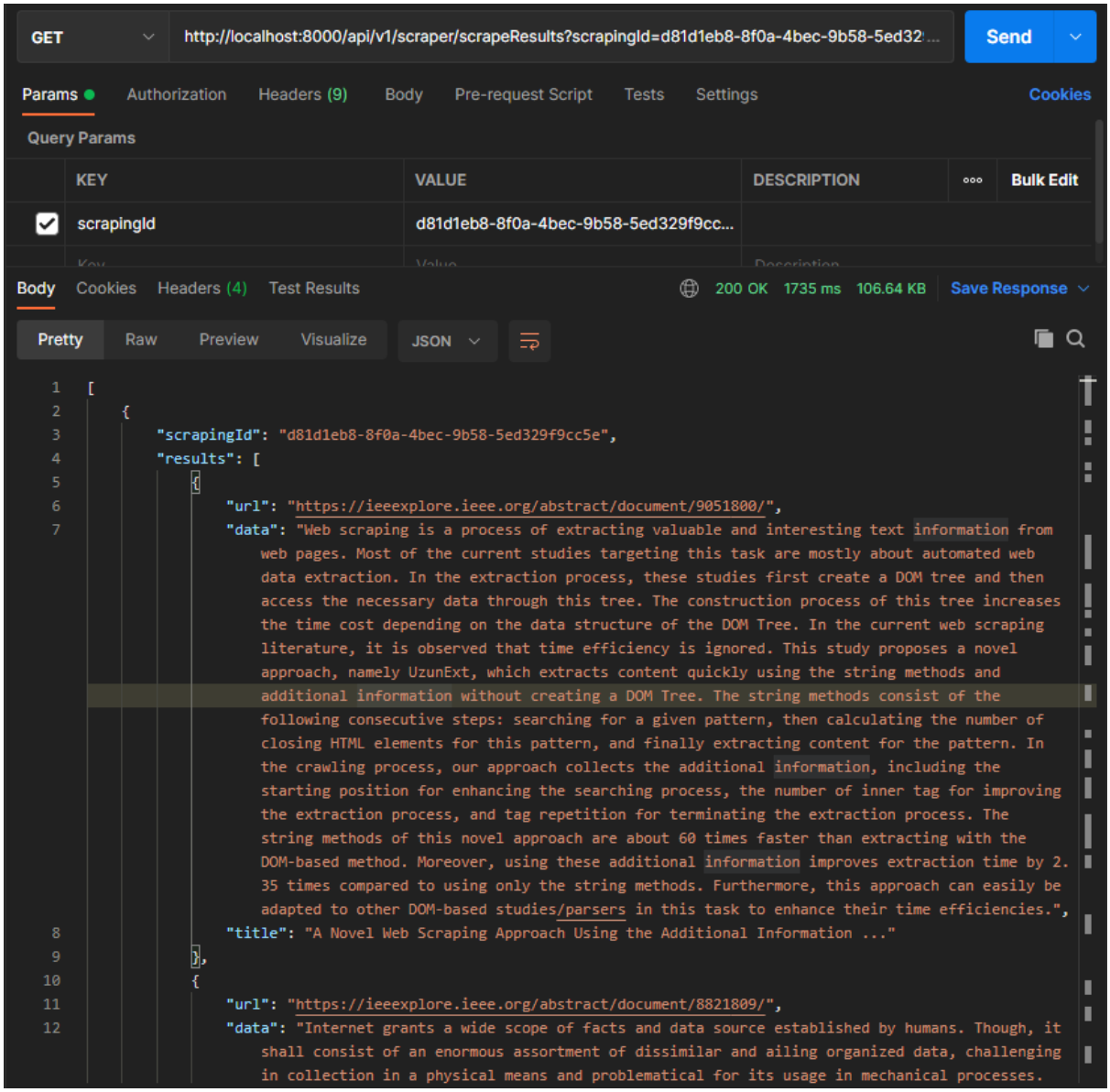
Parametreler:

Type	Name	Description	Schema
query	**scrapingId** *optional*		string

Cevap Şeması: ScraperScrapeResultsResponse

Name	Description	Schema
scrapingId *required*	**Title**: Scrapingid	string
results *required*	**Title**: Results	< ScraperScrapResultsItem > array

Kullanım Örneği:



Şekil 34. Scraper API Get Scraping Results kullanım örneği

3.3.3.5. Uvicorn

Uvicorn Python dilinde geliştirilen servislerin sürekli çalışabilmesini sağlayabilecek bir ASGI sunucusudur. Geliştirdiğimiz FAST API projesi Uvicorn yapısı kullanılarak derlenmekte ve çalışmaktadır. Şekil 35' te projeyi uvicorn ile çalışmasını sağlayan main.py dosyası gösterilmektedir.

main.py M

app > main.py > ...

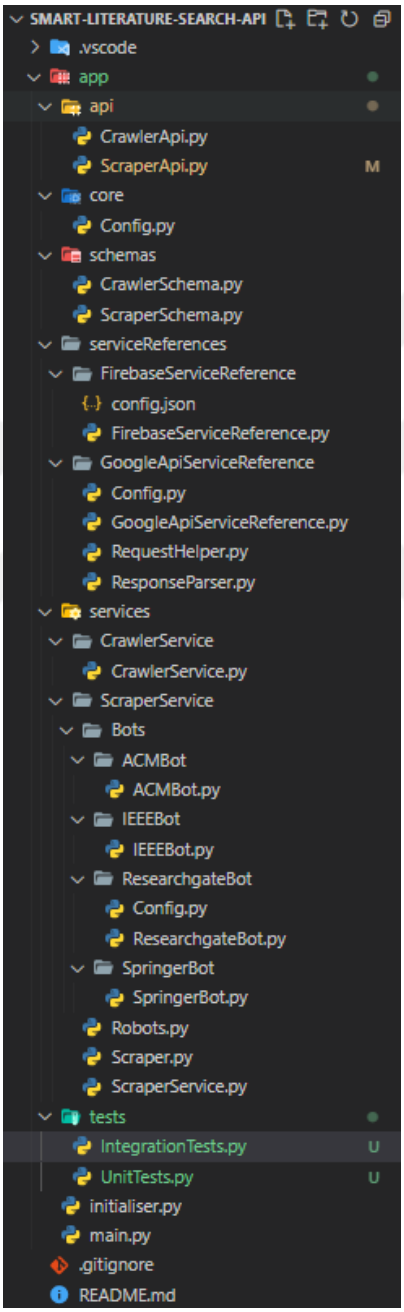
```python
#3rd party
from fastapi import FastAPI

# api routers
from api.CrawlerApi import api_crawler_router
from api.ScraperApi import api_scraper_router

# api configs & initialiser
from core import Config
from initialiser import configureCors

# create uvicorn app
app = FastAPI(title=Config.settings.PROJECT_NAME)

# configure cors
app = configureCors(app)

# define routes
app.include_router(api_crawler_router,prefix=Config.settings.API_V1_STR)
app.include_router(api_scraper_router,prefix=Config.settings.API_V1_STR)
```

Şekil 35. Uvicorn ile çalışacak proje main dosyası

Bu kod bloğunda, projede geliştirilmiş Crawler ve Scraper API' lerinin yönlendiricileri (routerları) ve proje özelinde ihtiyaç duyulan diğer yapılandırıcılar (config) dahil edilmiştir. Sonrasında FastAPI ile bir uvicorn uygulaması oluşturulmuştur. İlgili kod parçacığını terminal aracılığı ile "uvicorn main.py" komutu ile çalıştırdığımızda lokal makinede bir Uvicorn Sunucusu ayağa kaldırarak uygulamayı çalıştırmaktadır [49].

3.3.3.6. Proje iskeleti

Python FAST API mimarisiyle kurgulanmış proje iskeleti altı adet ana klasör ve alt klasörlerden oluşmaktadır. Şekil 36' da arkayüz uygulamasının proje iskeleti gösterilmektedir.



Şekil 36. Arkayüz proje iskeleti

Schemas klasörü

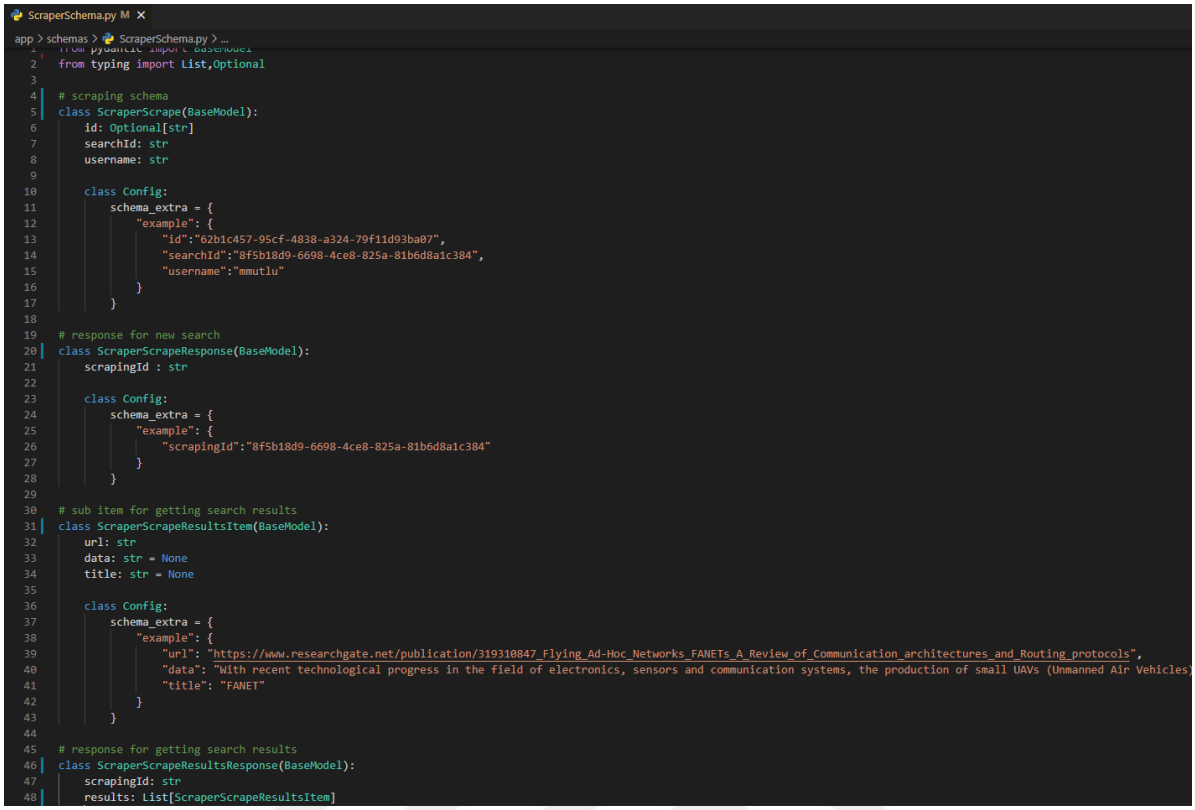
API' ye gelen tüm istekler ve cevaplar için Bölüm 3.3.2.3'de anlatılan Pydantic şemalar tanımlanmıştır. Şekil 37 ve 38' de kod bloklarında görüldüğü üzere Crawler için CrawlerSchema altında Scraper için ScraperSchema dosyası altında geliştirme yapılmıştır.

CrawlerSchema.py M

app > schemas > CrawlerSchema.py > ...

```python
from pydantic import BaseModel
from typing import List,Optional

# search schema
class CrawlerSearch(BaseModel):
    id: Optional[str]
    keyword: str
    username: str
    sites: List[str]

    class Config:
        schema_extra = {
            "example": {
                "id":"de277f84-50f3-41b9-b03f-49b5e49a8e64",
                "keyword": "FANET",
                "username": "mmutlu",
                "sites": ["ieeexplore.ieee.org","researchgate.net"]
            }
        }

# response for new search
class CrawlerSearchResponse(BaseModel):
    searchId : str

    class Config:
        schema_extra = {
            "example": {
                "searchId":"8f5b18d9-6698-4ce8-825a-81b6d8a1c384"
            }
        }

# sub item for getting search results
class CrawlerSearchResultsItem(BaseModel):
    url: str
    title: str

    class Config:
        schema_extra = {
            "example":{
                "url":"https://www.researchgate.net/publication/327579184_On_the_Issues_of_Communication_in_FANET_A_Light_Fidelity_Based_Approach",
                "title":"(PDF) On the Issues of Communication in FANET: A Light Fidelity ..."
            }
        }

# response for getting search results
class CrawlerSearchResultsResponse(BaseModel):
    searchId: str
    results: List[CrawlerSearchResultsItem]
```

Şekil 37. Crawler API için geliştirilen Python şeması



Şekil 38. Scraper API için geliştirilen Python şeması

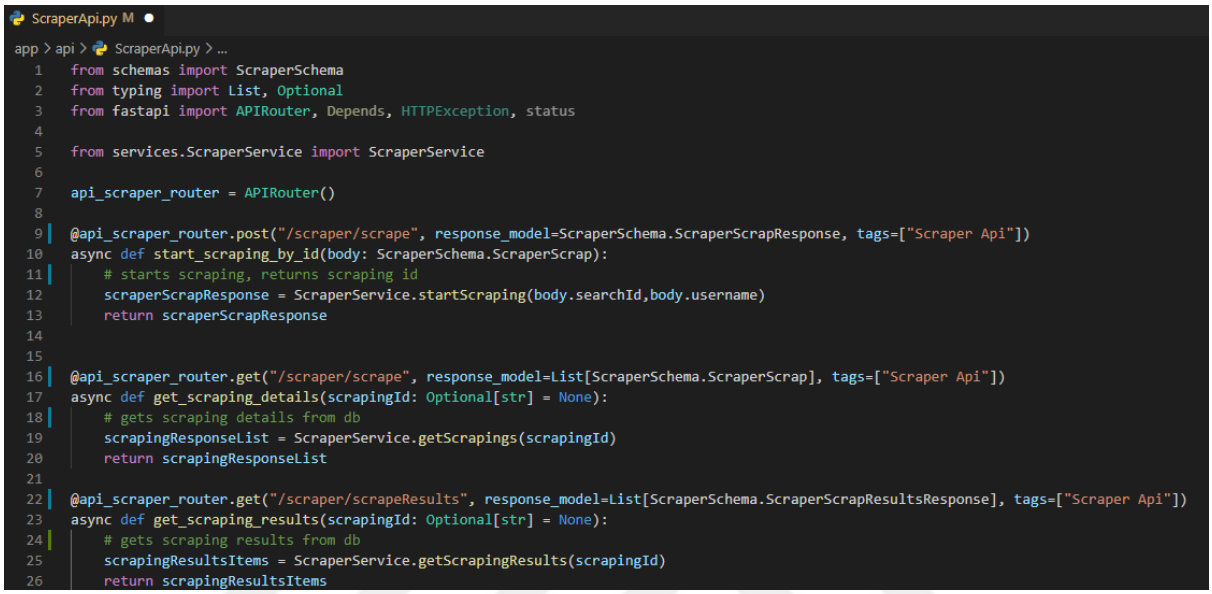
Testler

Bir yazılım geliştirilirken uygulamanın uçtan uca testlerinin manuel olarak yapılması oldukça zordur. Modern yazılım dünyasında bir projede testlerin otomatik olarak yapılabilmesini sağlayan entegrasyon (integration) ve birim (unit) testleri geliştirmeleri bulunmaktadır. Bu testler sayesinde uygulama geliştirmeye başlamadan önce, geliştirmeyi yaparken ve tamamladıktan sonra projenin hataları ve sonuçları hakkında bilgiler vermektedir. Integration test, uygulamanın içerisinde çalışan modüllerin birbiri ile uyumu hakkında testler yapmaya yararken; unit test de her bir modülün kendi içerisinde doğru çalışıp çalışmadığıyla ilgili testler yapabilmeyi sağlamaktadır.

Api

Arayüzden gelen tüm istekler için farklı amaçları karşılayan API' ler geliştirilmiştir. Crawling işlemlerini başlatması ve veritabanına yazma işlemini gerçekleştirmesi için Crawler API, crawl edilmiş URL'ler üzerinde scraping işlemlerini yapıp makale detaylarını veritabanına yazması için Scraper API geliştirmeleri yapılmıştır. İlgili API klasörün altında bulunan Crawler API "/api/v1/crawler" endpointine gelen isteklere,

Scraper API ise “/api/v1/scraper” endpointine gelen isteklere cevap vermekle yükümlüdür. Gelen isteğe göre uygulama içerisinde geliştirilmiş servis algoritmalarını çalıştırır ve sonucu önyüze döner. Şekil 39’da Scraper API’ye ait kod bloğu görüntülenmektedir.



Şekil 39. Api kod örneği

Scraper API için route’ları tanımlayan ve bu routelara gelen istekler için yapılacak işlemleri gerçekleştiren kod bloğudur. İsteği yakalayıp yapılması gereken işi servise iletir. Burada örnek olarak ilk route olan “scraper/scrape” için tanımlanmış scraping işlemini başlatmaya yarayan POST isteği incelenirse; İlk olarak “start_scraping_by_id” isminde asenkron bir fonksiyon tanımlanmıştır. Ve parametre olarak da ScraperSchema altında ScraperScrap türünde bir body beklemektedir. İşlemler tamamlandığında da ScraperSchema altında ScraperScrapResponse türünde bir nesne dönecektir. Route’ a atılan istekteki searchId yani daha önce yapılmış crawling işlemine ait id ve username ScraperService’indeki “startScraping” metoduna gönderilecek ve gelen cevap da istek sonucunda döndürülecektir.

Services klasörü

Çalışan uygulamada API route’larının yakaladığı ve ilettiği istekleri gerçekleştirmek üzere servisler tanımlanmıştır. Crawler Api tarafından “/crawler/” endpointine gelen istekler için CrawlerService isminde, Scraper Api tarafından “/scraper/” endpointine

gelen istekler için ScraperService isminde algoritmalar geliştirilmiştir. Şekil 40'da ScraperService içerisindeki bir kod parçacığına yer verilmiştir.

