

KOCAELİ ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BİTİRME ÇALIŞMASI

En Uygun Bileti Bulma Sitesi

Sena Öktem

Mehmet Aygün

KOCAELİ ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BİTİRME ÇALIŞMASI

En Uygun Bileti Bulma Sitesi

Sena Öktem

Mehmet Aygün

Dr. Öğr. Üyesi Mehmet Zeki Konyar

Danışman, Kocaeli Üniv.

.....

Dr. Öğr. Üyesi Kaplan Kaplan

Jüri Üyesi, Kocaeli Üniv.

.....

Dr. Öğr. Üyesi İrfan Kösesoy

Jüri Üyesi, Kocaeli Üniv.

.....

Tezin Savunulduğu Tarih: 07.06.2023

ÖNSÖZ VE TEŞEKKÜR

Bu tez çalışması selenium kütüphanesi kullanılarak dinamik web sitelerinden farklı firmalara ait otobüs biletlerinin çekilmesi ve en uygununun kullanıcıya sunulması amacıyla gerçekleştirilmiştir.

Tez çalışmamın tüm aşamalarında bilgi ve destekleriyle katkıda bulunan Dr. Öğr. Üyesi Mehmet Zeki Konyar Hocamıza teşekkür ediyoruz.

Hayatım boyunca bana güç veren en büyük destekçilerimiz, her aşamada sıkıntılarımızı ve mutluluklarımızı paylaşan sevgili ailemize teşekkürlerimizi sunarız.

Mayıs – 2023

Sena Öktem - Mehmet Aygün

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

Öğrenci No: 180202066

Adı Soyadı: Mehmet Aygün

Öğrenci No: 190202054

Adı Soyadı: Sena Öktem

İÇİNDEKİLER

ÖNSÖZ VE TEŞEKKÜR	iii
Şekiller Dizini	vi
ÖZET	vii
ABSTRACT	viii
1. GİRİŞ	1
2. Genel Bilgiler	2
2.1. Projenin Amacı	2
2.2. Online Alışveriş	2
2.2.1. Online Alışverişin Etkileri	3
2.2.2. Online Alışverişin Ekonomideki Yeri	4
2.2.3. Online Alışverişin Türleri	5
2.2.4. Online Bilet Alışverişi	6
2.2.5. Örnek Web Siteleri	6
2.3. Büyük Veri	8
2.3.1. Büyük Verinin Özellikleri	8
2.4. Web Kazıma	10
2.4.1. Web Kazıma Nereelerde Kullanılır?	10
2.4.2. Web Sitelerinde Web Kazıma	11
2.4.2.1. Dinamik Olmayan Web Sitelerinde Python Web Kazıma	11
2.4.2.2. Dinamik Web Sitelerinde Python ile Web Kazıma	13
3. Malzeme ve Yöntem	16
3.1. Python	16
3.2. Kullanılan Kütüphaneler	16
3.2.1. Flask	16
3.2.2. Selenium	18
3.2.3. Undetected_chromdriver	18
3.2.4. Pymongo	19
3.2.5. Time	20
3.2.6. Jinja	20
3.3. Projenin Class Yapısı	22

3.3.1. Templates Klsörü.....	22
3.3.2. Static Klsörü	22
3.3.3. Scraping.py Classı	22
3.4. MongoDB.....	22
3.5. MongoDB Compass	24
3.6. Csv Uzantılı Dosya.....	25
3.7. Şehirler.csv Dosyasının Oluşturulması	26
3.7.1. Şehirler.csv Dosyasının MongoDB Veri Tabanına İmport Edilmesi	26
3.8. Kullanılan Algoritmalar	27
3.8.1. Login ve Registration	27
3.8.1.1. Login ve Registration Ekranlarının Tasarımı	27
3.8.2. Bilet Seçim Ekranı.....	30
3.8.2.1. Dropdownlara Veri Çekilmesi	30
3.8.2.2. Web Kazıma İşlemleri	31
3.8.2.3. Driverın Özelliklerinin Değiştirilmesi	32
3.8.2.4. Web Kazıma ile Veri Çekme	32
4. Sonuçlar ve Öneriler	39
Kaynakça.....	40
ÖZGEÇMİŞ	42

Şekiller Dizini

Şekil 2-1 Metro turizm bilet arama sayfası	7
Şekil 2-2 Metro turizm bilet sonuç sayfası.....	7
Şekil 2-3 Obilet arama sayfası	8
Şekil 2-4 Obilet sonuç sayfası.....	9
Şekil 2-5 Web kazıma mimarisi	10
Şekil 2-6 Web kazıma adımları.....	12
Şekil 2-7 Static ve dinamik web sitelerinde farklılıklar	15
Şekil 3-1 Dosya yapısı	22
Şekil 3-2 MongoDB çalışma mantığı.....	24
Şekil 3-3 MongoDB Compass import ekranı	26
Şekil 3-4 MongoDB bağlantısı.....	27
Şekil 3-5 Register html kodu.....	28
Şekil 3-6 Geçiş sağlayan javascript kodu.....	28
Şekil 3-7 MongoDB'ye kaydetme kodu.....	29
Şekil 3-8 MongoDB veri arama kodu	29
Şekil 3-9 Login ekranı.....	29
Şekil 3-10 Şehirlerin aktarıldığı html kodu	30
Şekil 3-11 Bilet arama ekranı.....	31
Şekil 3-12 Driverın optionslarının değiştirilme kodu.....	32
Şekil 3-13 Metro firması url'i	33
Şekil 3-14 Metro firması kalkış yeri çekme kodu	33
Şekil 3-15 Metro firması kalkış saati çekme kodu	34
Şekil 3-16 Metro firması varış yeri çekme kodu	34
Şekil 3-17 Metro firması bilet fiyatı çekme kodu	34
Şekil 3-18 Varan firması url'i	35
Şekil 3-19 Varan firması kalkış saati çekme kodu	35
Şekil 3-20 Bilet fiyatlarının artan olarak sıralanma kodu	35
Şekil 3-21 Bilet fiyatlarının azalan olarak sıralanma kodu	36
Şekil 3-22 Bilet sonuç ekranına yönlendirme kodu	36
Şekil 3-23 Bilet sonuçlarının html'e aktarılma kodu	37
Şekil 3-24 Bilet sonuç ekranı	37
Şekil 3-25 Bilet sonuç ekranı artan fiyata göre sıralanması	38
Şekil 3-26 Bilet al seçeneğine tıklandıktan sonra yönlendirilen ekran	38