ScraperService.py

app > services > ScraperService > ScraperService.py > ...

```python
# 3rd party imports
import requests

# internal imports
from schemas import ScraperSchema
from serviceReferences.FirebaseServiceReference import FirebaseServiceReference
from .Scraper import scrapSites

def startScraping(searchId, username):
    # insert scraping details to db
    scrapingId = FirebaseServiceReference.insertScraping(searchId, username)

    print("insert scraping finished")

    # get search results from db
    searchResults = FirebaseServiceReference.readSearchResults(searchId)
    for doc in searchResults:
        docObject = doc.to_dict()
        searchResultsObject = docObject["results"]

    print("read search results finished")

    # scrap by beautifulsoup
    scrapingResults = scrapSites(searchResultsObject)

    print("scraping finished")

    # insert search results
    FirebaseServiceReference.insertScrapingResults(scrapingId, scrapingResults)

    print("insert scraping results finished")
    # start search, return search id
    scraperScrapingResponse = ScraperSchema.ScraperScrapResponse(scrapingId=scrapingId)

    return scraperScrapingResponse
```

Şekil 40. Servis kod örneği

Scraper API tarafından kazıma başlatma emri ScraperService' deki startScraping metoduna geldiği anda ilk olarak searchId ve username FirebaseServiceReference sınıfındaki "insertScraping" metodu ile veritabanına kaydedilir. Yine aynı sınıf içerisindeki "readSearchResults" ile veritabanında daha önceden yapılmış bu searchId'ye sahip tarama sonuçları alınır. Sonuçlarda yer alan web sitelerinin kazıma işlemleri Bölüm 3.2.2'de anlatılmış kazıma botları ile başlar. Makale özetlerini içeren sonuçlar elde edildikten yani kazıma işlemleri tamamlandıktan sonra yine FirebaseServiceReference

sınıfındaki “insertScrapingResults” metodu ile veritabanına kaydedilir. Veritabanına kaydedilen verinin id’si ile Scraper şemasındaki ScraperScrapResponse nesnesi oluşturulur ve bu sonuç API’nin cevap olarak iletebilmesi için döndürülür.

ServiceReferences klasörü

ServiceReferences altında uygulama içerisindeki servislere yardımcı olarak kullanılan bir başka yardımcı API ya da SDK ile iletişimler için oluşturulmuş sınıflar yer almaktadır. Tarama işlemlerinde kullanılmak üzere Google Search JSON API ile iletişim kuran ve işlemleri gerçekleştiren GoogleApiService sınıfı, veritabanı işlemlerinde kullanılmak üzere Firebase Cloud Firestore ile iletişim kuran ve işlemleri gerçekleştiren FirebaseServiceReference sınıfları oluşturulmuştur. Şekil 41’de FirebaseServiceReference sınıfı içerisindeki kod parçacığına yer verilmiştir.



Şekil 41. Service Reference kod örneği

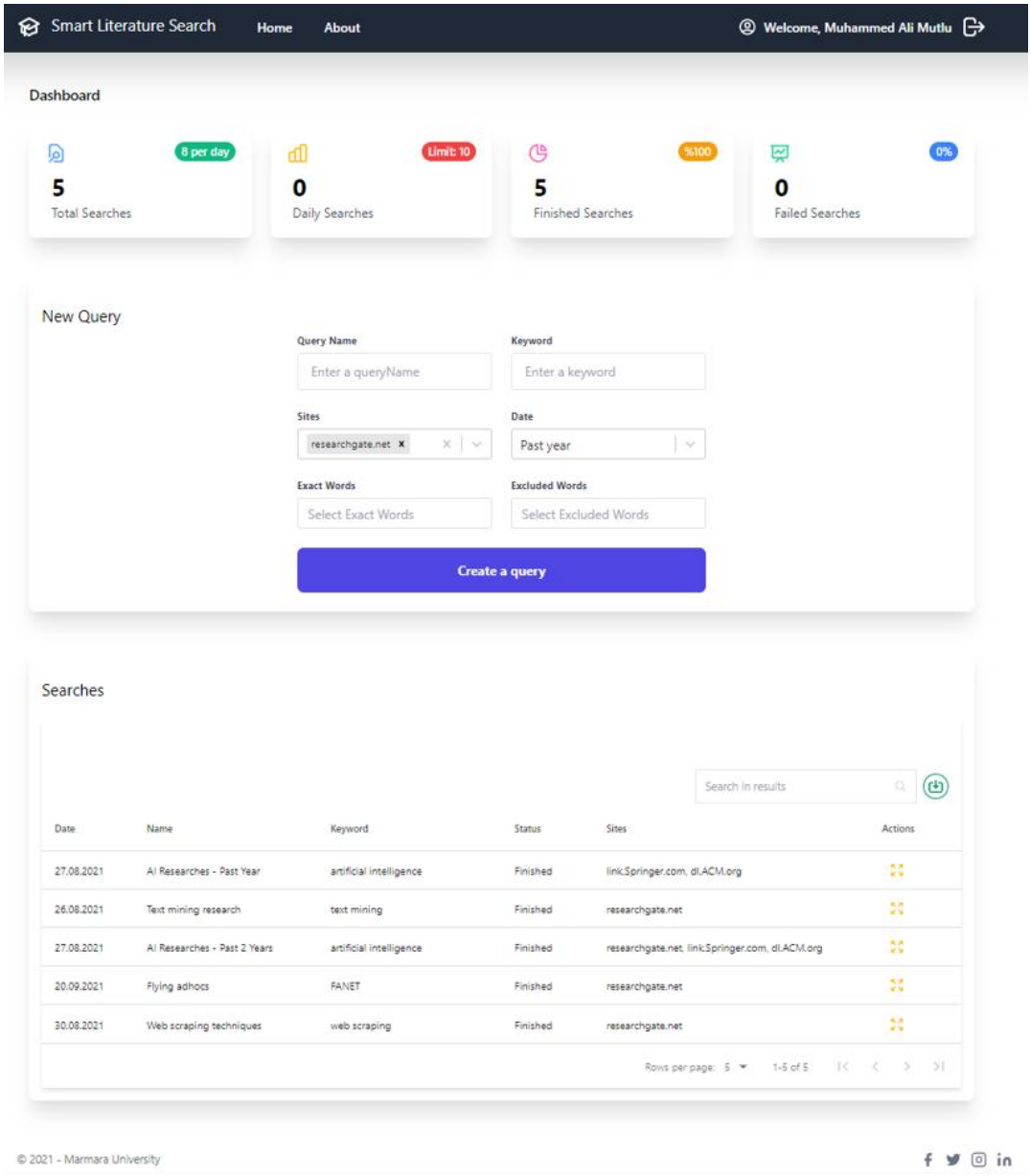
“Scraping service” içerisindeki veritabanı etkileşiminde de şekil 40’daki kod parçacığında da görüldüğü üzere bu sınıf kullanılmıştır. Kazıma başlatma işlemi geldiğinde ilk olarak bu referans sınıftaki “readSearchs” metodu ile veritabanına gidilir ve “searchResults” collection’unun altından bu tarama işleminin sonuçlarını içeren doc’a ulaşılır. Kazıma işlemleri tamamlandıktan sonra yine bu sınıftaki “insertScraping” ile kazıma sonuçları veritabanına yazılır.

3.4. Önyüz (Frontend) Geliştirmeleri

Geliştirilmiş Smart Literature Search uygulamasının etkin bir şekilde kullanılabilmesi için kullanıcının kullanımı kolay bir arayüzle, hızlı bir şekilde araştırmalarını yapabilmesi sağlanmalıdır. Önyüz, bir uygulamanın, yazılımın ya da web sitesinin kullanıcı tarafından görüntülenen görsel yüz kısmıdır [50]. Önyüz bölümünün kullanıcıdan gelen komutları yakalama, arkayüz tarafına istek gönderme ve gelen cevaba göre kullanıcıya sonuçları iletme görevleri bulunmaktadır.

3.4.1. Uygulama katmanı

Günümüz modern yazılım dünyasında önyüz web uygulaması geliştirmelerinde farklı yöntemler ve diller kullanılabilmektedir. Kullanıcının tüm tarama ve kazıma işlemlerini güvenli ve hızlı bir şekilde yapabilmesi için kullanıcı dostu bir arayüz geliştirilmiştir. Hızlı bir şekilde servis ile haberleşip arama filtrelerini seçerek arkayüz tarafında arama başlatma, başlatılan aramanın statüsünü görüntüleme ve arama sonucunda elde edilen verileri görüntüleme işlemleri bu arayüz üzerinden gerçekleştirilebilmektedir. Arayüz geliştirilmesi yaparken Javascript dili ve React kütüphanesi kullanılmıştır. Günümüz popüler teknoloji standartlarına uyması için de uygulama Tek Sayfa Uygulama (Single Page Application) olarak geliştirilmiştir. Şekil 42'de uygulama arayüzü görüntülenmektedir.



Şekil 42. Uygulama arayüzü

3.4.1.1. Tek Sayfa Uygulama (Single Page Application) nedir?

Tek Sayfa Uygulama (Single Page Application), uygulamanın tüm ekran ya da sayfaları için yalnızca bir HTML web sayfasını kullanan, uygulama içi herhangi bir path yönlendirmesi olmayan ve son kullanıcı etkileşimleri için arkaplanda Javascript, HTML ve CSS kullanan web uygulamasıdır. Geleneksel geliştirme yöntemlerinde kullanıcının her bir etkileşimi ya da yönlendirmesi (navigasyonu) web sunucusuna gider ve uygulama istenilen sayfayı web sunucusunda HTML olarak yükleyerek çalıştırır. SPA geliştirmelerinde kullanıcı etkileşimleri önyüzde tek bir HTML sayfasının içeriği

değiştirilerek karşılanmaktadır. Bu sayede uygulama içindeki geçişler hızlı bir şekilde yapılmaktadır. Ek olarak kullanıcı için tek bir sayfada kullanım kolaylığı sağlamaktadır. [50].

3.4.1.2. React

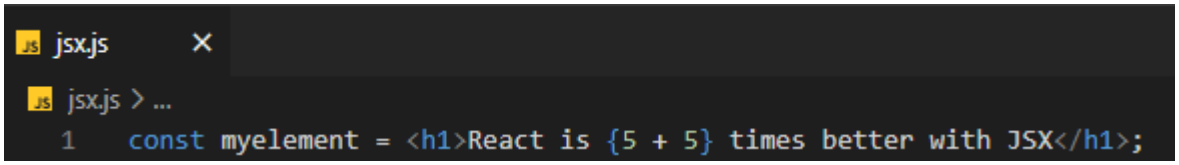
React, kullanıcı arayüzü oluşturmaya yarayan açık kaynak kodlu bir javascript kütüphanesidir. Facebook tarafından ilk olarak Facebook.com geliştirmesinin daha düzenli ve daha modern hale getirilmesini sağlamak amacıyla 2013 yılında geliştirilmiştir. Model-View-Controller prensibine dayanmaktadır. Genellikle tek sayfa uygulama geliştirmeleri yapıldığı gibi, React-Native kütüphanesi ile mobil uygulamalar da geliştirilebilmektedir [52]. Web uygulamalarını geliştirmeyi basit hale getirdiği, daha modern bir yapısı olduğu ve web topluluğu oldukça fazla olduğu için tercih edilmiştir. React kütüphanesinin altındaki "render" fonksiyonu içerisine yazılmış tüm javascript kodlarını HTML' e dönüştürüp web sayfası olarak sunmaktadır. Yapılan çalışmada Frontend geliştirmelerinde tamamıyla React kullanılmıştır.

ES6

ES6, EcmaScript sürüm 6' nın kısaltmasıdır. Dünyada yaygın olarak kullanılmakta olan Javascript dili geliştirilirken "Ecma International" şirketi tarafından dilin standartları belirlenmiştir. Bu standartlara da "EcmaScript" ismi verilerek markalaştırılmıştır. Günümüz Javascript dünyasında en güncel versiyon EcmaScript 12'dir. 2015 yılında çıkan ES6 versiyonuyla birlikte Javascript dilinde köklü değişikliklere gidilmiş ve dilin günümüzdeki modern iskeleti ortaya koyulmuştur. Her yeni versiyonda yeni özellikler eklense de ES6'daki kadar büyük bir güncelleme henüz gerçekleşmemiştir. React kütüphanesi de Javascript ES6 ve sonrasını desteklemektedir.

JSX

JSX, Javascript XML demektir. Yani Javascript kodunun içerisinde yazılmış XML' ler ile birlikte yeni bir söz dizimidir (syntax). JSX söz dizimi Javascript kodunun içerisinde HTML kodlarının yazılabilmesini sağlamaktadır. Şekil 43' te JSX söz dizimi ile yazılmış bir kod parçacığı verilmiştir.

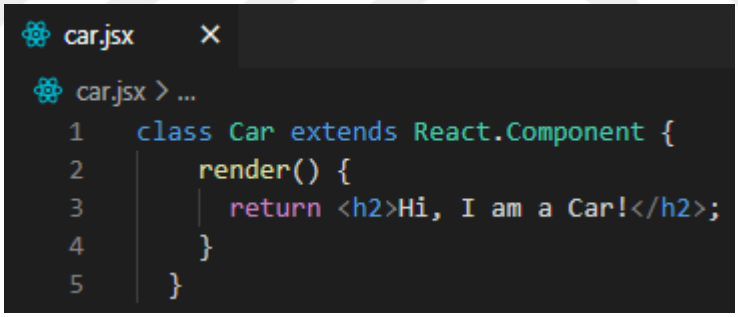


Şekil 43. JSX formatında yazılmış kod örneği

Bu kod parçacığına göre React ile oluşturulmuş web sayfasında bir "h1" HTML elementi içerisinde "{}" parantezleri arasında bir javascript toplama işlemi yapılmıştır. Bu element web sayfasında "React is 10 times better with JSX" şeklinde çıktı gösterir.

Components

JSX söz dizimine dayalı React web projesi içerisinde geliştirilmiş yeniden kullanılabilen kod parçacıklarına component denilmektedir. Tüm React projesi componentler üzerine kurulur ve componentlerin herhangi birinin içeriği değiştiğinde sadece o componenti yeniden derler. Javascript fonksiyonlarıyla aynı amaca hitap etmesiyle birlikte React render metodu ile kullanıldığında HTML formatında bir çıktı üretir. Şekil 44' te üretilmiş bir component görüntülenmektedir.

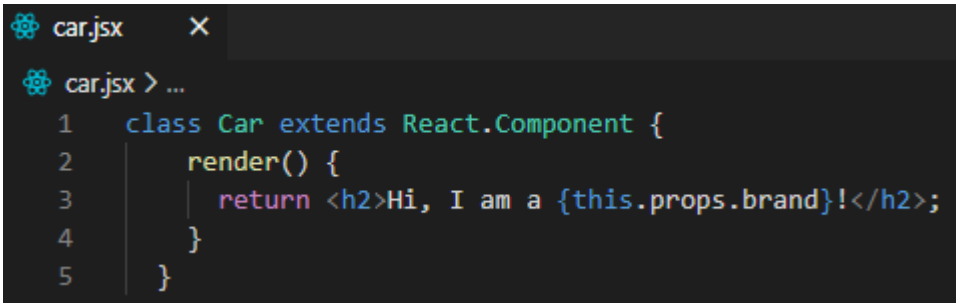


Şekil 44. React component örneği

Projenin istenilen sayfasında üretilmiş bu component tıpkı bir HTML elementi gibi "<Car/>" şeklinde çağrılarak kullanılabilmektedir.

Props

Bir javascript fonksiyonuna gönderilen parametre gibi React component'ine de parametre gönderilebilmektedir. Bu parametreler de props olarak adlandırılmaktadır. Component kullanılırken prop'un ismi ve değeri "<Car brand="Mercedes"/>" gibi gönderilir. Şekil 45'te Car componenti aldığı props ile kullanılmıştır.



Şekil 45. React componentinin props ile kullanılması

Artık bu component kullanıldığı yerde "Hi, I am a Car!" şeklinde bir "h2" elementi üretmek yerine kendisine gönderilen parametrede gelen props değerindeki brand'e göre dinamik olarak değişecek ve aracın markasını da içeren bir "h2" elementi üretecektir.

State

React uygulama içerisindeki herhangi bir component değeri değiştiğinde sadece o componenti yeniden render ettiği Bölüm 3.4.1.2.3'de bahsedilmiştir. Bu değişikliği algılaması için de componentin o anki değerlerini ifade eden nesnelerine de state denmektedir. Şekil 46' da state'in kullanım örneği verilmiştir.



Şekil 46. React componentinin state ile kullanılması

3.4.1.3. Next

React' ın popüler kullanımıyla birlikte React geliştirmelerini daha kolay hale getirmek için frameworkler çıkarılmıştır. Next.js aynı amaçla ortaya çıkmış, React'taki web uygulamasındaki navigasyonları daha kolay hale getiren, web sayfasının render etme işlemlerini tarayıcı yerine frameworkün ayağa kaldırdığı mini sunucuda gerçekleştiren bir yazılım geliştirme aracıdır [54].

Frontend uygulamasında React geliştirme framewörk' ü olarak Next tercih edilmiştir.

3.4.1.4. Redux

React'ta geliştirme yaparken proje büyüdükçe componentlarda kullanılan prop' ların ve state' lerin takibi oldukça zorlaşmaktadır. Bu yüzden orta büyük ölçekli React projelerinde merkezi bir state yönetimine ihtiyaç duyulur. Yazılım dünyasında en üstteki

parent componentten uygulamanın en ucundaki child componentine state ya da prop' ları güvenli ve kolay bir şekilde aktarabilen "state management" kütüphaneleri bulunmaktadır.