En Uygun Bileti Bulma Sitesi

ÖZET

Günümüzde birden çok siteye girerek ihtiyaç duyulan bilginin her bir sitede aratılarak tek tek sonuçlarına ulaşmak ve bu verileri karşılaştırmak oldukça zaman alan bir işlemdir. Bu problemi çözmek adına istenilen tarih aralığında ve istenilen konuma giden otobüsleri bulup bu seferlere ait bilet fiyatlarını farklı sitelerden alıp karşılaştıran ve en uygun olanı bulan bir web sitesi öngörülmüştür. Bu problemi çözmek adına seferlerin farklı sitelerdeki bilet fiyatlarını bulabilmek için web kazıma işlemi kullanılarak diğer sitelerde bilet fiyat taraması yapılması şeklinde bir çözüm üreilmeye çalışılmıştır. Bu sistemin gerçekleştirilmesinde flask Web frameworkü ve veri tabanı olarak NoSQL veritabanlarından biri olan MongoDB kullanılacaktır.

Anahtar Kelimeler : Web Programlama, MongoDB, NoSQL veritabanı, Web Kazıma, headless browser, selenium

Best Ticket Finder Website

ABSTRACT

Today, it is a time-consuming process to search for the required information by searching individually on multiple websites and comparing the results. In order to solve this problem, a website is envisaged that can find buses traveling to the desired location within the desired date range, and compare ticket prices from different sites to find the most suitable one. To find ticket prices for trips on different sites, web scraping will be used, and Flask Web framework and MongoDB, one of the NoSQL databases, will be used as the database.

Keywords: Web Programming, MongoDB, NoSQL database, Web Scraping, headless browser, Selenium.

1. GİRİŞ

Gidilecek olan yere en uygun olan bileti bulma işlemi tek tek web sitelerine girerek gidilecek yeri ve tarihi seçerek bulunabilmektedir. Ancak bu aratma işlemi her otobüs firmasının web sitesine girmeyi ve her sayfada aynı filtreleme işlemlerinin yapılmasını gerektirmektedir. Her otobüs firmasının ismi bilinemeyeceğinden ve her otobüs firmasına ait bilet fiyatlarının manuel olarak not tutulup bilet fiyatlarının kıyaslanması çok zor olacağından bu işlem zaman kaybına sebep olmaktadır. Bu şekilde her şirketin anlık fiyat değişimlerinin takip edilmesi de oldukça zordur. İstenilen en uygun otobüs biletini bulabilmek için her otobüs şirketinin bilet fiyatlarına ulaşılabilinmeli ve bu bilgilerin anlık olarak güncellenebilmesi gerekmektedir. Bu problemi çözmek adına Web Kazıma teknolojisi ortaya çıkmıştır. Web kazıma ile web sitelerinden istenilen veriler çekilebilmektedir. Web kazıma işlemi, sistem tasarımına uyan veri tabanlarıyla desteklenerek verilerin depolanması sağlanmaktadır. Bu çalışmamızda web scraping teknolojisi kullanılarak otobüs biletlerinin başka firmalardaki fiyatlarının çekilmesi hedeflenmiştir. Çekilen verilerin depolanması için NoSql veri tabanlarından biri olan MongoDB veri tabanı kullanılması planlanmaktadır. Kullanıcı dostu bir ara yüz tasarımıyla aratılmak istenilen biletin bilgileri diğer web sitelerinden çekilerek kullanıcı ekranına gelmesi ve kullanıcının gelen biletleri ucuzdan pahalıya veya pahalıdan ucuza şeklinde filtrelerle filtreleyebilmesi hedeflenmiştir.

2.Genel Bilgiler

2.1. Projenin Amacı

Bu projede farklı otobüs seyahat sitelerinden çekilen verilerin bir web sitesinde kıyaslanarak, artan veya azalan bir şekilde kullanıcıya sunulması amaçlanmıştır.

Web sitesinin kullanımı için kullanıcıların üye olması gerekmektedir. Açılan giriş ekranı sayfasından kullanıcılar siteye üyelik oluşturabilirler. Üye olma ekranında kullanıcı email ve şifre bilgilerini girerek üyeliğini oluşturur. Kullanıcının bilgiler veri tabanına kaydedilir. Kullanıcı üye ise giriş ekranından email ve şifre bilgilerini girerek bilet arama sayfasına yönlendirilir. Kullanıcı bilet arama sayfasında aratmak istediği sefere ait kalkış varış ve tarih bilgilerini seçerek bilet arama işlemini gerçekleştirir. Kullanıcı bu bilgileri seçtikten sonra seyahat sitelerinden web kazıma ile veriler çekilir. Çekilen bu veriler karışık bir şekilde ekranda gösterilir. Sonuç ekranında kullanıcıya sıralama yapması için bir filtre çıkar. Kullanıcı bu filtrede azalan veya artan seçeneklerinden birini seçerek karışık olarak gelen biletleri seçimine göre ekranında sıralar. Kullanıcının karşısına çıkan sonuç ekranında kullanıcının seçtiği sefere ait biletlerin firma bilgisi, seferlerin kalkış varış noktaları, sefer tarihi ve saati, fiyatı kullanıcıya sunulmaktadır. İlgili sefer ile ilgili web kazıma yapılan seyahat sitelerinde yer kalmayan seferlere ait biletler kullanıcının ekranına yansıtılmaz. Kullanıcının uygun bulduğu sefere ait satın al butonuyla kullanıcı seçtiği sefere ait firmanın sitesine yönlendirilir.