Web topluluğu oldukça geniş olan ve dokümantasyonu başarılı olan hızlı ve kolay geliştirilebilir olan Redux kütüphanesi ile birlikte uygulama veri yönetimi mekanizması için de state management işlemleri kolaylaştırılmıştır. Bu kütüphane sayesinde kullanıcının eriştiği ilk ekran olan üye girişi ekranından elde edilen veriler ile en uç ekrandaki verilerin birbiriyle iletişimi güvenli bir biçimde sağlanmıştır. [54]

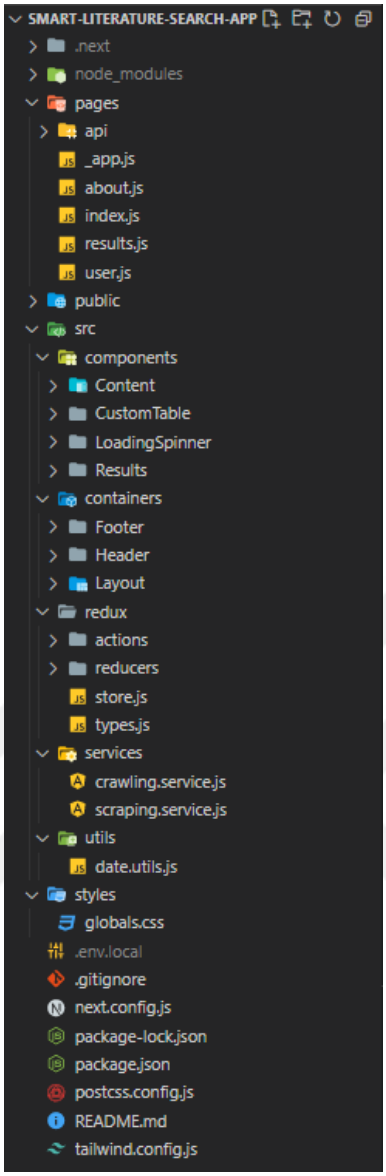
3.4.1.5. Tailwind

Web uygulaması geliştirmelerinde her yeni proje geliştirmesinde tasarım için yeniden CSS yazmak ve geliştirme anlarında yaşanılan sıkıntıları çözmek zordur. Bunun yerine geliştirilmiş CSS kütüphaneleri kullanıldığında hem stil için ekstra çalışma gerekmemekte hem de community'de soruna kolayca çözüm bulunabilmektedir.

React dünyasında Bootstrap [56], Tailwind [57], Material [58] gibi CSS kütüphaneleri popüler olarak kullanılmaktadır. Yapılan çalışmada daha yaygın olarak kullanılan ve yeni geliştirilmiş olan Tailwind kütüphanesi tercih edilmiştir. Web uygulamasında Tailwind kütüphanesi ile tüm ekranlara uyumlu, renkleri ve tasarımları sade sayfalar geliştirilmiştir.

3.4.1.6. Proje iskeleti

Web uygulaması Nextjs Frameworkü kullanılarak React kütüphanesiyle Javascript dilinde geliştirilmiştir. Şekil 47'de frontend uygulamasının proje iskeleti görülebilmektedir. "pages", "public" ve "src" isminde 3 ana klasörden, uygulama dosyalarından ve alt klasörlerden oluşmaktadır.



Şekil 47. Frontend uygulama iskeleti

React geliştirmesi yapılırken üretilen componentler uygulamanın herhangi bir parçasını oluşturabilmektedir. Sayfanın tamamını çizen container componenti olabilirken sadece uygulama logosunu çizen basit bir <img> elementi de olabilmektedir. React'ta sayfalar arası geçiş yaparken <Route> elementleri ile sayfaların yolları belirlenmektedir. Örneğin ana sayfayı çizdiren bir <Home> componentimiz varsa, bu ana sayfaya ulaşmak için *<Route path='/home'><Home/></Route>* şeklinde tanımlama yapılır.

Uygulamada React geliştirme frameworkü olarak kullanılan Nextjs, proje iskeletinde de görülen "pages" isimli ana klasör altına oluşturulmuş herbir component Next derleyicisi tarafından sayfa olarak algılanıp route tanımlaması yapılmasına gerek kalmadan otomatik route edilmiş şekilde çalışmaktadır.

İlgili klasörde de uygulamamızın sayfalarını oluşturan componentler bulunmaktadır. "_app.js" componenti uygulamayı ayağa kaldıran ana componenttir. "index.js" uygulamanın ana sayfası, "about.js" uygulamanın hakkında sayfasını, "results.js" uygulamanın arama sonucu oluşan verilerinin görüntülendiği sonuç sayfasını, "user.js" uygulamadaki kullanıcı ayar sayfasını derleyen componentlerdir.

Public klasörü, ugulamanın her yerinden erişilebilen logo, çeşitli kaynak dosyaları gibi elementlerin bulunduğu klasördür.

Src klasörü, uygulamanın ana kodlarının bulunduğu klasördür.

Containers klasörü, uygulama içerisinde kullanılacak template için üretilmiş parent componentleridir. Kullanıldıığı component içerisinde Header'ı, Footer'ı ve Layout'u çizdirir. Şekil 48'de Layout componentinin yapısı verilmiştir.

src > containers > Layout > index.js > ...

```
1  import React from 'react';
2  import Footer from '../Footer';
3  import Header from '../Header';
4
5  function Layout({ children }) {
6    return (
7      <>
8        <Header />
9        <div className='h-screen max-w-7xl mx-auto'>
10         {children}
11         <Footer />
12       </div>
13     </>
14   );
15 }
16
17 export default Layout;
```

Şekil 48. Layout container componenti

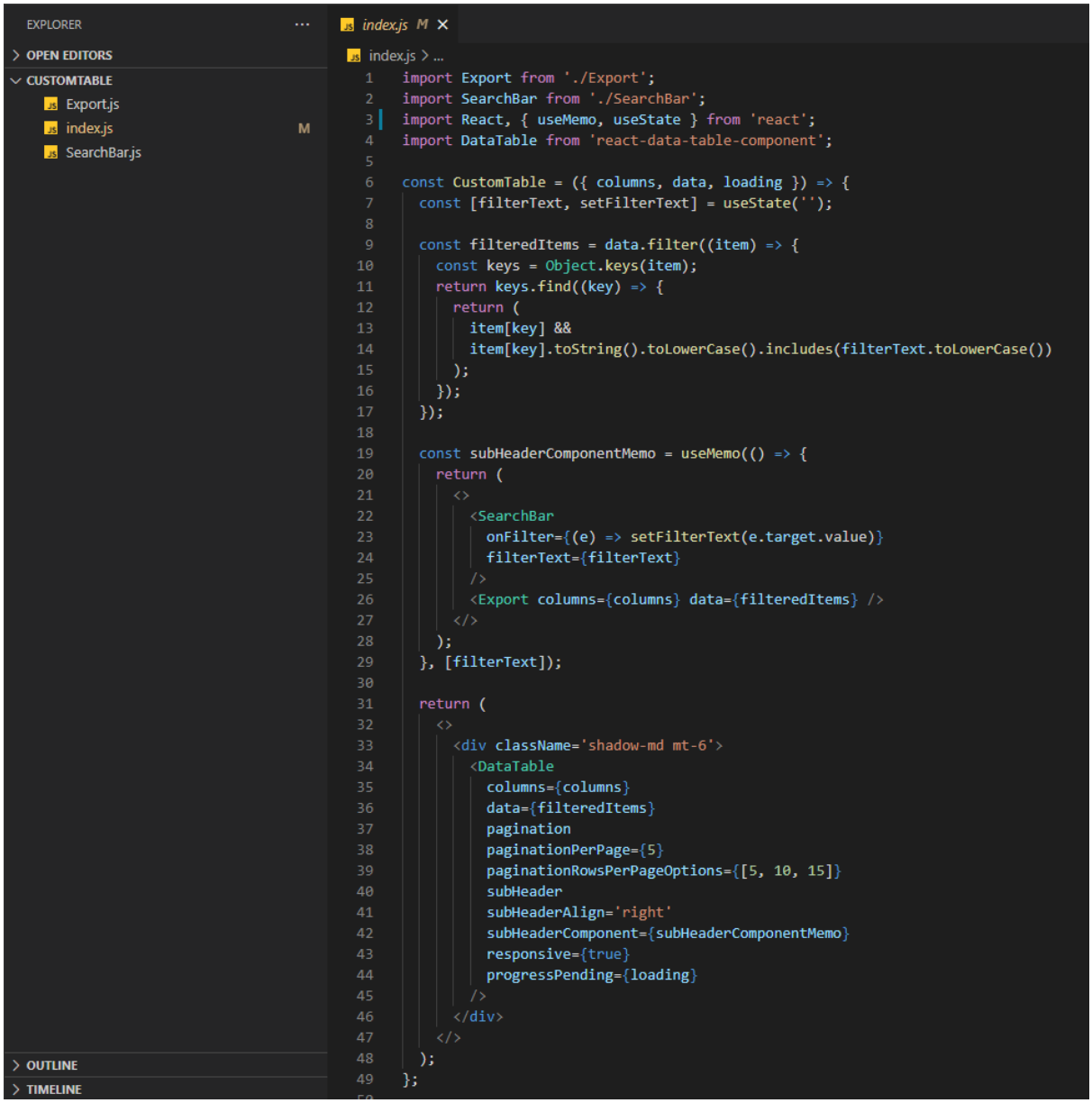
Layout kendi içerisinde "Header" ve "Footer" componentlerini kullanmaktadır. Ek olarak props ile kendisine gönderilen componentleri header ve footer arasında çizdirmektedir. Şekil 49'da Layout componentinin kullanıldığı bir örnek verilmiştir.

pages > index.js > ...

```
import Head from 'next/head';
import Layout from '../src/containers/Layout';
import 'react-toastify/dist/ReactToastify.css';
import Content from '../src/components/Content';
import { ToastContainer } from 'react-toastify';

export default function Home() {
  return (
    <Layout>
      <Head>
        <title>Smart Literature Search</title>
        <meta name='description' content='Generated by create next app' />
        <link rel='icon' href='/favicon.ico' />
      </Head>
      <Content />
      <ToastContainer
        position='bottom-center'
        closeOnClick
        pauseOnHover
        autoClose={4000}
        hideProgressBar
      />
    </Layout>
  );
}
```

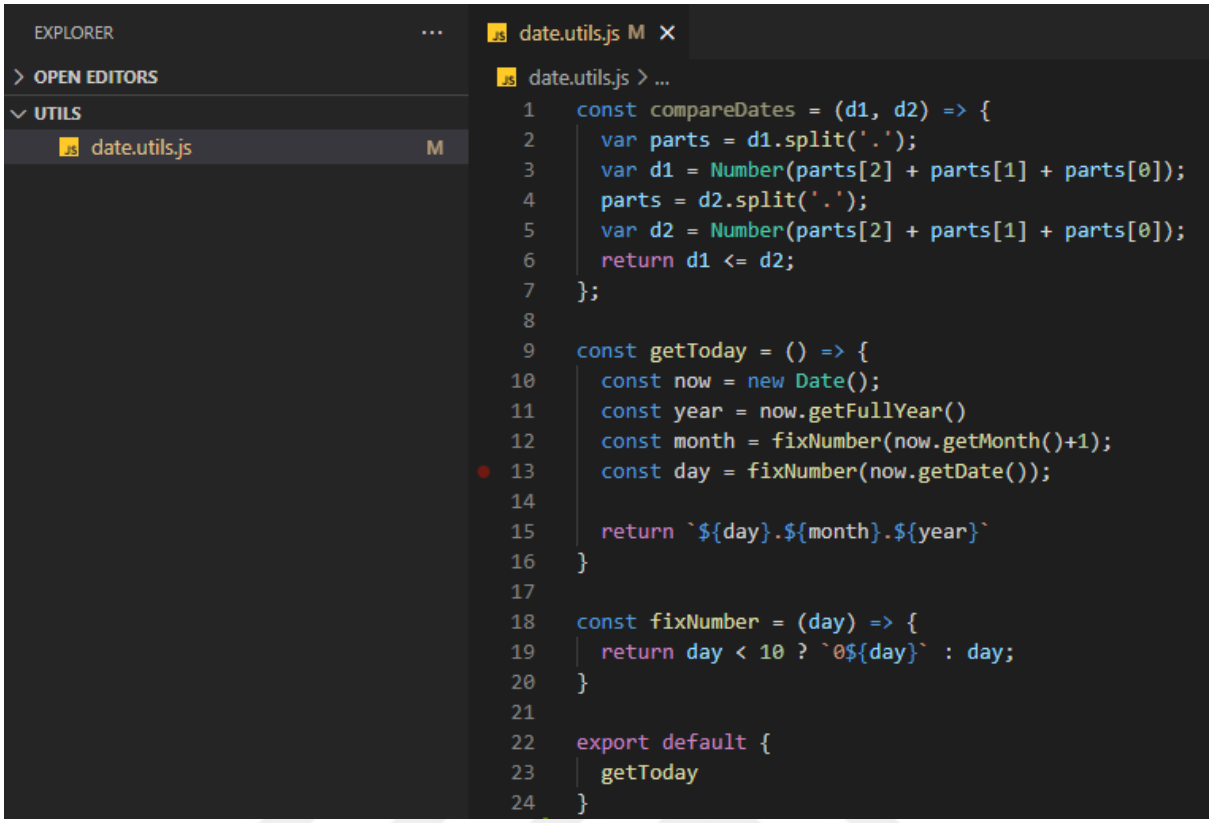
Şekil 49. Layout componentinin kullanım örneği

Anasayfamızı çizdiren "Home" componentinde tüm componentlar "Layout" componentinin içerisine alınarak uygulama temasının bu sayfada da olması sağlanmıştır. Components klasörü, uygulamada bir ya da birden fazla kısımda kullanılan componentlerin bulunduğu klasördür. Uygulama genelinde kullanılan componentler burada toplanmakla beraber, sadece bir component için hizmet veren alt componentlerde bu componentlerin alt dosyası olarak uygulama iskeletinde yer bulmuştur. Şekil 50'de görüldüğü üzere "CustomTable" isminde uygulama geneline hitap eden component kendi içerisinde de tablo içi arama işlemlerinin yapılmasını sağlayan "SearchBar" ve tablo çıktısının alınmasına yarayan "Export" componentlerini barındırmaktadır.



Şekil 50. İç içe component yapısı örneği

Utils klasörü, yazılım geliştirmelerinin çoğunda da bulunan "utilization" sınıflarının bulunduğu utils klasörüdür. Uygulama geneline hitap eden yardımcı methodları içinde barındırır. Uygulama içerisinde tarihsel işlemlere yardımcı olarak "date.utils.js" sınıfı ve içerisinde bulunan kod örnekleri Şekil 51'de gösterilmiştir.



Şekil 51. Utilization dosyası örneği

Redux klasörü, Bölüm 3.4.1.4'te anlatılan uygulama içi veri alışverişini sağlayan state management işlevini üstlenen Redux geliştirmeleri bu klasör altında yapılmıştır.

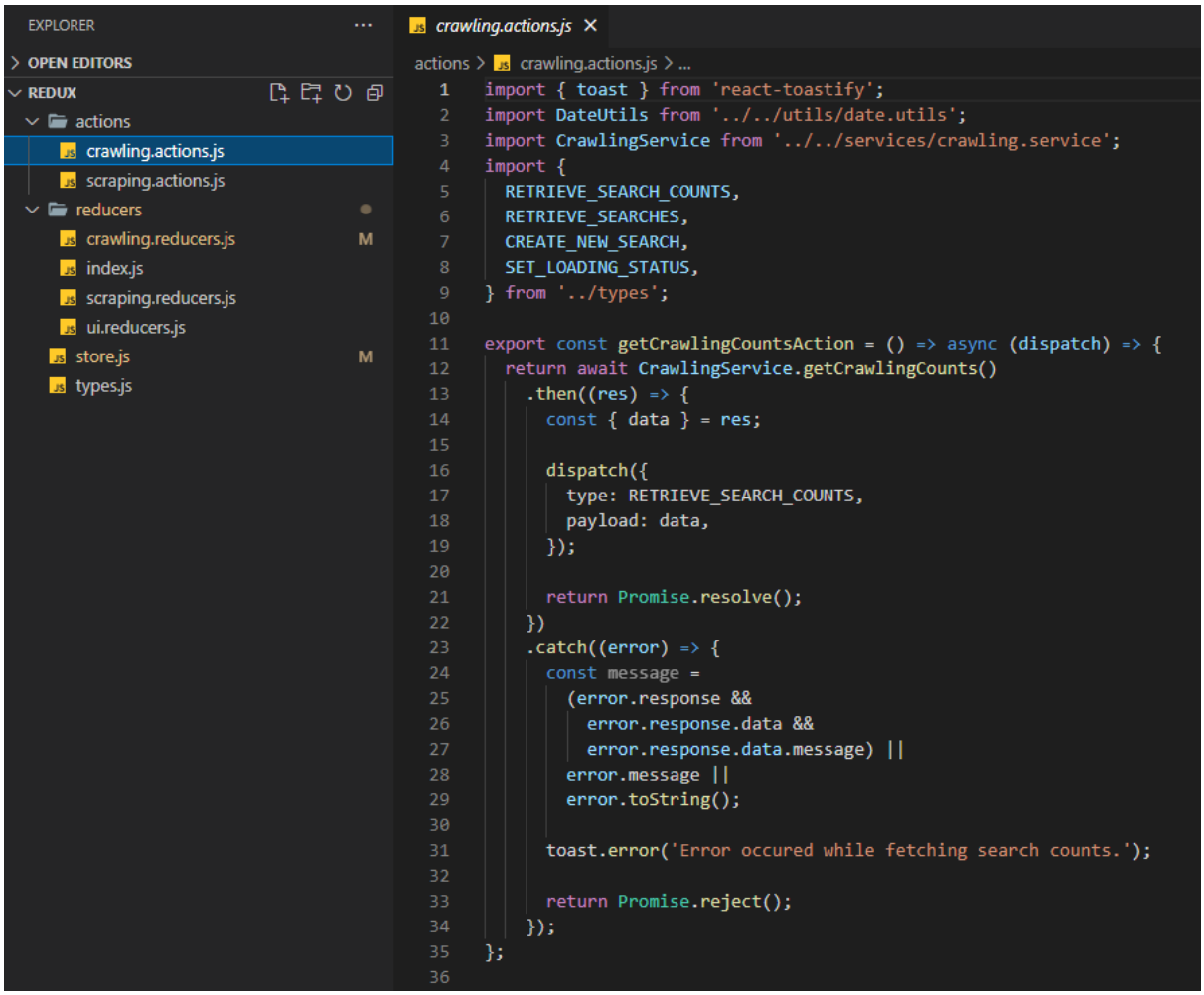
Redux geliştirmesi yaparken verilerin tutulacağı bir "store" oluşturulması gerekmektedir. Şekil 52'de uygulama için store oluşturan kod parçacığı görüntülenmektedir.



Şekil 52. Redux store oluşturulması

Kütüphane içerisinde asenkron veri yazma ve okuma işlemlerinin yapılmasını sağlayan "thunk" kütüphanesi, uygulama dışı veri alışverişi (api ya da servis aracılığıyla erişilen veriler) için "applyMiddleware" methodu, store'un o anki verilerini ve anlık aksiyon takibini tarayıcı üzerinde yapabilmek için de "redux-devtools-extension" kütüphaneleri import edilip store oluşumunda kullanılmıştır.

Actions klasöründe redux döngüsünde “dispatch” denilen redux işlemlerini tetiklemeye ve store’dan alışveriş yapmaya yarayan tetikleyici işlemler bulunmaktadır. Bu dispatch işlemi ile “actions” klasörü altındaki methodlar tetiklenir. Tetikleme işlemi yapılınca işlem sonucu oluşan veriye “payload” yapılan işlemin türüne de “type” denmektedir. Tetiklenen bu aksiyon reducer’a iletilir. Dispatch edilen işlemin type’ına göre reducer’da istenilen güncelleme yapılır. Şekil 53’te Crawling Api ile iletişim kuran “Crawling servisi” kullanılarak redux aksiyon işlemleri gerçekleştiren bir kod parçacığı verilmiştir.

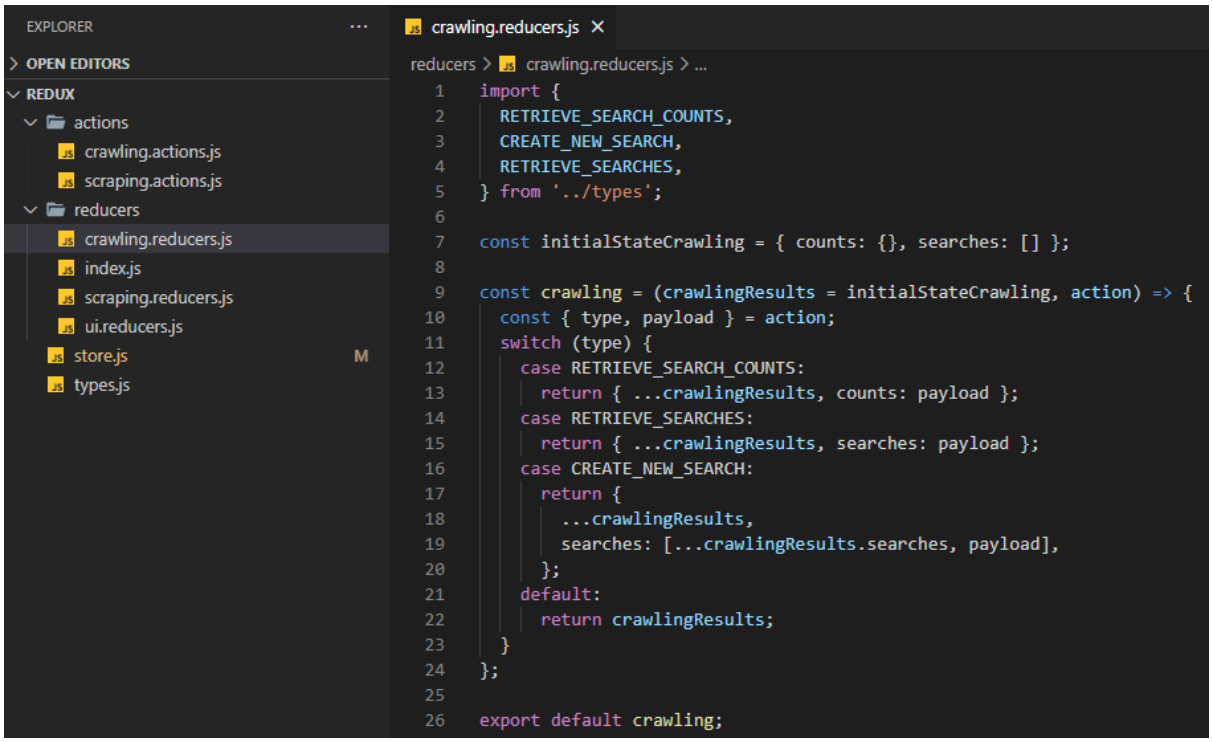


Şekil 53. Redux action örneği

“getCrawlingCountsAction” isimli method Crawling Service’indeki methodu kullanarak API tarafından crawling işlem sayılarını çektikten sonra, “RETRIEVE_SEARCH_COUNTS” type’ında bir action dispatch etmektedir.

Redux’ta store üzerinde tutulan verilere “global state” denmektedir. Yani bölüm 3.4.1.2.5’de bahsedilen state’in aynısı olmakla birlikte uygulamanın her yerinden erişilebilmektedir. Tutulan verilerin yönetimi de “reducer” ile sağlanmaktadır. Verilerin

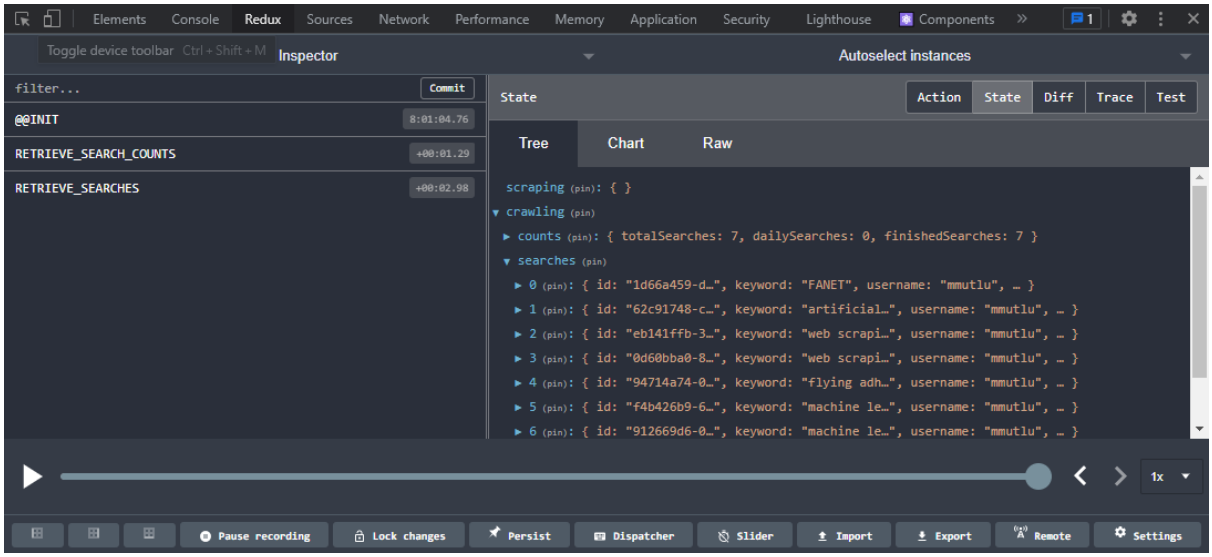
yönetim işlemini de action'dan gelen dispatch işlemine göre yapmaktadır. "reducers" klasörünün altında farklı amaçları karşılayan reducerlar tanımlanmış ve bunlar "rootReducer" isminde bir araya getirilmiştir. Store oluşturulurken rootReducer da parametre olarak verilmiştir. Şekil 53'te Crawling Api ile veri alışverişinden sorumlu "crawling" reducer'ı verilmiştir.



Şekil 54. Redux reducer örneği

Reducer tanımlaması yaparken "initial" denilen ilgili reducer'da tutulan verinin ilk state'i tanımlanır. Burada da bir obje içerisinde "counts" isminde bir boş obje ve "searches" isminde boş bir array tanımlanmıştır. Örnek vermek gerekirse action'dan gelen "RETRIEVE_SEARCH_COUNTS" type'ındaki dispatch ile gelen payload bu crawling reducer'ındaki counts state'ine kaydedilir.

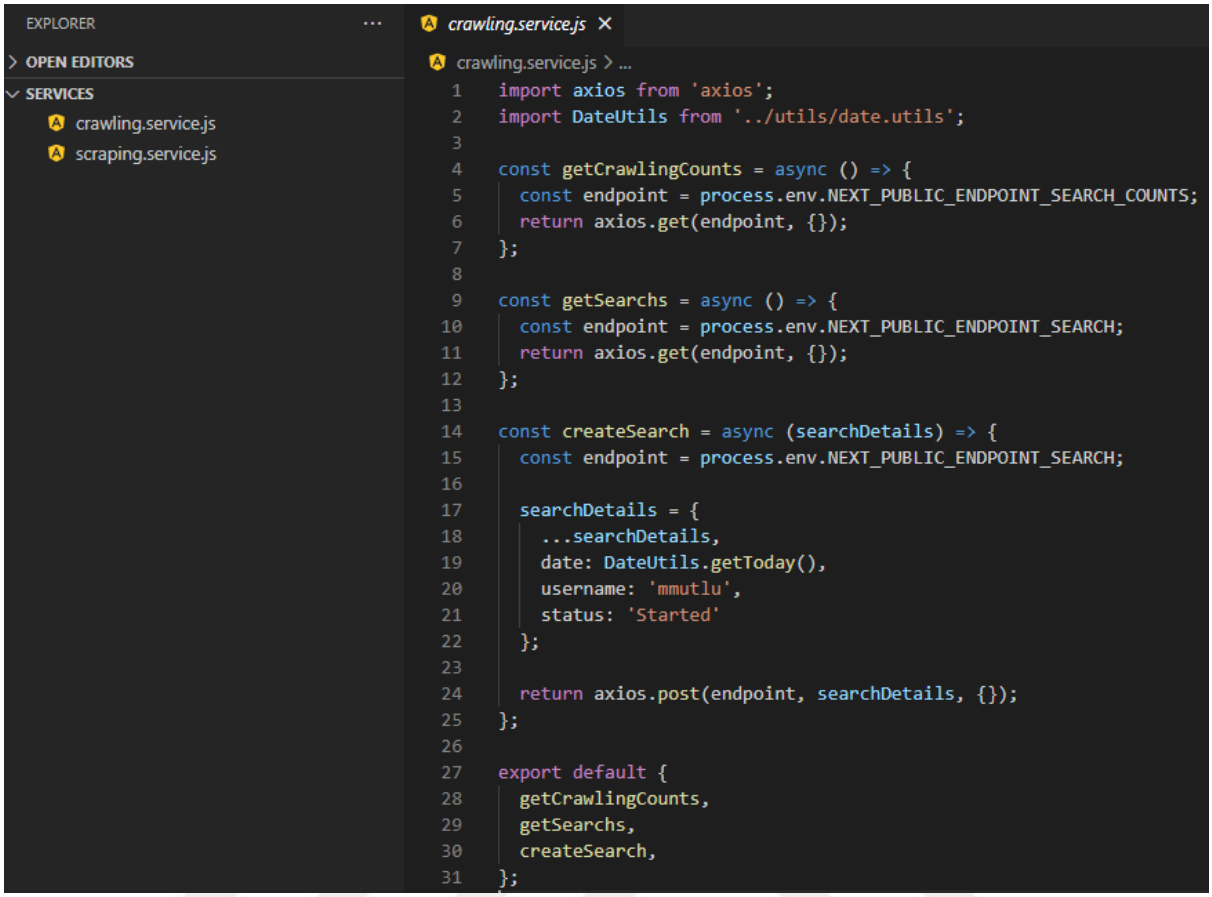
Uygulama içerisinde kullanılan store'un tarayıcı üzerindeki görüntüsü Şekil 55'te verilmiştir.



Şekil 55. Redux store tarayıcı görünümü

Burada görüldüğü üzere ilk olarak "@@INIT" işlemi yani store'un ayağa kaldırılması işlemi gerçekleştirilmiş ardından da iki farklı crawling işlemi gerçekleştirilmiş. Ve bu işlemler sonucunda store'un görüntüsü sağ taraftaki gibi olmuştur.

Services, backend üzerinde çalışmakta olan API'lerle iletişim kurmaya yarayan servislerin geliştirildiği klasördür. Crawler API ile iletişimde olan "Crawling Service" ve Scraper API ile iletişimde olan "Scraping Service" isimlerinde servisler geliştirilmiştir. Şekil 56'da Crawling Service içerisindeki bir kod parçacığı görüntülenmektedir.



Şekil 56. Frontend service örneği

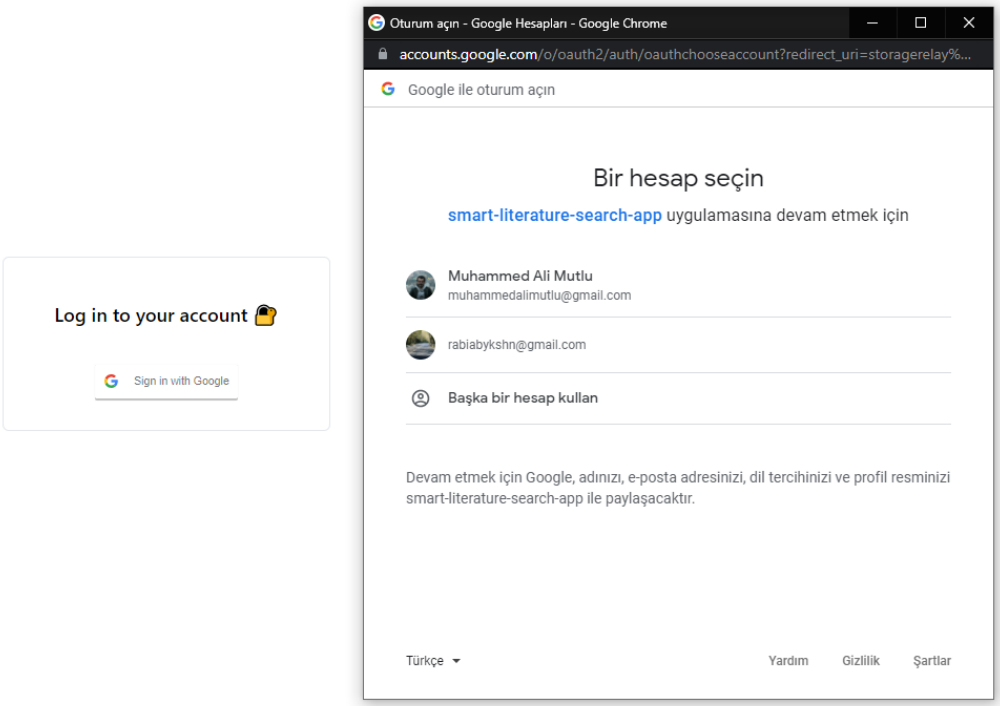
"Axios" kütüphanesi API iletişim kütüphanesidir. Axios kullanılarak API endpointlerine istekler atan farklı methodlar burada geliştirilmiştir. Geliştirilen bu methodlar Redux action'larında kullanılmaktadır.

4. SONUÇLAR VE TARTIŞMA

Bu bölümde geliştirilen uygulamanın kullanım kılavuzuna yer verilmiştir. Uygulama içerisindeki lojik, sonuçların performans analizi gibi konular üzerinde yoğunlaşılmıştır.

4.1. Uygulama Kullanımı

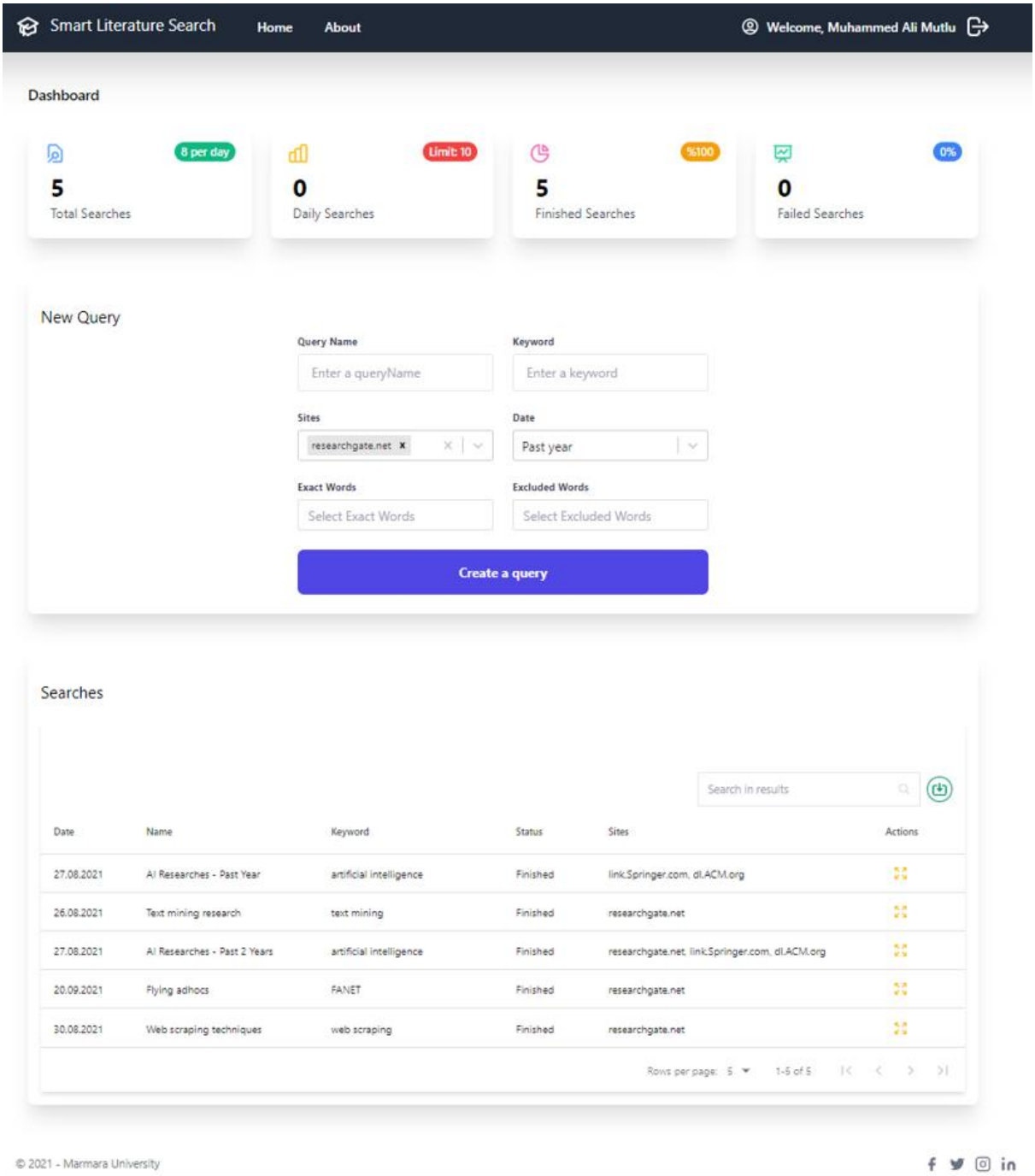
Uygulamada kullanıcının gerçekleştirebileceği işlemler ve işlem sıraları Şekil 66' de gösterilmiştir. Uygulamaya erişim sağlandığında ilk olarak kullanıcıyı karşılayan ekran kullanıcı girişi ekranıdır. Kullanıcı "Sign in with Google Account" butonuna tıkladıktan sonra açılır pencere aracılığı ile Şekil 57' de görüldüğü gibi Google hesabını seçerek giriş yapar.