2.2. Online Alışveriş

Online alışveriş, insanların internet üzerinden yaptığı alışveriş işlemleridir. Bu alışveriş yönteminde, insanlar istediği ürünleri bir web sitesi veya telefonlarından o sitelerin uygulamaları aracılığıyla satın alır. Kullanıcılar, internet üzerinden veya uygulamalardaki mağazalardan ürünleri seçebilir, sipariş verebilir ve ödeme yapabilirler. Ürünler genellikle kargo ile kullanıcıya ulaştırmaktadır. Online alışveriş, kullanıcılara çok fazla ürün seçeneği, birçok siteden araştırarak karşılaştırma ve çevrimiçi bir şekilde ödeme kolaylığı sunmaktadır.

2.2.1. Online Alışverişin Etkileri

Online alışveriş, çok fazla etkiye sahip olan bir ticaret yöntemidir. Aşağıda online alışverişin bazı etkileri verilmiştir:

Kolaylık: Online alışveriş, kullanıcıların ürünleri mağazalara gitmeden veya herhangi bir yerden kolayca satın almasına yardımcı olur. Mağazalara gitme, sıra beklemek gibi fiziksel alışverişteki zorluklarla uğraşmadan alışveriş yapılabilir.

Geniş ürün seçeneği: Online alışveriş siteleri, tüketicilere geniş bir ürün yelpazesi sunar. Kullanıcıların çok fazla mağaza ve markaların ürünlerine erişebilmesi sağlanmıştır. Bu da tüketicilerin istedikleri ürün veya benzeri ürünleri bulmak için çok fazla seçenek sunmaktadır.

Fiyat karşılaştırması: Online alışveriş, farklı sitelerdeki satıcıların ürünlerinin fiyatlarını kolay bir şekilde karşılaştırmak için uygun bir ortam sağlamaktadır. Bu şekilde kullanıcılar, istedikleri üründe en uygun fiyatı bulmak ve tasarruf etmek için kendisine en uygun olan seçenekleri değerlendirebilir.

7/24 alışveriş imkanı: Zaman kısıtlaması olmadan alışveriş yapma imkanı, internetin olduğu her yerden ve zamandan alışveriş yapmayı kolaylaştırmaktadır. [1]

Uluslararası alışveriş: Online alışveriş siteleri, kullanıcılara dünya her yerindeki mağazalara ulaşma imkanı sağlamaktadır. Bu sayede kullanıcıların, bulunduğu ülkede bulunmayan fakat farklı ülkelerde bulunan özel ürünleri ve bu ürünlerin oradaki fiyatları ile kıyaslayarak daha iyi fiyatları bulmasına yardımcı olmaktadır.

Kişiselleştirilmiş deneyim: Online alışveriş siteleri, kullanıcıların tercih ettikleri ürünleri veya mağazaları ve geçmişte yapmış oldukları alışverişleri takip ederek kişiselleştirilmiş önerilerde bulunmaktadır. Bu, kullanıcıların istedikleri ürünleri daha kolay bulmalarını sağlamaktadır.

İncelemeler ve değerlendirmeler: Online alışverişte, kullanıcılar diğer kullanıcıların ürünler hakkındaki yorumlarını okuyup ve bu ürünün kalitesi hakkında daha fazla bilgi sahibi olabilir. Bu da kullanıcıların ürünler daha bilinçli kararlar vermelerine yardımcı olmaktadır.

Çevrimiçi kampanyalar ve indirimler: Online alışveriş siteleri, çeşitli indirimler, kampanyalar ve kuponlar vererek kullanıcıların ürünleri kendisi için daha uygun olmasına yardımcı olmaktadır. Bu da yapılan alışverişlerin maliyetlerini düşürmektedir.

Yukarda belirtilen olumlu etkilerin yanı sıra, online alışverişin bazı olumsuz etkileri de bulunmaktadır. Örneğin, bir ürünü mağazaya gidip denemeden almak o ürünün beden, renk gibi özelliklerinin istenilen şekilde olup olmadığı anlaşılmamaktadır. Ayrıca, online alışverişte kullanıcılar ürünleri fiziksel olarak denemeden satın alıp beğenmediğinde ürünü geri iade etme süreçleri bazı zamanlarda uzun sürmektedir. Güvenlik ve gizlilik gibi konular da online alışverişin olumsuz etkileri arasında yer almaktadır. Kullanıcıların kredi kartı bilgilerinin güvenliği ve kişisel verilerin korunması konusunda endişeler olmaktadır.

Bununla birlikte, online alışverişin etkileri genellikle kullanıcılar için olumlu olmaktadır. İnternetin yaygınlaşmasıyla birlikte insanlar daha fazla online alışveriş yapmaktadır ve bu durumun devam etmesi beklenmektedir.

2.2.2. Online Alışverişin Ekonomideki Yeri

Online alışveriş, son zamanlarda hızla büyüyen ve ekonomide önemli bir yere sahip olan bir sektördür. İnternetin yaygınlaşmasıyla birlikte insanların alışveriş yapma yöntemi de değişmiş ve insanlar ürün ve hizmetleri online alışveriş sitelerinden satın almaya başlamıştır.

Online alışveriş, birçok avantaj sağlamaktadır. Öncelikle, kullanıcılara daha geniş bir ürün yelpazesi sunar ve coğrafi olarak sınırları ortadan kaldırmaktadır. İnsanlar artık dünyanın herhangi bir yerindeki istediği bir mağazadan istediği ürünü satın alabilmektedir. Ayrıca, online alışveriş kullanıcılara zaman ve enerji tasarrufu sağlamaktadır. Alışveriş yapmak için mağazalara gidip gezmek yerine, siteye girip ile istedikleri ürünlere erişebilmektedirler.

Ekonomik açıdan, online alışverişin büyük bir etkisi olmuştur. İnternet üzerinden yapılan satışlar, perakende sektöründeki mağazaların pazar payını azaltmıştır. Bu, büyük perakende zincirlerinin online alışveriş sitelerine yatırım yaparak çevrimiçi varlık geliştirmesine neden olmaktadır. Ayrıca, online alışverişin lojistik sektörü

üzerinde de olumlu etkisi bulunmaktadır. Ürünlerin teslimatı için yeni dağıtım ağları ve süreçler oluşturulup bu ağlarda istihdam fırsatları yaratılmıştır.