Şekil 57. Smart Literature Search Giriş Sayfası

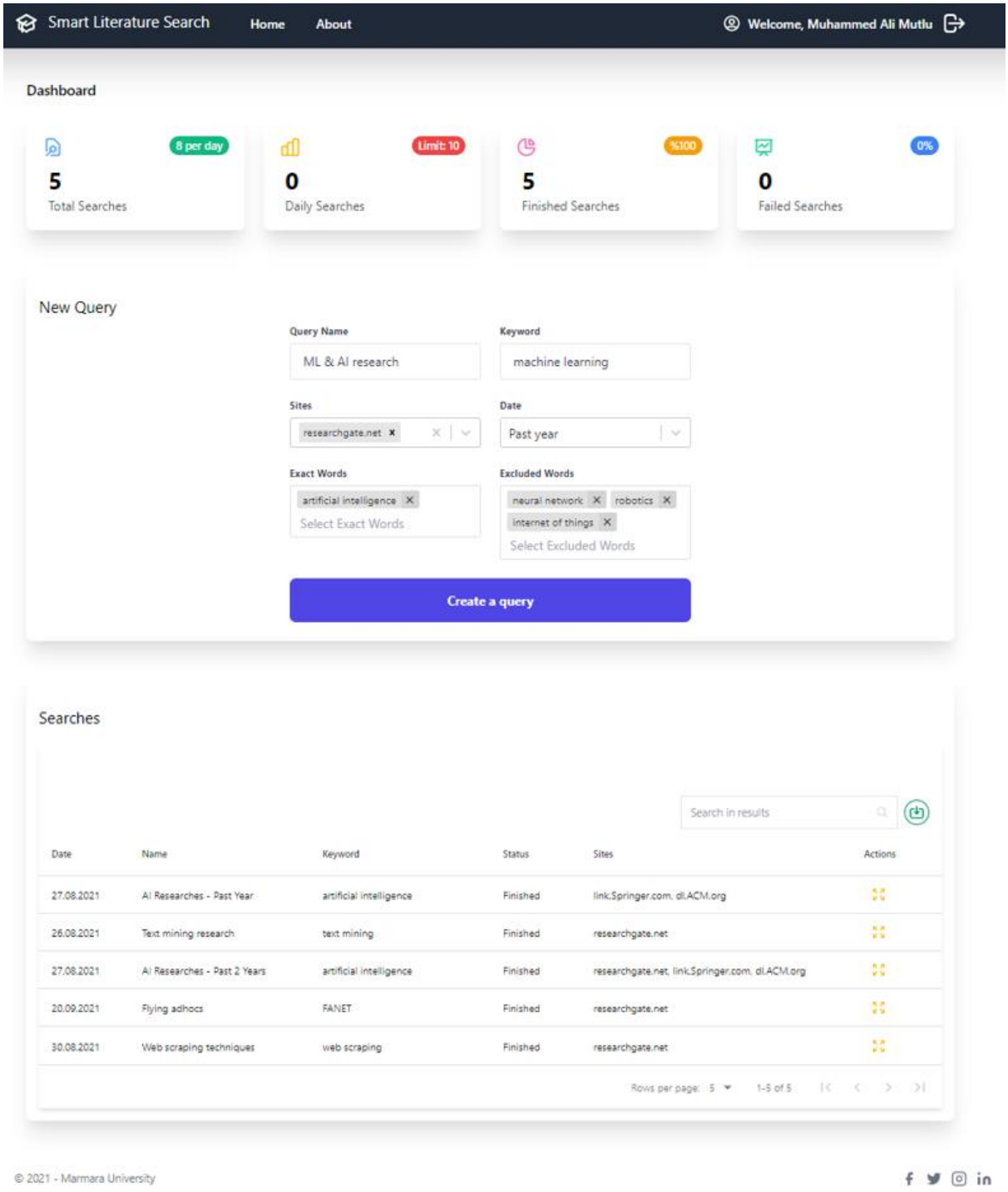
Daha sonra kullanıcı anasayfa ekranına yönlendirilir. En üst tarafta header' da uygulama içerisindeki sayfalara geçiş yapabileceği navigasyon paneli bulunmaktadır. En üstte bulunan KPI panelinde daha önce yaptığı tüm aramaları, o gün yaptığı tüm aramaları, tamamlanan aramaları ve hata almış aramaların sayısını görebilmektedir. Ek olarak gün başına ortalama arama sayısı, o gün için kalan arama limiti, tamamlanmış ve hata almış

aramaların oranlarını görebilmektedir. Bir alt kısımda kullanıcının yeni arama başlatabileceği arama paneli bulunmaktadır. En alt kısımda ise daha önce yaptığı aramaların detayını görebildiği bir tablo bulunmaktadır. Bu tabloyu çıktı olarak alabilmekte ve tablo içerisinde arama yapabilmektedir. Ayrıca sayfalar arası geçiş yapıp diğer aramaları da kontrol edebilmektedir. Anasayfanın görünümü Şekil 58'de verilmiştir.



Şekil 58. Smart Literature Search Anasayfa

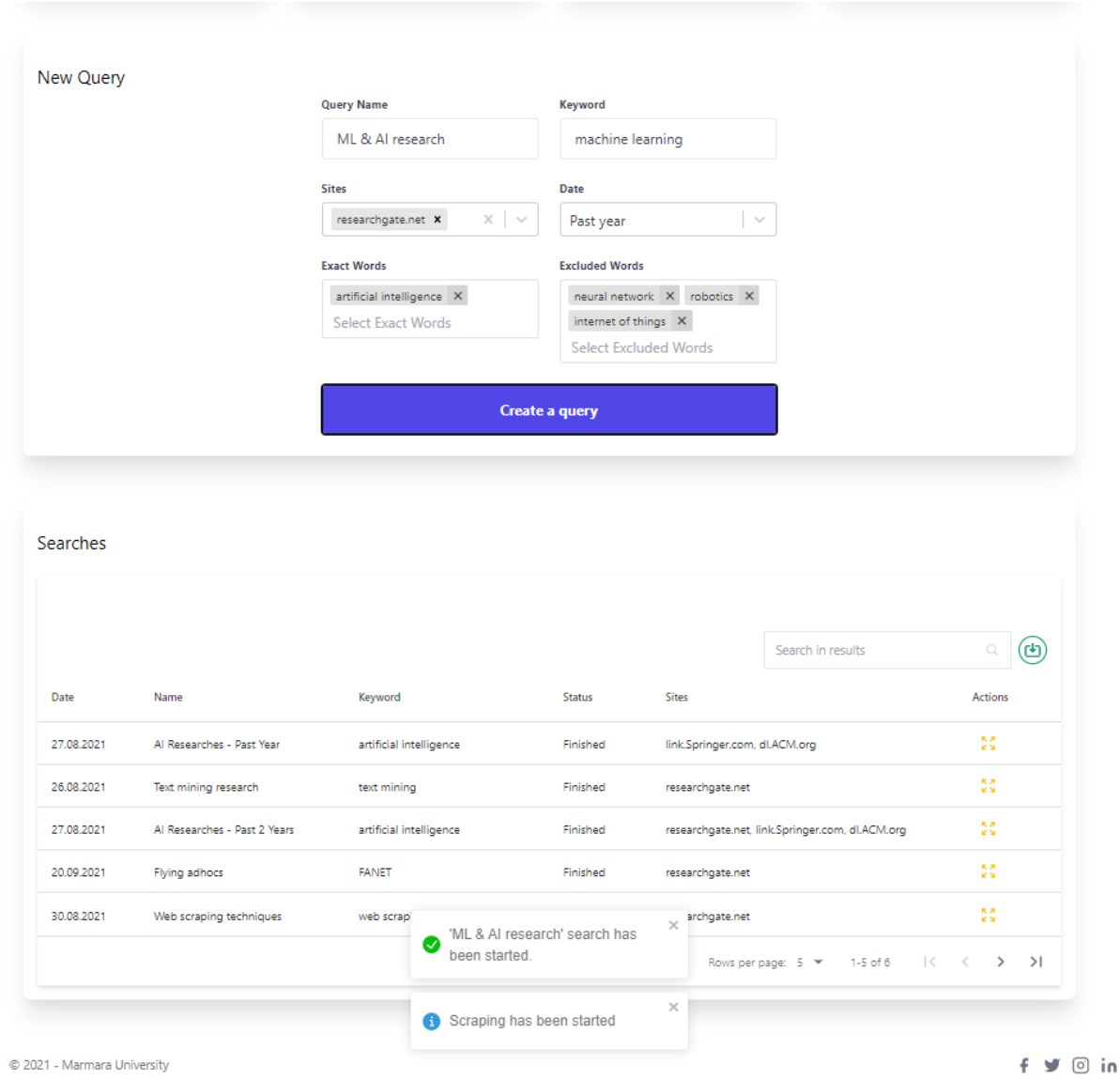
Kullanıcı yeni arama başlatmak istediğinde aramasının ismini ve aratmak istediği anahtar kelimeyi ilgili kısımlara yazar. Daha sonra “Sites” menüsünden arama yapmak istediği akademik site veya siteleri seçer. Aramasını yapmak istediği yıllar için “Date” kısmından tarih filtrelendirmesi yapar. Aramasında arattığı anahtar kelime ile birlikte makalelerde mutlaka bulunmasını istediği anahtar kelimeleri “Exact Words” kısmına, makalelerde bulunmasını istemediği anahtar kelimeleri ise “Excluded Words” kısmına yazar. “Create a query” butonuna basarak aramasını başlatmış olur. Şekil 59’de arama başlatma işlemi görüntülenmiştir.



Şekil 59. Smart Literature Search yeni arama oluşturma

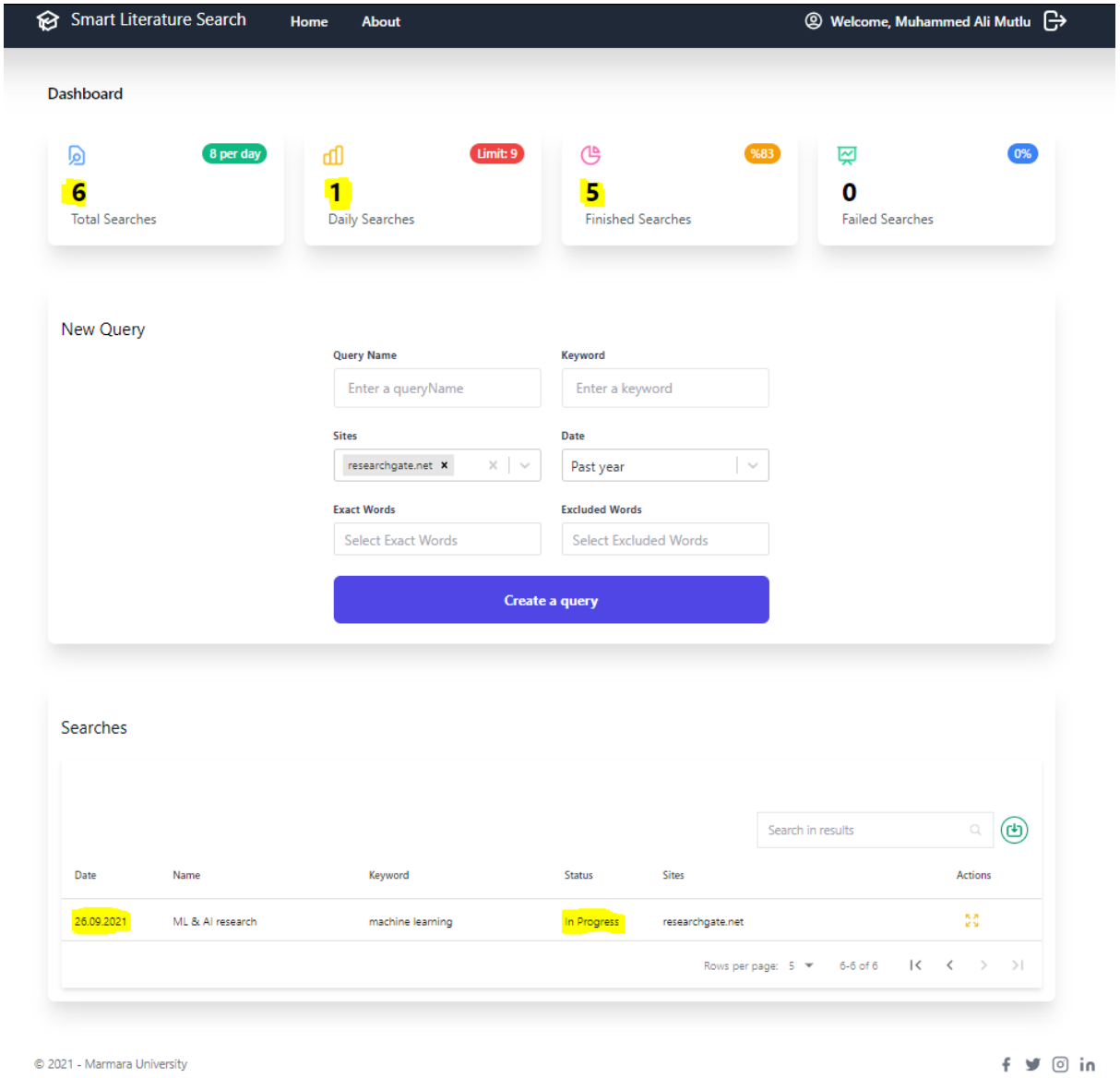
Burada kullanıcı "ML & AI Researches" ismini verdiği aramasında "Researchgate" üzerinden "machine learning" anahtar kelimesi ile makale araması başlatmıştır. Bu araştırmasında exact words' e girdiği değer ile "artificial intelligence" ve "machine learning" anahtar kelimelerinin bir arada bulunduğu makaleleri hedeflemiştir. Ayrıca "internet of things", "neural network" ve "robotics" gibi anahtar kelimelerle ilgilenmediği

için excluded words kısmına filtre olarak eklemiştir. Create butonuna tıkladığında tarama işlemi başlar ve açılır pencereler ile çıkan bilgi mesajlarda görüldüğü gibi scraping işlemi başlamıştır [Şekil 59].



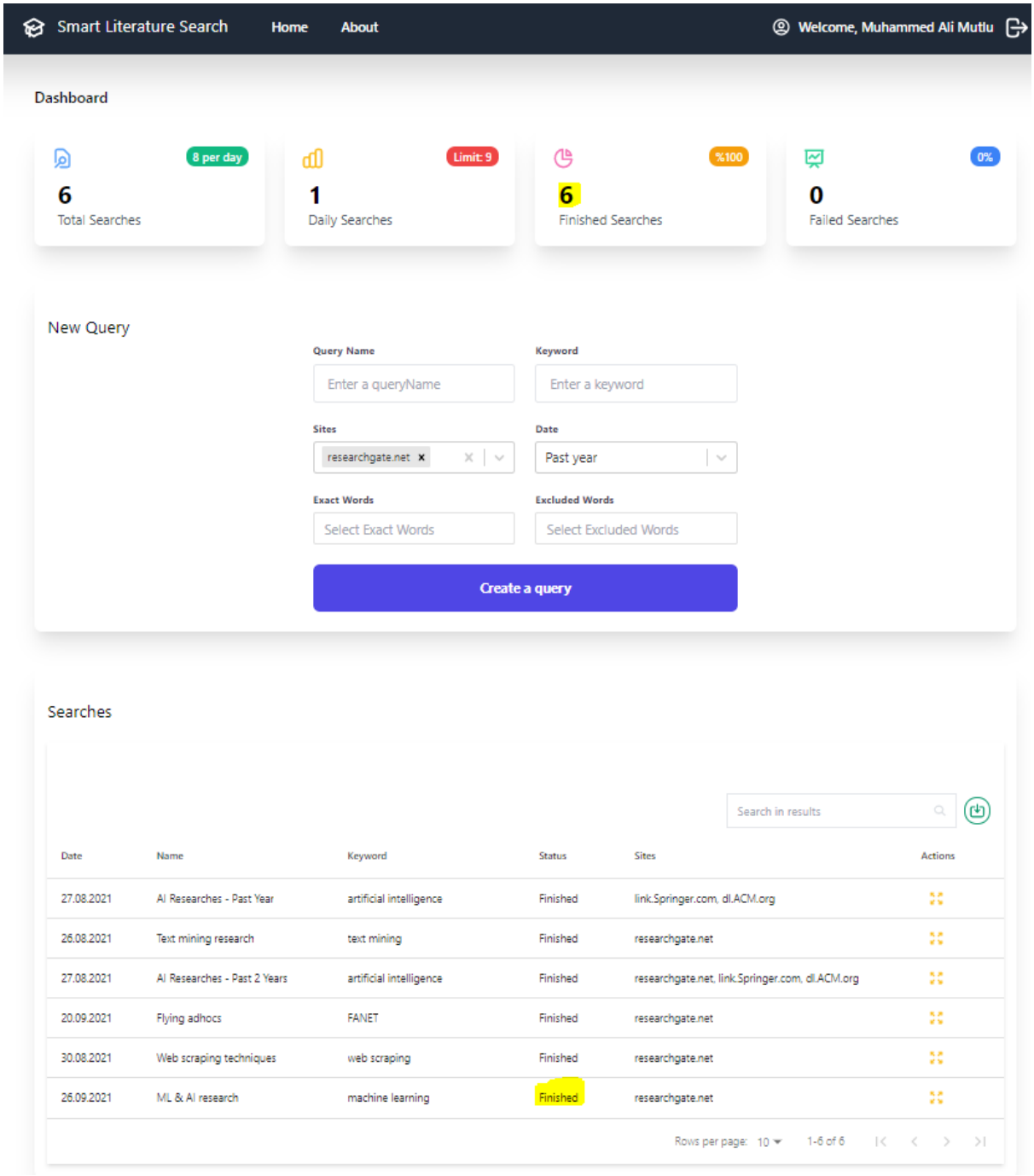
Şekil 60. Smart Literature Search yeni arama başlatma

Şekil 61'de görüldüğü gibi yeni arama başlattıktan sonra anasayfadaki tabloda değerler güncellenir. Yapılan aramanın son durumu tablodaki "Status" kısmından görüntülenmektedir. Yeni başlatılan arama "Started" statüsünde yani yeni başlamıştır. Arama başlatıldıktan birkaç saniye sonra sayfa yenilendiğinde güncel değerlerin geldiği görülmektedir. Arama statüsü "Started" dan "In Progress" a geçmiştir. Yani bu durum aramanın arkaplanda hala devam ettiğini ifade etmektedir. Ayrıca KPI panelinde de değerler yeni başlayan aramayla birlikte değişmiştir.