Online alışveriş ayrıca küçük işletmeler için de büyük fırsatlar sunmaktadır. İnternet üzerinden satış yapma imkanı, bu işletmelere düşük maliyetli bir şekilde çok fazla müşteriye ulaşma ve satmış olduğu markaların öğrenilmesine yardımcı olmaktadır. Bu da yerel ekonomide rekabeti arttırarak katkıda bulunmaktadır.

2.2.3. Online Alışverişin Türleri

Online alışverişin birçok çeşidi bulunmaktadır. Bazı çeşitleri aşağıdaki gibi sıralanabilir:

E-ticaret: Genel olarak perakende ürünlerin online alışveriş sitelerinden satın alınmasıdır. Bu tür alışverişte kullanıcılar, e-ticaret sitelerindeki ürünleri seçerek bu ürünleri satın almaktadır.

İndirimli alışveriş: İndirimli online alışveriş siteleri veya uygulamaları, kullanıcılara indirimli fiyatlarda ürünler sunmaktadır. Bu tür online alışveriş siteleri genellikle sürpriz satışlar, toplu bir şekilde satın alma veya kuponlar gibi yöntemlerle ürünlerde indirimler sağlamaktadır.

Pazar yerleri: Pazar yerleri, farklı satıcıların kendi ürünlerini sergileyebildiği online alışveriş siteleridir. Kullanıcılar, pazar yerlerindeki farklı satıcılardan farklı ürünleri satın alabilmektedir.

C2C (tüketici-tüketici): C2C alışveriş türü, kullanıcıların kendi kullanmış olduğu ürünlerini diğer kullanıcılara satarak veya ikinci el ürünleri satın alarak gerçekleştirdikleri bir alışveriş modelidir. Bu tür alışveriş genellikle online alışveriş sitelerinde veya açık artırma sitelerinde gerçekleşmektedir.

B2B (işletme-işletme): B2B alışveriş türü, bir işletmenin diğer bir işletmeden ürün veya hizmet satın aldığı iş-to-business (işletme-işletme) modelidir. Bu tür alışveriş genellikle büyük ticari sitelerde gerçekleşmektedir.

Bu türler, online alışverişin farklı yönlerini ve çeşitliliğini temsil etmektedir.

2.2.4. Online Bilet Alışverişi

Online otobüs bileti alışverişi, internet üzerinden otobüs seyahatleri için bilet satın alınması anlamına gelir. Otobüs firmaları, online bilet satış siteleri veya mobil uygulamalar aracılığıyla biletlerini satışa sunabilir. Kullanıcılar tercih ettikleri şehere göre seyahat tarihini ve koltuk tercihlerini belirleyebilir, ödeme işlemlerini gerçekleştirebilir ve bileti elektronik olarak alabilmektedir. Bu sayede, otobüs seyahatleri için fiziksel bir gişede veya telefonla rezervasyon yapmak yerine, internet üzerinden kolayca bilet satın alınabilmektedir.

2.2.5. Örnek Web Siteleri

Bu bölümde sefer bilgilerini sunan farklı web sitelerinin genel yapısı üzerinde durulmuştur. Benzer bir yaklaşım tren, uçak, feribot ve diğer ulaşım araçları seferleri için de geçerli olsa da otobüs sefer bilgilerine odaklanılmıştır. Bu web sitelerindeki sefere ait otobüs doluluk oranları, bilet fiyatları vb. bilgiler anlık olarak güncellenmektedir. Bu bölümde bu verileri web kazıma yoluyla çekebilmek için önemli olan zorlukların hangileri olduğunu ve web sitelerinin nasıl görüldüğü açıklanmıştır.

Metro Turizm

Metro Turizm, ciro büyüklüğü ve taşınan yolcu sayısı açısından en büyük kara yolu taşıma şirketlerinden birisidir. Günde ortalama 1400 sefer gerçekleştirip, yılda ortalama 19.000.000 yolcu taşımaktadır. Bu web sitesi kullanıcıların farklı sefer bilgilerini görüntüleyebileceği bir arayüz sunar. Aşağıdaki görüntüde olduğu gibi kullanıcının nerden nereye gideceği ve hangi tarihte gideceği ile ilgili bilgilerin seçilebilmesi için bazı alanlar bulunmaktadır.

Şekil 2-1 Metro turizm bilet arama sayfası[2]

Seçim yapılan seferlerin listelenmesi durumunda aşağıdaki tablo ortaya çıkmaktadır. Aşağıdaki tabloda belirtilen gidiş tarihinde gerçekleştirilecek olan seferlerin saatleri, kalkış ve varış noktaları, otobüsün türü, otobüsün kalkış ve tahmini varış saati, otobüsün sahip olduğu internet, tv, priz, usb, ikram vb. özellikler, sefere ait kalan koltuk sayısı, bilet fiyatı ve tercih edilen seferin bilet alış ekranına aktarılması için her sefere ait seç butonu gösterilmiştir.

→Gidiş Seferleri 25.12.2022 Pazar						
OTOBÜS	KALKIŞ / VARİŞ		ÖZELLİKLER	K.KOLTUK	FİYAT	
	SAMANDIRA TESİS	ANKARA		9	225 ₺	Seç
	SAMANDIRA TESİS	ANKARA		8	225 ₺	Seç
	SAMANDIRA TESİS	ANKARA		15	225 ₺	Seç
	SAMANDIRA TESİS	ANKARA		21	225 ₺	Seç
	SAMANDIRA TESİS	ANKARA		17	225 ₺	Seç
	SAMANDIRA TESİS	ANKARA		27	225 ₺	Seç
	SAMANDIRA TESİS	ANKARA		34	225 ₺	Seç

Şekil 2-2 Metro turizm bilet sonuç sayfası[3]

Obilet

Obilet.com, Türkiye'deki ulaşım firmalarını bir araya getiren bir online seyahat bileti arama portalıdır. Bu bölümde otobüs bileti arama portalı incelenecektir.. Bu web sitesi kullanıcıların farklı firmalardaki farklı sefer bilgilerini kıyaslayabileceği ve en uygun seferleri görüntüleyebileceği bir arayüz sunar. Aşağıdaki görüntüde olduğu gibi kullanıcının nerden nereye gideceği, hangi ulaşım aracıyla gideceği ve hangi tarihte gideceği ile ilgili bilgilerin seçilebilmesi için bazı alanlar bulunmaktadır.