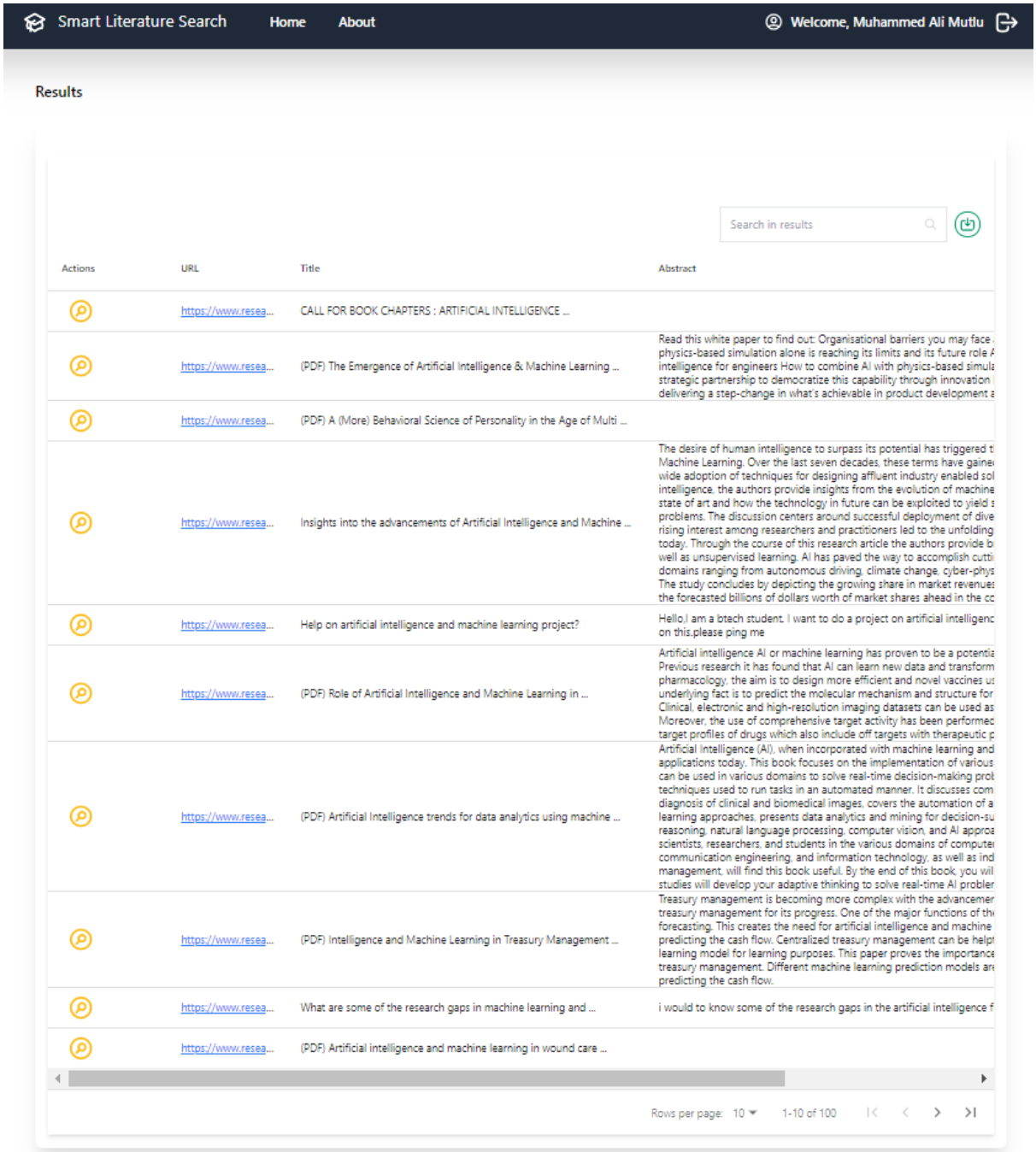
Şekil 61. Smart Literature Search devam eden arama

Ortalama bir aramanın tarama ve kazıma işlemlerinin tamamlanması iki ya da üç dakika sürmektedir. Bu süre zarfında uygulama servisleri hala ayakta olmakla birlikte yeni arama başlatmak için diğerini beklemek gerekmemektedir. İstenirse paralelde yeni aramalar başlatılabilir. Süre sonunda sayfa yenilendiğinde Şekil 62' deki gibi görüntü oluşmaktadır. Aramanın statüsü bittiğini ifade eden "Finished" e geçmiştir. Ve KPI panelindeki değerler güncellenmiştir.

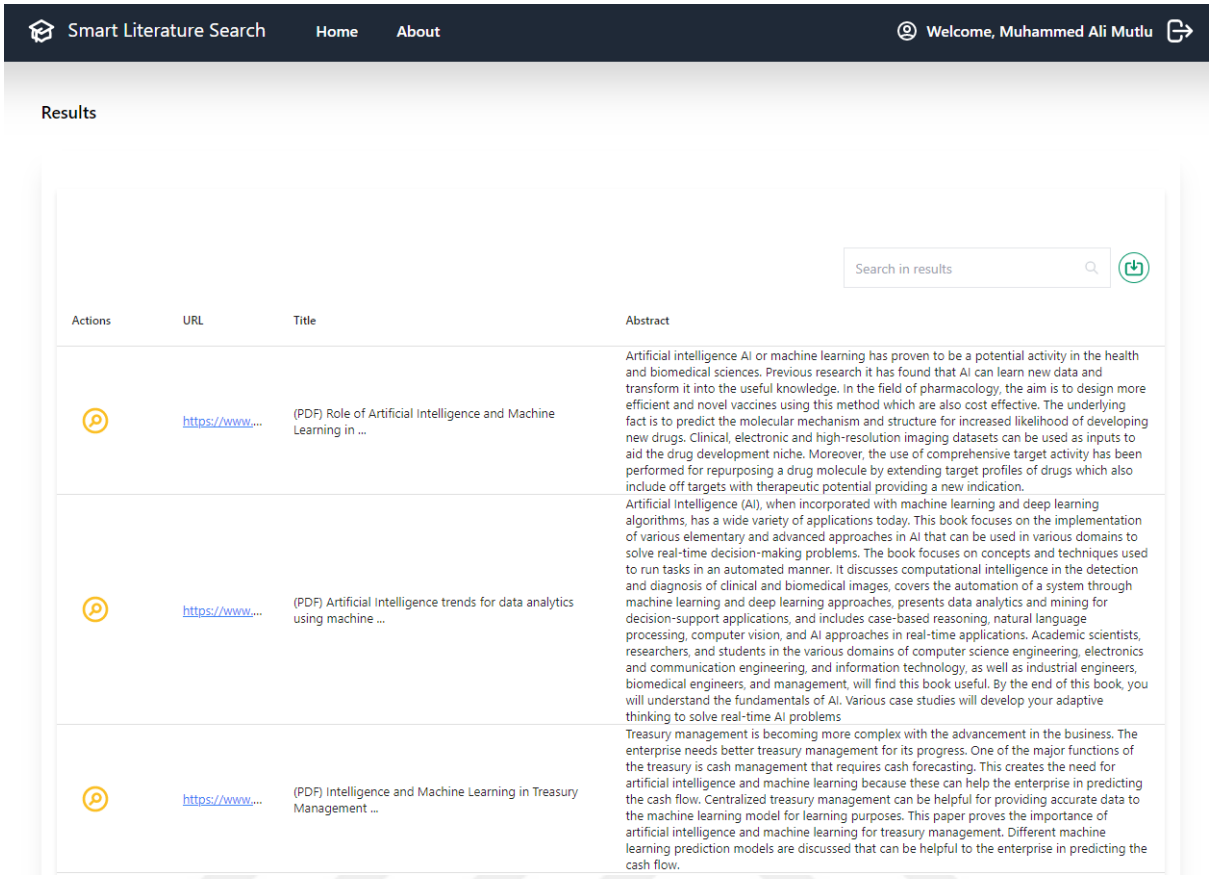


Şekil 62. Smart Literature Search tamamlanmış arama

Kullanıcı aramanın yanındaki "Actions" sütunundaki "expand" butonuna tıkladığında o aramayla alakalı sonuçların olduğu sayfaya yönlendirilmektedir. Şekil 63 ve 64'te yönlendirilmiş sayfanın tam ekran ve yakınlaştırılmış görüntüsü bulunmaktadır.

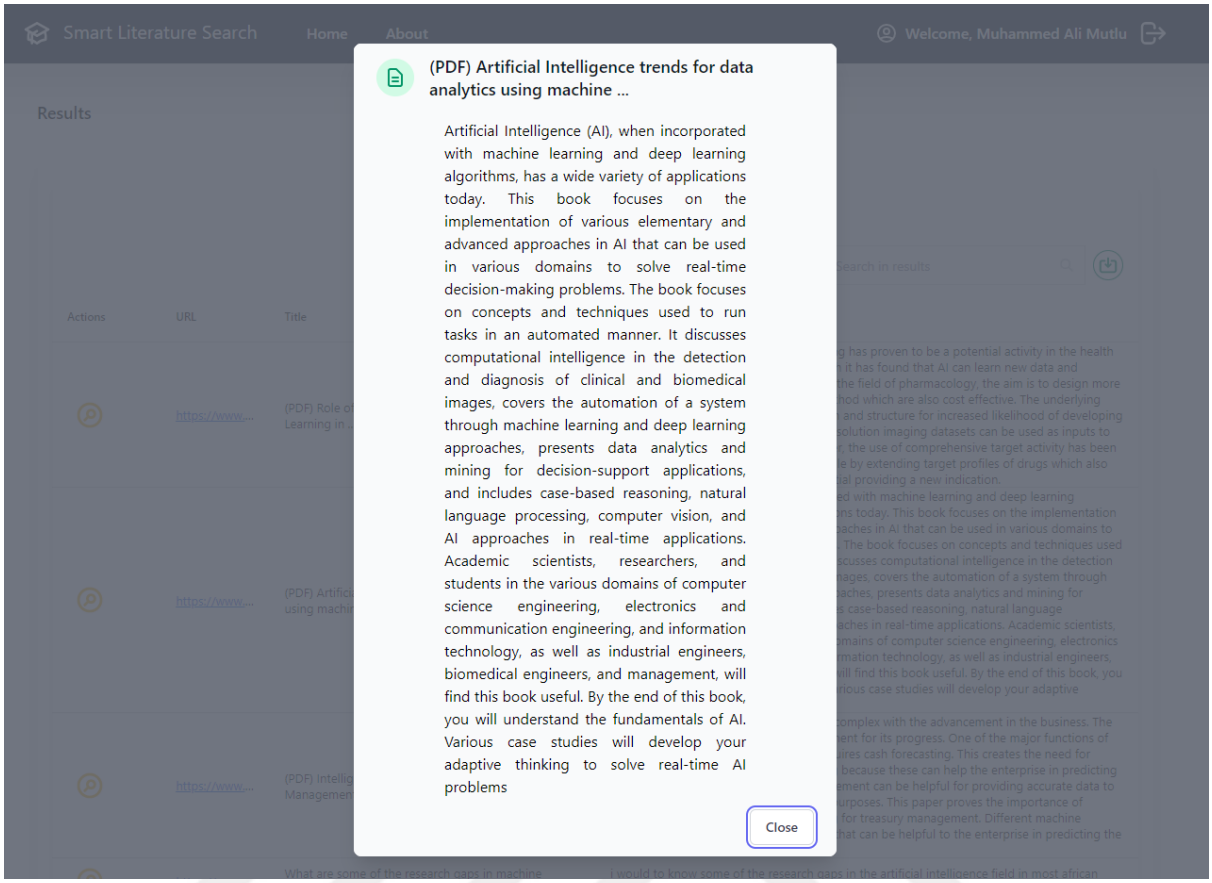


Şekil 63. Smart Literature Search arama sonuçları - I



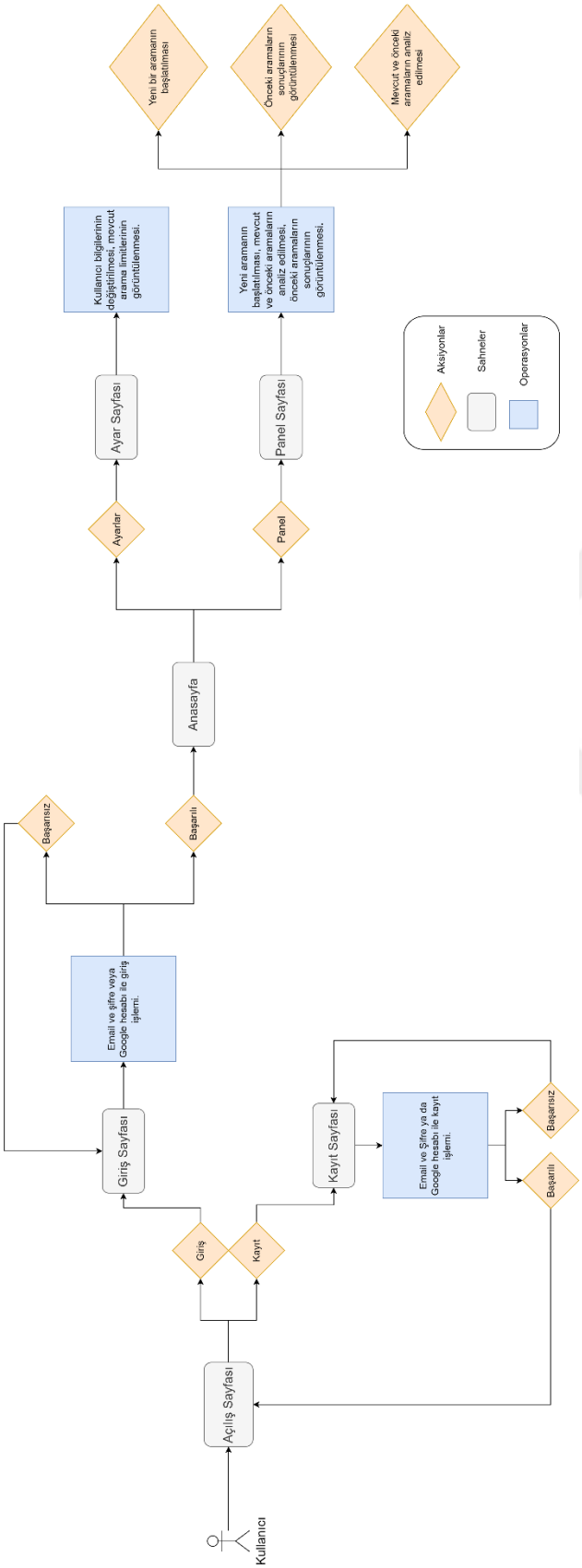
Şekil 64. Smart Literature Search arama sonuçları – II

Sonuç sayfası ilgili arama sayfası için açıldığında kullanıcıyı arama sonuçlarını içeren bir tablo beklemektedir. Tabloda yapılan aramada bulunan makalenin URL-Title eşleşmesi ve bu makalenin kazınmış hali bulunmaktadır. Kullanıcı tablodaki sayfaları gezerek makale özetlerini inceleyebilmekte, arama yapabilmekte ve çıktı alabilmektedir. Kullanıcı makale ilgisini çektiği takdirde URL sütunundaki değere tıklayarak makalenin ana kaynağına gidebilmektedir. İlgili makale özelinde özeti daha yakın detaylı incelemek isterse tabloda karşısında bulunan büyüteç ikonuna bastığı takdirde Şekil 65'teki gibi makale detaylarının bulunduğu bir popup açılmaktadır.