Şekil 2-3 Obilet bilet arama sayfası[4]

Seçim yapılan seferlerin listelenmesi durumunda aşağıdaki tablo ortaya çıkmaktadır. Aşağıdaki tabloda belirtilen gidiş tarihinde gerçekleştirilecek olan seferlerin hangi otobüs firması tarafından gerçekleştirileceği, sefer saatleri, kalkış ve varış noktaları, otobüsün kalkış saati, otobüsün sahip olduğu internet, tv, priz, usb, ikram vb. özellikler, bilet fiyatı ve tercih edilen otobüs firmasına ait seferin bilet alışı ekranına aktarılması için her sefere ait koltuk seç butonu gösterilmiştir.

varan	00:20 (7 Saat 30 Dakika)*	2+1 Alibeyköy Otogan → İzmit Otogan	249,00₺	KOLTUK SEÇ İndirimli Son Koltuklar
60% İndirim Kodu	Yeni Otoban	İncele		
METRO	00:20 (7 Saat 10 Dakika)*	2+1 Alibeyköy Otogan → İzmit Otogan	275,00₺	KOLTUK SEÇ
		İncele		
KALE	00:20 (7 Saat 40 Dakika)*	2+1 Alibeyköy Otogan → İzmit Otogan	279,00₺	KOLTUK SEÇ İndirimli Son Koltuklar
70% İndirim Kodu		İncele		
KAMILKOÇ	00:20 (7 Saat 40 Dakika)*	2+1 Alibeyköy Otogan → İzmit Otogan	300,00₺	KOLTUK SEÇ
		İncele		

Şekil 2-4 Obilet bilet sonuç sayfası[5]

2.3. Büyük Veri

Büyük veri (big data) klasik veri tabanı sistemleri ile depolanması, yönetilmesi ve analiz edilmesi zor olan verilerdir. Büyük veri analiz edilmesi ve verimliliği artırılması için toplanan anlamlı ve işlenebilir veriler bütünüdür.

2.3.1. Büyük Verinin Özellikleri

1. Hacim (Volume): Büyük veri, geleneksel veri işleme sistemlerinin işleyebileceğinden çok daha büyük miktarlarda veriyi içerebilir. Bu veri, terabaytlar, petabaytlar veya hatta daha fazla büyüklükte olabilir. Bu yüzden veri akışı sürekli ve büyük hacimlerde olmalıdır. Dolayısıyla verinin gerçek zamanlı olarak işlenmesi ve bilgiye dönüştürülmesi gerekmektedir.

2. Çeşitlilik (Variety): Büyük veri çeşitli kaynaklardan gelen farklı veri türlerini içerebilir. Yapılandırılmış veriler (veri tabanlarından), metin tabanlı veriler (makaleler, bloglar), multimedya verileri (resimler, videolar), sosyal medya gönderileri ve sensör verileri gibi çeşitli formatlarda olabilir. Sağlıklı bir analiz için çeşitli formattaki bilgilerin birbirine dönüştürülebilir olmaları önemlidir.

3. Hız (Velocity): Verinin, süreklilik arz etmesi ve hızlı olması gerekmektedir. Örneğin, sosyal medya akışları, sensör verileri veya finansal piyasalarla ilgili gerçek zamanlı veriler gibi hızlı bir şekilde toplanan veriler örnek gösterilebilir. Veriyi işleyerek analizinin gerçekleştirilme sürecinin de verinin üretimi ile aynı hızda olması gerekmektedir.

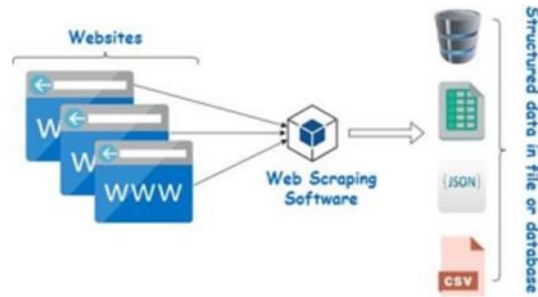
4.Doğruluk (Veracity): Verinin güvenilir olması ve doğru bilgiler içermesi gerekmektedir. Bir veri analizinin doğru sonuçlanabilmesi için doğru olmayan veya analiz konusu ile ilgili olmayan verilerin temizlenmesi gerekmektedir.[6]

5.Değer (Value): Veri setindeki bilgilerin önemli, anlamlı veya faydalı olduğunu ifade eder.

2.4.WEB KAZIMA

Web kazıma, bir web sitesinden verilerin çıkarılmasını ifade eder. Bir web sitesinden veri kopyalayıp yapıştıran bir kullanıcı tarafından manuel olarak yapılabilir veya bir bilgisayar kullanılarak programlı olarak yapılabilir. Yerleşik geliştirici araçlarına sahip web tarayıcıları, komut satırı yardımcı programları ve çeşitli programlama dilleri için kitaplıklar ve çerçeveler dahil olmak üzere web kazıma için birçok araç vardır.

Web kazıma, veri madenciliği, veri analizi ve makine öğrenimi gibi çeşitli amaçlar için yararlı olabilir. Ancak web kazımanın yasal ve etik sonuçlarının farkında olmak önemlidir. Bazı durumlarda, sitelerini kazımadan önce web sitesi sahibinden izin almak gerekebilir. Kazımayı önlemek için web sitesinin hizmet şartlarına saygı duymak da önemlidir.



Şekil 2-5 Web kazıma mimarisi[7]

2.4.1. Web Kazıma Nerelerde Kullanılır?

- Veri madenciliği : Web kazıma, web sitelerinden büyük miktarda veri toplamak ve ardından kalıpları bularak bu verileri analiz etmek için kullanılabilir.

- Fiyat karşılaştırması: Web kazıyıcıları, farklı web sitelerindeki fiyatları karşılaştırmak için kullanılabilir ve bu da en iyi içeriği bulmayı kolaylaştırır.
- Pazar araştırması: Web kazıma, birden fazla web sitesinden ürünler, fiyatlar ve incelemeler hakkında veri toplamak için kullanılabilir ve bu da pazar araştırması için yardımcı olabilir.
- İletişim bilgileri: Web kazıyıcıları, satış ve pazarlama çabaları için yararlı olabilecek web sitelerinden iletişim bilgileri toplamak için kullanılabilir.
- İçerik toplama: Web kazıyıcıları, birden fazla web sitesinden içerik toplamak ve ardından bu içeriği tek bir sitede görüntülemek için kullanılabilir ve bir tür "toplayıcı" web sitesi oluşturur.

gibi pek çok konu başlığında kullanılabilir.

2.4.2. Web Sitelerinde Web Kazıma

2.4.2.1. Dinamik Olmayan Web Sitelerinde Python Web Kazıma

BeautifulSoup ve requests ile Web Kazıma

Web sitesine bir GET isteği göndermek için requests kütüphanesi kullanılmaktadır.[8] Bu istek, web sitesinden sayfanın HTML içeriğinin alınmasını sağlamaktadır. requests.get() fonksiyonu kullanılarak istek gönderebilir ve yanıtı response değişkenine atanmaktadır.

Alınan HTML yanıtını işlemek için BeautifulSoup kütüphanesinin kullanılması gerekmektedir.[9] Bu kütüphane, HTML içeriğinin ayrıştırılmasına ve içindeki öğelere erişilmesine olanak tanır. HTML yanıtını BeautifulSoup ile ayrıştırarak, içerik üzerinde çalışılabilmektedir. Oluşturulan BeautifulSoup nesnesi soup değişkenine atanmaktadır. soup nesnesi üzerinden veri çekilebilmektedir. find() veya find_all() gibi yöntemlerle, HTML içerisindeki belirli öğeler bulunabilmektedir. Bu yöntemler, belirli etiketleri, sınıfları, kimlikleri veya diğer özellikleri kullanarak öğelerin seçilmesini sağlar. Örneğin, find() yöntemini kullanarak belirli bir etikete sahip bir öğeyi bulabilir ve .text özelliğini kullanarak içeriği alınabilmektedir. Böylece, ilgili veri alınabilir ve işlenebilmektedir.

Scrapy ile Web Kazıma

Scrapy kullanarak web kazıma yapmak için öncelikle bir Scrapy projesi oluşturulması gerekmektedir.[10] Terminal üzerinde "scrapy startproject proje_adi" komutu çalıştırılarak bir proje dizini oluşturulabilmektedir.

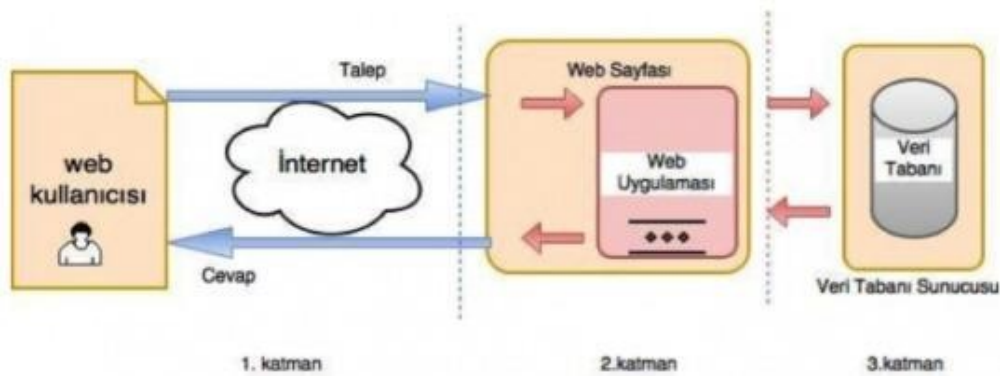
Proje dizinine gittikten sonra, yeni bir spider oluşturmak için "scrapy genspider spider_adi website.com" komutu kullanılmaktadır. Bu komut, spider_adi olarak adlandırılan bir örümcek dosyası oluşturacak ve website.com olarak belirtilen web sitesinde gezinme ve veri kazıma işlemlerini gerçekleştirecektir.

Spider dosyasında, start_urls özelliğiyle başlayacağınız URL'leri belirlenebilmektedir. Ayrıca, parse yöntemini kullanarak sayfaları ayrıştırma ve veri kazıma işlemleri gerçekleştirilebilmektedir. Bu yöntemde XPath veya CSS seçicilerini kullanarak veriler bulunabilir ve gerektiğinde diğer sayfalara geçiş yapılabilmektedir.[11]

Veri kazıma işlemlerini parse yöntemi içinde gerçekleştirdikten sonra, kazınan veriler kullanılabilir veya başka sayfalara geçmek için yield ifadesi kullanılabilir.

Kazınmış Verileri Dışa Aktarma

Kazınmış veriler json dosyası, .csv dosyası vb. aktarılır.



Şekil 2-6 Web kazıma adımları[12]

Bir önceki başlıkta dinamik olmayan web sitelerinde kullanılan yöntemler bahsedildi. Ancak bu yöntemler javascript'e dayanan sitelerde özellikle de Angular, React veya Vue.js ile yapılan sitelerden veri kazırken uygun değildir. Çünkü bu yöntemler ile dinamik bir web sitesine çok fazla istek atmanız ve isteğe dönecek cevabı beklemeniz gerekmektedir. Bu işlem uzun sürebilir, her isteğe geri dönüş alınamayabilir dolayısıyla da bu yöntemler çok sayıdaki Html'i geri döndüremez ve istenilen verim elde edilememiş olurdu. Bu sorunu çözmek için dinamik web sitelerinde web kazıma işlemi için başka yöntemler ortaya çıkmıştır.