Şekil 65. Smart Literature Search makale detayı

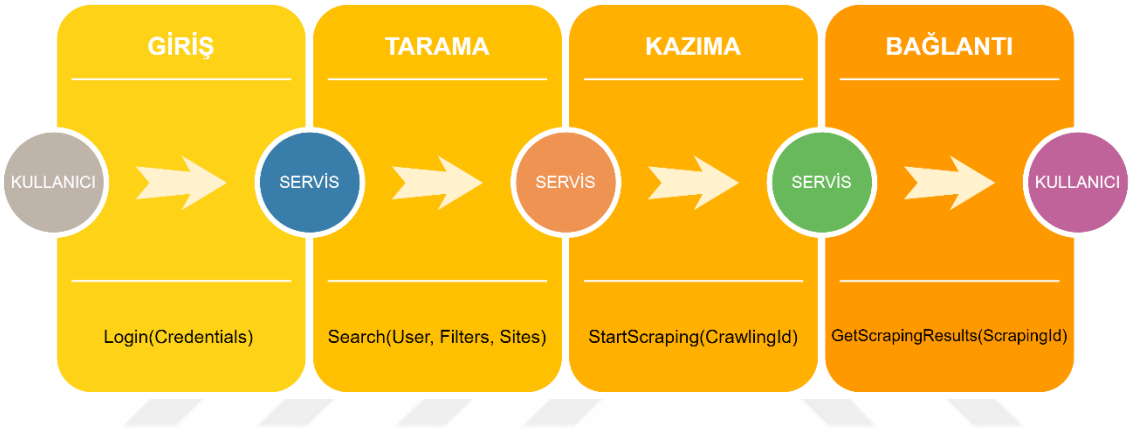
Daha önce başlattığı sonlanan ya da taraması devam eden aramalarını dashboard ekranından görebilmektedir. Taramanın güncel durumunu, hangi filtrelerle ne zaman aratıldığı gibi bilgileri bu sayfada görebilir. Tamamlanmış aramaların sonucunda elde edilen makale özetleri verisini yine bu sayfadan ulaşıp dilediği takdirde export alabilir. Kullanıcı isterse anasayfaya dönüş yaparak sıfırdan filtre menüsünden filtrelerini seçip anahtar kelimesini yazdıktan sonra arama yapılmasını istediği akademik web sitelerini seçtikten sonra yeni bir arama başlatabilir.



Şekil 66. Önyüz uygulama akışı

4.2. Uygulama Akışı

Kullanıcı etkileşimleriyle kullanılan uygulamada yeni bir arama başlatılmak istendiğinde süreç Şekil 67'deki gibi ilerlemektedir. Kullanıcı sisteme giriş yaptıktan sonra filtrelerini, akademik web sitelerini seçip istediği anahtar kelime ile arama başlatır. Arkayüz servisinde tarama operasyonları işletildikten sonra sonuçlar veritabanına yazılır. Sonraki aşamada yapılmış aramanın "crawling id"si ile arkayüz servisinde kazıma operasyonları işletilir. Elde edilen sonuçlar veritabanına yazılır. Son kısımda ise önyüz-arkayüzbağlantısı ile elde edilen sonuçlar kullanıcıya sunulur.



Şekil 67. Uygulama akış grafiği

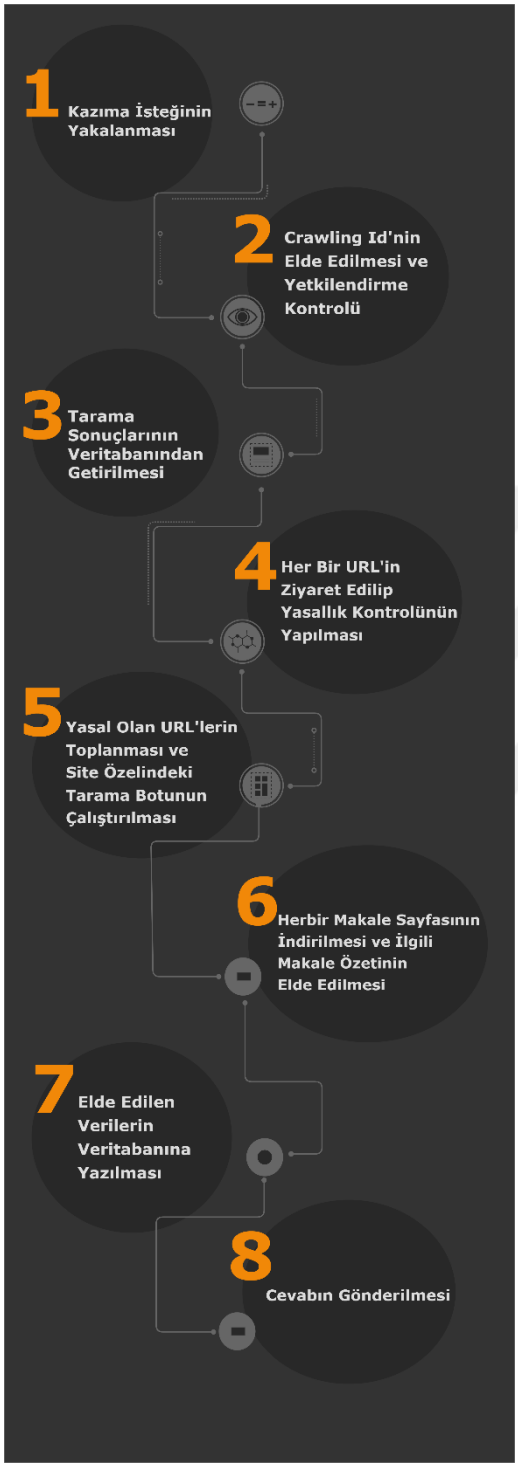
4.2.1. Tarama akışı



Şekil 68. Tarama (crawling) akışı

Kullanıcı sistemden istediği filtreleri ve akademik web sitelerini belirledikten sonra aramasını başlatır. Servis tarafında Crawler API' ye yollanan bir istek ile de tarama işlemleri başlamış olur. Tarama işlemlerinin akışı Şekil 68' de gösterilmiştir. Tarama aşamasında servisimiz ilk olarak kullanıcıdan gelen isteği yakalar ve kendisine gelen istek (request) şemasına göre filtrelerini, arama yapılacak web sitelerini ayıklar. Daha sonra Google ile oluşturulmuş özel arama motorumuza bu filtrelerle birlikte çağrı yapar. Google'ın döndüğü büyük ve ihtiyacımız olmayan verilerin de bulunduğu cevaplara erişilerek ihtiyacımız olan web sitelerinin URL' i ve başlığı ayıklanır. Elde edilen bu nihai sonuçlar paralel bir şekilde kullanıcı bilgileriyle birlikte Cloud Firestore üzerindeki veritabanımıza yazılır. İşlem tamamlandıktan sonra önyüze "crawling id" içeren başarılı cevabı (response) iletilir.

4.2.2. Kazıma akışı

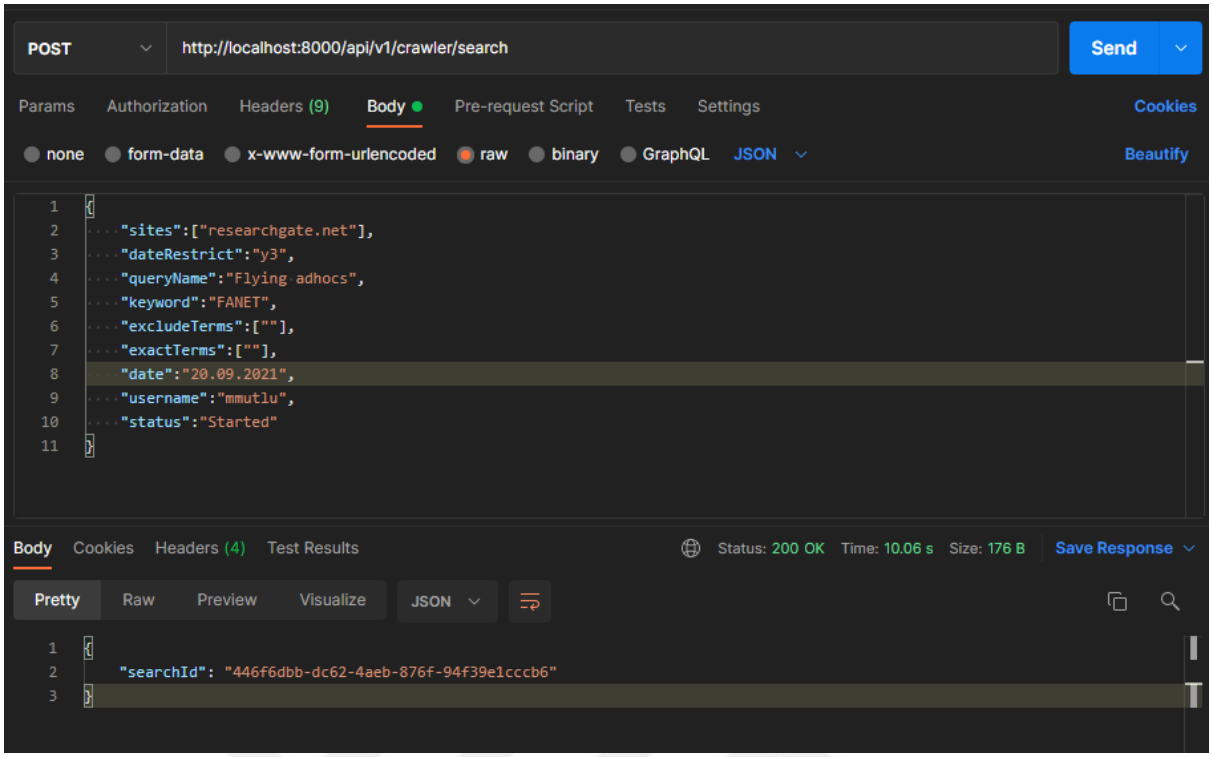


Şekil 69. Kazıma (scraping) akışı

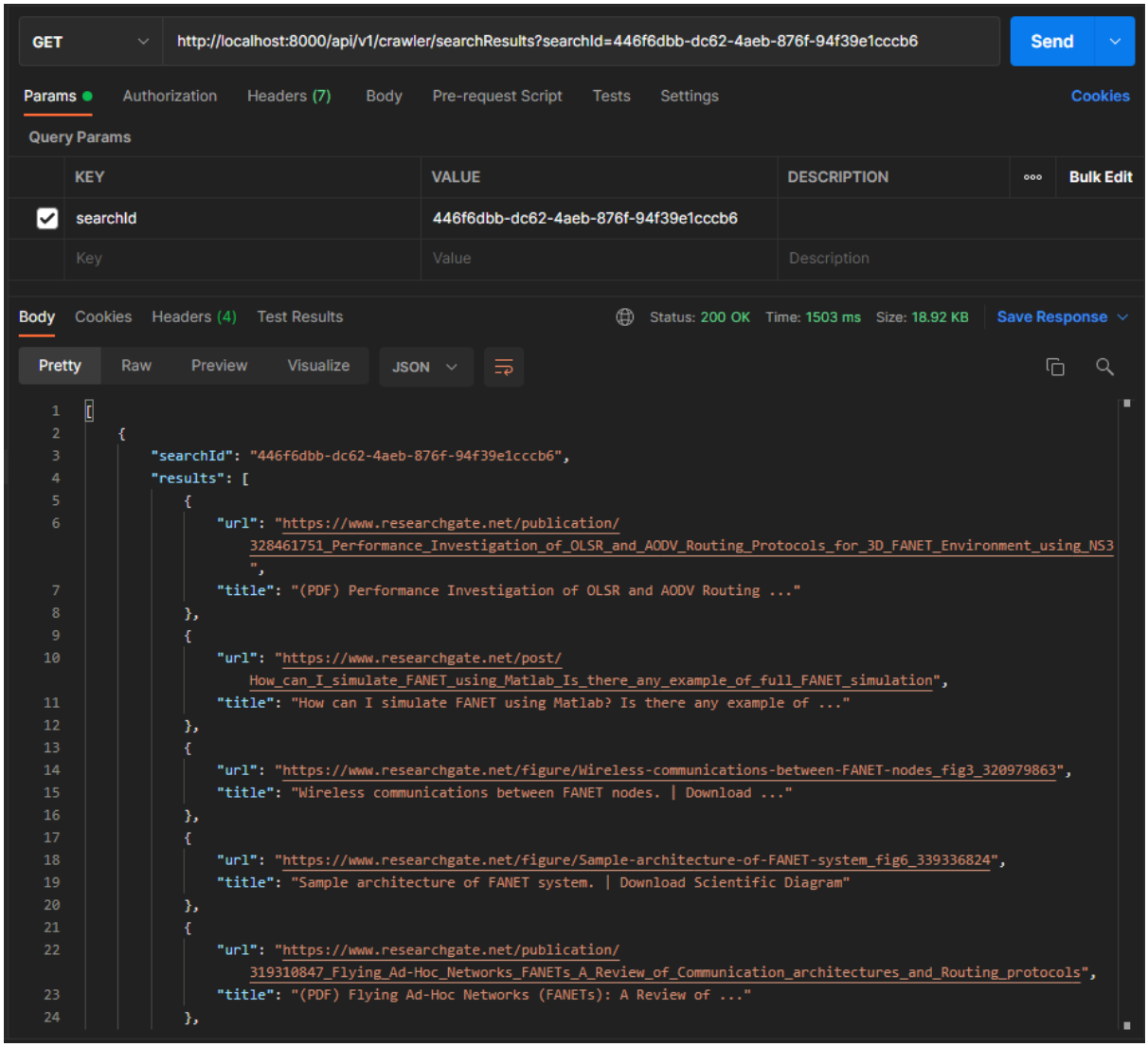
Çalışmada yapılan scraping işlemlerinin gerçekleştirilme akışı Şekil 69' da sunulmuştur. Öncelikle önyüze iletilen başarılı mesajından sonra atılmış "crawling id" yi içeren scraping isteği (request) servis tarafındaki Scraper API' miz tarafından yakalanır. Yakalanan istek içerisinden "crawling id" elde edilir ve bu taramaya ait kullanıcı (user) ile istek atan kullanıcı karşılaştırılır. Eğer kullanıcıya ait bir crawling id ise veritabanından bu taramayla alakalı web site URL' leri çekilir. Her bir URL gezilerek robots.txt kontrolü yapılarak yasallık (legality) durumu kontrol edilir. Yasal (legal) URL'ler elde edildikten sonra bu URL'lerdeki web sayfalarına ait kazıma botları çalıştırılır. Her bir yasal makale sayfası gezilerek uygun algoritmalarla özetleri elde edilir. Elde edilen özetler Cloud Firestore veritabanımıza yazılır. İşlem tamamlandıktan sonra önyüze "scraping id" içeren başarılı cevabı iletilir.

4.3. Çıktılar

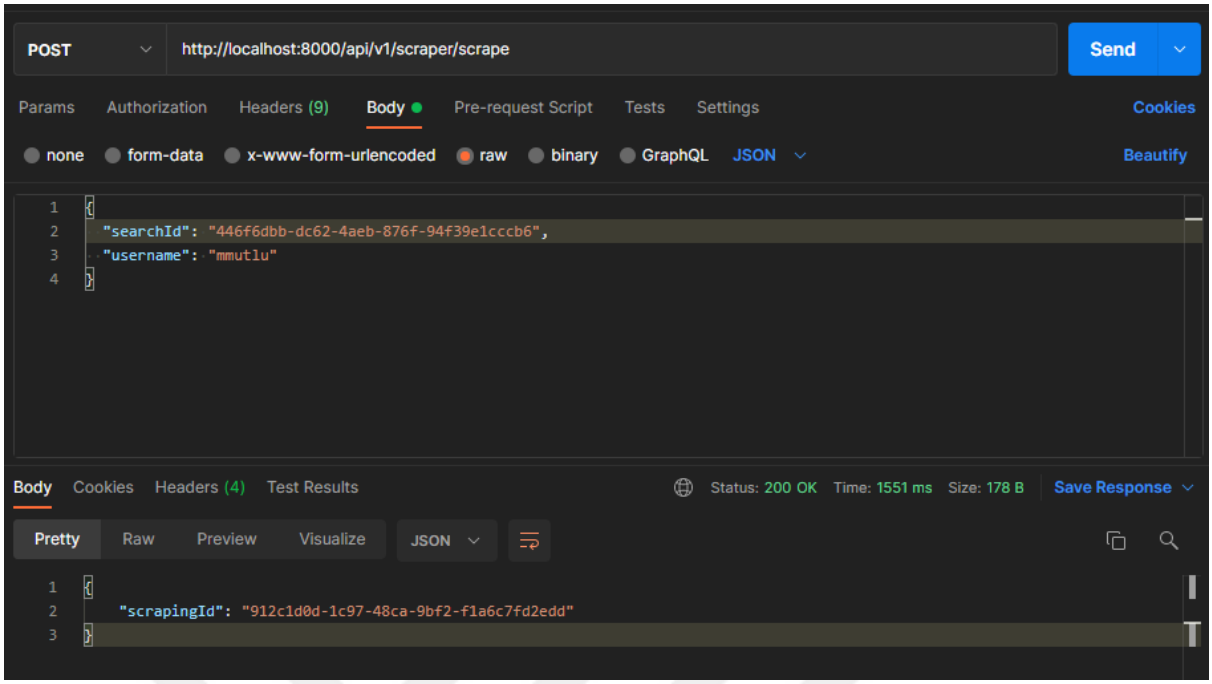
Geliştirilen “Smart Literature Search” uygulamasında “flying adhocs” anahtar kelimesi ile Postman üzerinden testler yapılmıştır. Arama “Researchgate” akademik sayfası ile sınırlandırılmıştır. Bunun yanı sıra son 3 yılın verileri istendiği için “y3” tarih sınırlandırılması yapılmıştır. 10 saniye sonucunda Crawler API arama işlemlerini gerçekleştirip bulduğu url’ leri veritabanına kaydetmiştir [Şekil 70]. Kaydedilen veriler incelendiğinde “IEEE” akademik web sayfasında ilgili anahtar kelime ile ilgili 39 içerik, “Researchgate” akademik web sayfasında 61 içerik olmak üzere toplamda 100 adet sonuç elde edildiği görülmüştür. Elde edilen sonuçlar Şekil 71’ de sunulmuştur. Daha sonra ilgili search id ile Scraper API kullanılarak kazıma işlemleri için istek atılmıştır. 4 dakika 18 saniye sonucunda kazıma işlemleri tamamlanmış ve elde edilen makaleler veritabanına kaydedilmiştir. Kaydedilen makaleler Şekil 72’ de gösterilmiştir. Şekil 73’te görüldüğü üzere yapılan kazıma işlemlerinin sonucunda ise 100 adet url içerisinden akademik makale olan 84 url’in kazınabildiği, makale özetlerinin veritabanına kaydedilmiştir. Aratılan anahtar kelime için tarama ve kazıma işlemlerinin tamamlanması için geçen toplam süre yaklaşık 4,5 dakikadır. Bu geçen süre uygulamada kullanılan kuyruk yapısı ile kullanıcıya hissettirilmeden arkaplanda yapılmakta ve işlemler sonlandığında dashboard üzerinde ilgili arama için status alanından takip edilebilmektedir. İlk olarak “Started” durumu ile başlar, eğer başka işlemler de varsa “Queued” durumuna alınır, yoksa “In Progress” sürecinde bitimi beklendikten sonra “Finished” durumuna alınır. Bu aşamadan sonra kullanıcı sonuçları görebilmektedir.