2.4.2.2. Dinamik Web Sitelerinde Python ile Web Kazıma

Dinamik web sayfaları, sunucuda depolanan statik HTML sayfaları değil, sunucu tarafından oluşturulan ve tarayıcıya HTML olarak gönderilen sayfalardır.

Headless Browser Nedir, Neden Kullanılır?

Son birkaç yıldır web uygulamaları, HTML ve CSS ile oluşturulmuş basit web sitelerinden evrimleşmektedir. Artık genellikle Angular, React, Vue gibi çerçevelerle oluşturulmuş kullanıcı arayüzlerine sahip çok daha etkileşimli web uygulamaları kullanılmaktadır. Başka bir deyişle, günümüzde JavaScript, web sitelerinde etkileşimde bulunduğunuz hemen hemen her şey dahil olmak üzere web'i yönetmektedir. Bizim amaçlarımız için JavaScript, istemci taraflı(client-side) bir dildir. Sunucu, bir HTML yanıtına enjekte edilen JavaScript dosyalarını veya komut dosyalarını döndürür ve tarayıcı bunu işler. Şimdi, bir tür web kazıma veya web otomasyonu yapıyorsak bu bir sorundur, çünkü çoğu zaman görmek veya kazımak istediğimiz içerik aslında JavaScript koduyla oluşturulur ve sunucunun sağladığı ham HTML yanıtından erişilemez. Bu sorunu çözmek için headless browser kullanılır. Headless browser'ın özelliği grafiksel kullanıcı arayüzünün(GUI) olmamasıdır. Ekranda görüntülemeyen HTML belgelerini işler ve JavaScript kodunu çalıştırır. Komut satırı aracılığıyla headless bir tarayıcıyla etkileşime girilmektedir.[13]

Headless Chrome ile Selenium Kullanımı

Selenium kullanarak dinamik bir web sayfası kazındığında, aslında web sayfasıyla, bağlantıları tıklamak ve formları doldurmak gibi bir kullanıcının eylemlerini simüle edecek şekilde etkileşimde bulunulur. Bu, sık güncellenen veya kullanıcı girdisinin

görüntülenmesini gerektiren web sayfalarından veri kazınmasına olanak tanır. Headless Chrome, görüntü kullanıcı arayüzü olmadan arka planda çalışabilen bir tarayıcıdır. Selenium ise web tarayıcıları otomatize etmek için kullanılan bir otomasyon aracıdır. Bu iki araç bir araya getirilerek, Python programlarıyla web tarayıcısı otomatikleştirilebilir.

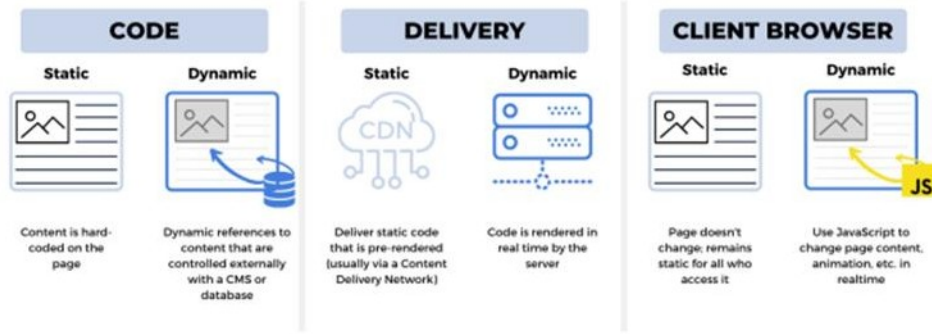
İlk adımda, Selenium WebDriver'ı ve ChromeDriver'ın yüklenmesi gerekmektedir. Selenium WebDriver, web tarayıcılarıyla etkileşim sağlamak için kullanılan bir arabirimdir. ChromeDriver ise Chrome tarayıcısını otomatikleştirmek için gereken bir sürücüdür.[14]

Selenium WebDriver'ı kullanarak bir Chrome tarayıcısı başlatılır. Bu aşamada, ChromeDriver'ın yolunun belirtilmesi gerekmektedir. Ayrıca, Chrome tarayıcısının gizli modda çalışmasını ve GPU kullanımını devre dışı bırakmasını sağlayacak seçenekleri ayarlanabilmektedir. Bu sayede, tarayıcı arka planda çalışacak ve görüntü kullanıcı arayüzünde sunulmayacaktır.

Tarayıcı başlatıldıktan sonra, Selenium ile tarayıcı üzerinde işlemler yapılabilmektedir. Örneğin, bir web sitesine gitmek, belirli öğeleri bulmak, formları doldurmak veya tıklama işlemleri yapmak gibi işlemler gerçekleştirilebilir. Bunun için Selenium'ın sağladığı yöntemler kullanılabilir. Örneğin, get() yöntemi ile bir URL'ye gitmek, find_element_by_*() yöntemleri ile öğeleri bulmak ve click() veya send_keys() gibi yöntemlerle etkileşim sağlamak gibi işlemler yapılabilmektedir.[15]

Headless Chrome ile Selenium Kullanarak Web Kazıma Adımları :

- Yeni bir headless Chrome tarayıcısı açılır.
- Linki verilen web sayfasına gidilir.
- Linki verilen web sayfasının tamamen yüklenmesi beklenir.
- Sayfa kimliğine göre bir öğe bulunur.
- O öğeye ait veri çıkartılır.



Şekil 2-7 Statik ve dinamik web sitelerinde farklılıklar[16]

3.Malzeme ve Yöntem

3.1.Python

Python, genel amaçlı, yüksek seviyeli ve açık kaynaklı bir programlama dilidir. Guido van Rossum tarafından 1991 yılında geliştirilmeye başlanmıştır. Birçok platformda çalışabilmektedir. Windows, macOS, Linux gibi işletim sistemlerinde ve ayrıca mobil cihazlar üzerinde de kullanılabilir. Python, çeşitli programlama alanlarında kullanılan birçok amaca yönelik işlevsel kütüphanelere ve esnek bir çerçeveye sahiptir. Bu kütüphaneler sayesinde bilimsel hesaplama, veri analizi, yapay zeka, web geliştirme, oyun geliştirme vb. birçok alanda programlar oluşturulabilmektedir.