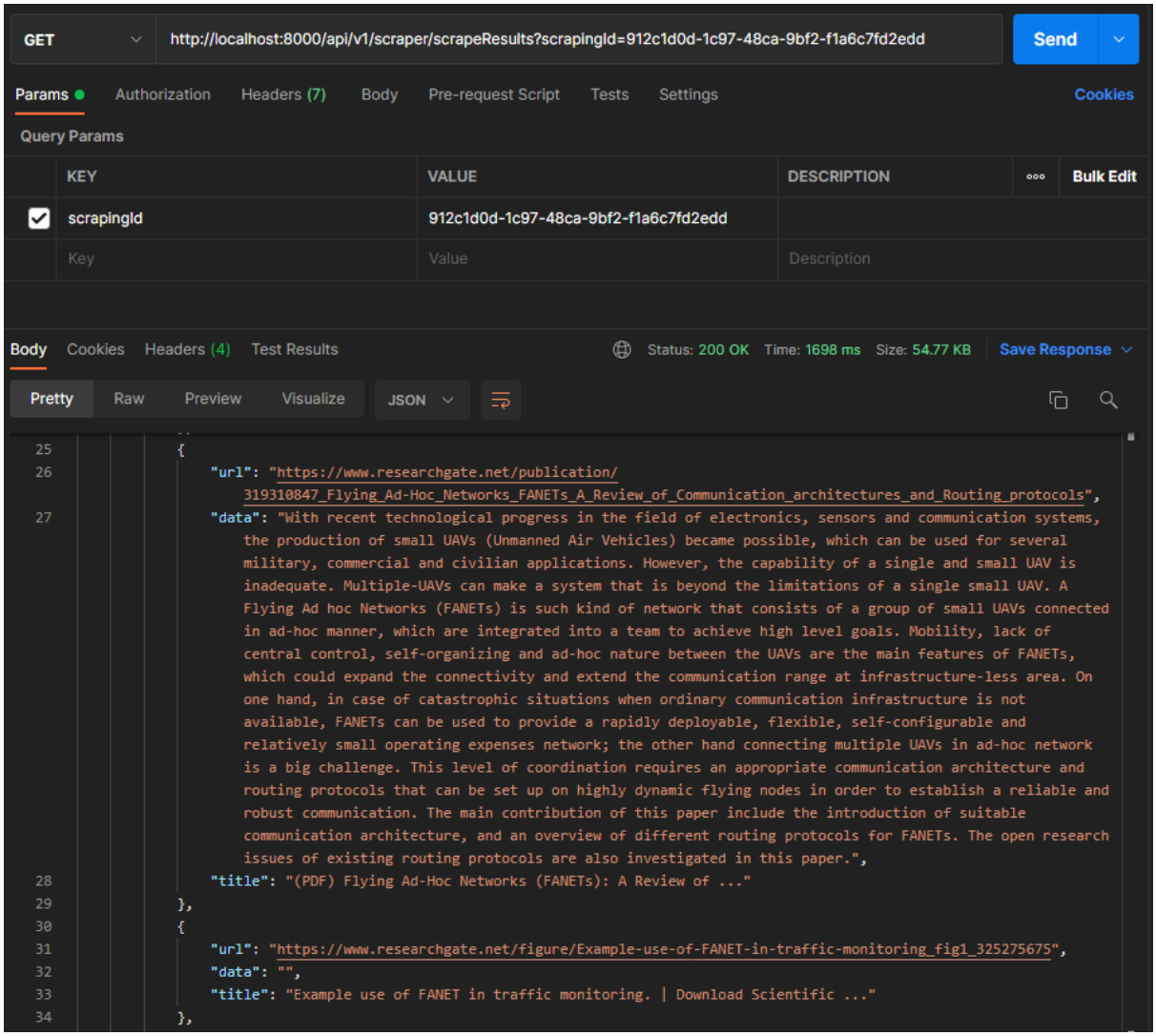
Şekil 70. Crawler API, tarama



Şekil 71. Crawler API, tarama sonuçları



Şekil 72. Scraper API, kazıma



Şekil 73. Scraper API, kazıma sonuçları

5. BULGULAR

Günümüzde akademik çalışmaların en önemli ve ilk aşaması olan literatür araştırmaları aynı zamanda araştırmaların yol haritasını belirleyen bir süreçtir. Literatür taraması, yapılacak çalışmalar ile ilgili mevcut gerçekleştirilmiş çalışmalara ulaşmamızı ve böylelikle literatüre katkı sağlanabilecek alanları belirlememizi sağlayan bir süreçtir. Bu çalışmada, literatür taramasının web kazıma yöntemlerini kullanarak kolaylaştırılmasını ve hızlandırılmasını sağlayacak bir sistem geliştirilmiştir. Yaygın olarak kullanılan ve birçok çalışmaya erişmemizi sağlayan IEEE, Researchgate, Springer ve ACM akademik tarama siteleri aracılığıyla bulunan makalelerin özetlerini elde edebilen, kullanımı kolay bir uygulama geliştirdik. Kullanıcı bu uygulamaya giriş yapıp daha önceden yaptığı aramaları ve sonuçlarını görebilmekte aynı zamanda yeni araştırmalar yapabilmektedir. Standart yöntemlerle tarayıcı üzerinden ya da makale siteleri üzerinden uzun bir sürede gerçekleştirilebilen taramayı uygulamamız sayesinde dakikalar içerisinde yapabilmektedir [Şekil 70-73]. Üstelik kullanıcı gözünden kaçabilen, belki de işine yarayabilecek makalelere de erişim sağlanarak verimlilik en üst noktada tutulmuştur. Uçtan uca "on-demand" olarak kullanıcı tercihiyle şekillenebilen bir arama motoru ile araştırma yapılabilen uygulamamız kendi bölümümüz içerisindeki akademisyenler ve lisansüstü öğrenciler tarafından beş aylık süreyle test edilmiş ve kullanılmaya başlanmıştır. Geliştirilen sistem ile araştırma yapılacak alanda gerçekleştirilmiş mevcut çalışmaların büyük çoğunluğuna erişim sağlanarak literatür taramasının sistematik, düzenli ve daha verimli bir şekilde gerçekleştirilmesi sağlanmaktadır.

Geliştirilen sistem, kullanıcıların literatür taramasını hızlı ve kolay bir şekilde gerçekleştirilmesini sağlamaktadır. Bununla birlikte gelecek çalışmalarda bu sisteme eklenecek yeni özellikler ile literatür taraması çok daha verimli bir şekilde gerçekleştirilebilecektir.

Kullanıcılar daha önceden yaptığı araştırma konularında kendini güncel tutabilmesi için otomatize şekilde belirlediği periyotlarda arama yapmasını sağlamak eklenmesi planlanan özelliklerden bir tanesidir. Bu sayede kullanıcılar çalıştığı alanlar hakkında yeni yayınlanmış makalelere de göz atma fırsatı yakalayabileceklerdir.

Uygulamamızda kullanıcıyı anahtar kelime ve filtrelerle yaptığı arama sayesinde ilişkilendirilmiş sonuçlarla direkt olarak buluşturmaktayız. Yapısal olarak dağıtık olan

verimiz üzerinde hiçbir işlem şu aşamada yapılmamaktadır. Hem kullanıcı için daha faydalı olması hem de verinin daha anlamlı hale getirilebilmesi için text-mining algoritmalarından faydalanarak en doğru ve en ilgili sonuçlara ulaşımın sağlanması çalışmanın gelecek hedefini oluşturmaktadır.

Yapısal olarak düzenlenmiş ve anlamlandırılmış verinin üzerinde yapılabilecek bir çalışma olarak da analitik sonuçlar ve tahminlemeler ortaya koymak istiyoruz. Bu bağlamda kullanıcının yaptığı geçmiş aramalar doğrultusunda elde edilen sonuçları görseller ile sunmak ve yaptığı aramalarla ilişkili olan anahtar kelimeler önermek istiyoruz. Geliştirilecek olan bu öneri sistemi sayesinde uygulamamızı kullanan araştırmacının kendi çalıştığı alanlar ile popüler konular arasında bir bağ kurabileceğini sağlamaya çalışacağız.

KAYNAKLAR

[1] Akademik Tez Merkezi. https://akademiktezmerkezi.com/faydali-bilgiler-literatur-taramasi-nasil-yapilir Literatür Nedir? Literatür Taraması Nasıl Yapılır?

[2] S. Khalil, M. Fakir, RCrawler: an R package for parallel web crawling and scraping, SoftwareX 6 (2017) 98–106.

[3] E. Ferrara, P. De Meo, G. Fiumara, R. Baumgartner, Web data extraction, applications and techniques: a survey, Knowledge-based systems 70 (2014) 301–323.

[4] J. Myllymaki, Effective web data extraction with standard XML technologies, Computer Networks 39 (2002) 635–644.

[5] United Stats District Court for the Northern District of California United States, Court of Appeals for the Ninth Circuit 2019 HiQ Labs, Inc v Linkedin Corp San Fancisco.

[6] A. Markham, E. Buchanan, Ethical Decision-Making and Internet Research: recommendations from the AoIR Ethics Working Committee (ver. 2.0), 2012.

[7] Amalia, A., Afifa, R.M., Herriyance, H. (2018) Resource Description Framework Generation for Tropical Disease Using Web Scraping. IEEE International Conference on Communication, Networks and Satellite (Comnetsat)

[8] Sundaramoorthy, K., Durga, R., Nagadarshini, S. (2017). NEWSONE- AN AGGREGATION SYSTEM FOR NEWS USING WEB SCRAPING METHOD. International Conference on Technical Advancements in Computers and Communications

[9] Chaudhari, S., Aparna, R., Tekkur, V.G., Pavan, G.L., Karki, S.R. (2020). Ingredient/Recipe Algorithm using Web Mining and Web Scraping for Smart Chef. 2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)

[10] Uzun, E. (2020). A Novel Web Scraping Approach Using the Additional Information Obtained From Web Pages. IEEE Access (Volume: 8). Pages: 61726 – 61740.

[11] Thomas, D.M., Mathur, S. (2019) Data Analysis by Web Scraping using Python. Proceedings of the Third International Conference on Electronics Communication and Aerospace Technology [ICECA 2019] IEEE Conference Record # 45616; IEEE Xplore ISBN: 978-1-7281-0167-5

[12] Ertam, F. (2018). Deep learning based text classification with Web Scraping methods. International Conference on Artificial Intelligence and Data Processing (IDAP)

[13] Junjoewong, L., Sangnapachai, S., Sunetnanta, T. (2018). ProCircle: A promotion platform using crowdsourcing and web data scraping technique. Seventh ICT International Student Project Conference (ICT-ISPC)

[14] Diouf, R., Sarr, E.N., Sall, O., Birregah, B., Bousso, M., Mbaye, S.N. (2019). Web Scraping: State-of-the-Art and Areas of Application. IEEE International

Conference on Big Data (Big Data)

[15] Kieron, T., Pastrana, S., Collier, B. (2020) A tight scrape: methodological approaches to cybercrime research data collection in adversarial environments. IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)

[16] Akrianto, M.I., Hartanto, A.D., Priadana, A. (2019). The Best Parameters to Select Instagram Account for Endorsement using Web Scraping. 4th International Conference on Information Technology, Information Systems and Electrical Engineering (ICITISEE)

[17] Verma, A., Vanshika, Mittal., Dawn, S. (2019). FIND: Fake Information and News Detections using Deep Learning. Twelfth International Conference on Contemporary Computing (IC3)

[18] Park, A.J., Quadari, R.N., Tsang, H.H. (2017). Phishing website detection framework through web scraping and data mining. 8th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)

[19] Kanetkar, S., Nayak, A., Swamy, S., Bhatia G. (2014). Web-based personalized hybrid book recommendation system. International Conference on Advances in Engineering & Technology Research (ICAETR - 2014)

[20] Belfin, R.V., Shobana, A.J., Manilal, M., Mathew, A.A., Babu, B. (2019). A Graph Based Chatbot for Cancer Patients. 5th International Conference on Advanced Computing & Communication Systems (ICACCS)

[21] Trifa, A., Sbai, A.H., Chaari, W.L. (2017). Evaluate a Personalized Multi Agent System through Social Networks: Web Scraping. IEEE 26th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE).

[22] Uriawan, W., Wahana, A., Wulandari, D., Darmalaksana, W., Anwar R. (2020). Pearson Correlation Method and Web Scraping for Analysis of Islamic Content on Instagram Videos. 6th International Conference on Wireless and Telematics (ICWT).

[23] Kayaalp, M., Ozyer, T., Ozyer, S.T. (2009). A Collaborative and Content Based Event Recommendation System Integrated with Data Collection Scrapers and Services at a Social Networking Site. International Conference on Advances in Social Network Analysis and Mining

[24] Chaulagain, R.S., Pandey, S., Basnet R.S., Shakya, S. (2017). Cloud Based Web Scraping for Big Data Applications. IEEE International Conference on Smart Cloud (SmartCloud).

[25] Lunn, S., Zhu. J., Ross, M. (2020). Utilizing Web Scraping and Natural Language Processing to Better Inform Pedagogical Practice. IEEE Frontiers in Education Conference (FIE).

[26] Malik, K.S., Rizvi S. (2011). Information Extraction Using Web Usage Mining, Web Scrapping and Semantic Annotation. International Conference on Computational Intelligence and Communication Networks

[27] Vikipedi, Özgür Ansiklopedi. https://en.wikipedia.org/wiki/Web_crawler Web Crawler
[28] Google Developers. https://developers.google.com/custom-search/v1/overview Custom Search JSON API
[29] Google Developers. https://developers.google.com/custom-search/v1/site_restricted_api Custom Search Site Restricted JSON API
[30] The Python Package Index. https://pypi.org/project/reppy/ Reppy kütüphanesi
[31] The Python Package Index. https://pypi.org/project/requests/ Requests kütüphanesi
[32] The Python Package Index. https://pypi.org/project/beautifulsoup4/ BeatifulSoup4 kütüphanesi
[33] The Python Package Index. https://pypi.org/project/extraction/ Extraction kütüphanesi
[34] Springer Link. http://link.springer.com/
[35] ACM Digital Library. http://dl.acm.org/
[36] IEEE Xplore Digital Library. https://ieeexplore.ieee.org/
[37] Researchgate. https://www.researchgate.net/
[38] Marmara Üniversitesi Kütüphane ve Dokümantasyon Daire Başkanlığı. https://katalog.marmara.edu.tr/vetisbt/
[39] Science Direct. https://www.sciencedirect.com/
[40] Scopus. https://www.scopus.com/
[41] Academia. https://www.academia.edu/
[42] Webmaster Portalı. https://wmaraci.com/nedir/backend Backend Nedir? Backend Yazılım ve Backend Developer
[43] Firebase Cloud Firestore Veritabanı. https://firebase.google.com/docs/firestore
[44] https://medium.com/@akanesen/api-nedir-entegrasyonu-nas%C4%B1l-yap%C4%B1l%C4%B1r-578f93578e46
[45] FastAPI. https://fastapi.tiangolo.com/ Python FastAPI Framework
[46] Flask. https://flask.palletsprojects.com/ Python Flask Framework
[47] Pydantic. https://pydantic-docs.helpmanual.io/ Python pydantic kütüphanesi
[48] Swagger. https://swagger.io/docs/ Servis dokümantasyon kütüphanesi
[49] Uvicorn. https://www.uvicorn.org/ Python sunucu kütüphanesi
[50] Webmaster Portalı. https://wmaraci.com/nedir/frontend Frontend Nedir? Frontend Yazılım ve Frontend Developer
[51] Fink, G., Flatow I. (2014). Introducing Single Page Applications. Pro Single Page Application Development. Pg. 171-186
[52] Vikipedi, Özgür Ansiklopedi. https://tr.wikipedia.org/wiki/React React
[53] Nextjs. https://nextjs.org/ React geliştirme frameworkü
[54] Redux. https://redux.js.org/ Redux state yönetim kütüphanesi
[55] Bootstrap. https://getbootstrap.com/ Bootstrap CSS kütüphanesi
[56] TailwindCss. https://tailwindcss.com/ Tailwind CSS kütüphanesi
[57] Material UI. https://material-ui.com/ Material UI CSS kütüphanesi

ÖZGEÇMİŞ

Kişisel Bilgiler

Adı Soyadı:	Muhammed Ali Mutlu

Eğitim Durumu

Lisans 2:	Bahçeşehir Üniversitesi, 2012-2017 Elektrik-Elektronik Mühendisliği, GANO:3.22/4.
Lisans 1:	Bahçeşehir Üniversitesi, 2012-2017 Bilgisayar Mühendisliği, GANO: 3.58/4.

İş Deneyimi

Kurum 2:	NTT Data Business Solutions Turkiye, 2018 – Devam Kıdemli Yazılım Mühendisi.
Kurum 1:	Vektora Bilişim, 2016 – 2018 Yazılım Mühendisi.
Staj 2:	Isısan A.Ş. Temmuz 2016 Proje Asistanı.
Staj 1:	Erciyes Çelik Halat, Temmuz 2015 Stajyer Mühendis.

Dil Bilgisi

Dil 1:	Türkçe, Ana dil yetkinliği.
Dil 2:	İngilizce, İkinci dil yetkinliği.

İletişim

Adres:	Ümraniye, İstanbul