Python, okunması ve yazması kolay bir sözdizimine sahiptir, bu nedenle öğrenmesi diğer bazı programlama dillerine göre daha hızlıdır.

Python'un bazı özellikleri şunlardır:

- 1.Kolay Okunabilirlik: Python, sade ve anlaşılır bir sözdizimine sahiptir. Kodun okunabilirliği ve anlaşılabilirliği yüksektir.
- 2.Dinamik ve Esnek: Python, değişkenlerin türlerini tanımlamak için tip belirleme gerektirmez. Bu durum kodun esnekliğini sağlamaktadır.
- 3.Büyük Standard Kütüphane: Python, zengin bir standart kütüphaneye sahiptir. Bu kütüphane, dosya işleme, ağ socketleri, veri tabanı bağlantıları, veri işleme, metin işleme, grafikler oluşturma ve daha pek çok işlevi destekler.
- 4.Yüksek Seviyeli Veri Yapıları: Python, liste, demet, sözlük gibi yüksek seviyeli veri yapılarını doğrudan destekler. Bu veri yapıları, veri organizasyonunu kolaylaştırır ve işlemleri daha verimli hale getirir.

3.2. Kullanılan Kütüphaneler

3.2.1.Flask

Flask, Python programlama diliyle web uygulamaları geliştirmek için kullanılan bir web frameworküdür.[17] Web frameworkleri, web uygulamalarını oluşturmak için

gereken temel işlevleri içeren bir dizi araç ve kütüphane sağlamaktadır.

Flask, basit bir yönlendirme (routing) mekanizması sunmaktadır. Yani, kullanıcıların web tarayıcılarından gelen URL'lere göre uygulamanın hangi kodun çalıştırılacağını belirlemesini sağlar. Bu sayede, web uygulamasının farklı sayfalarına erişim ve işlemler yönlendirilebilir.

Flask ayrıca HTTP isteklerini ve yanıtlarını işlemek için yardımcı işlevler sunar. Örneğin, kullanıcıdan gelen form verilerini almak, veri tabanına erişmek, kullanıcı oturumlarını yönetmek gibi yaygın web işlevlerini kolayca gerçekleştirebilmektedir.

Flaskte Kullanılan Bazı Fonksiyonlar

Flaskte sıklıkla kullanılan bazı temel fonksiyonlar şunlardır:

1.redirect: redirect fonksiyonu, kullanıcıyı başka bir URL'ye yönlendirmek için kullanılır. Bu fonksiyon, kullanıcı tarayıcısına yeni bir istek gönderir ve belirtilen URL'e yönlendirilir.

2.render_template: HTML şablonlarının kullanılmasını sağlar. Fonksiyon, bir HTML şablonunu alır ve dinamik verilerle birlikte render edilmiş HTML'i kullanıcıya gönderir. Bu, web uygulamanızda dinamik içerik göstermek için kullanışlıdır.

3.request: request nesnesi, gelen HTTP isteği hakkında bilgilere erişilmesini sağlamaktadır. Kullanıcı tarafından gönderilen form verilerini, URL parametrelerini veya diğer istek bilgilerini bu nesne üzerinden alabilirsiniz. Örneğin, bir kullanıcının girdiği bir form değerini almak için request.form['input_name'] şeklinde kullanılabilir.

4.url_for: url_for fonksiyonu, flask uygulamasında tanımlanmış bir fonksiyonun URL'sini oluşturmak için kullanılır. Normalde, web uygulamalarında bir sayfaya erişmek için o sayfanın URL'sini doğrudan belirtmek gerekir. Ancak url_for kullanarak, sayfalar arasında daha esnek bir bağlantı sağlanabilmektedir. Fonksiyon, verilen fonksiyon adına ve gerekli parametrelere dayanarak URL'yi otomatik olarak oluşturur.

Bu fonksiyonlar, flask uygulamanızda sayfalar arasında geçiş yapmayı, dinamik

içerik göstermeyi ve kullanıcıların gönderdiği verileri işlemeyi sağlar. Flaskin temel özellikleri arasındadır ve birçok flask uygulamasında yaygın olarak kullanılmaktadır.

3.2.2. Selenium

Selenium, web kazıma için kullanılan bir kütüphanedir. Dinamik içerikli web sitelerinden veri çekmek için uygun bir kütüphanedir. Verinin çekileceği sayfanın içeriğini dinamik olarak yüklemesi selenium kütüphanesini dinamik web sitelerinden veri çekmek için uygun olmasını sağlamaktadır.

Selenium, web tarayıcılarını otomatik olarak kontrol edebilir ve tarayıcı üzerinde gerçek kullanıcı etkileşimini taklit edebilmektedir. Bu, sayfaları yüklerken JavaScript kodunu çalıştırabilir, AJAX çağrılarını yapabilir ve kullanıcı etkileşimlerini simüle edebilir. Böylece, dinamik içeriğe veya kullanıcı etkileşimine dayalı web uygulamalarından veri çekmek için selenium kullanılabilir.

3.2.3. undetected_chromedriver

undetected_chromedriver, gelişmiş tarayıcı otomasyonu engelleme tekniklerine karşı dirençli bir şekilde çalışan özelleştirilmiş bir ChromeDriver sürümüdür. Bu kütüphane, tarayıcı otomasyonunu önlemek için kullanılan tespit yöntemlerini atlatabilir ve normal bir kullanıcı gibi davranırken web tarayıcısının otomatik olarak kontrol edilmesini sağlamaktadır.

undetected_chromedriverın bazı özellikleri şunlardır:

- 1.Tarayıcı otomasyonunu önleme: undetected_chromedriver, tarayıcı otomasyonunu tespit etmeye yönelik önlemleri aşar ve botların normal kullanıcılar gibi görünmesini sağlar.
- 2.JavaScript uyumu: undetected_chromedriver, JavaScript'in tam uyumlu bir şekilde çalışmasını sağlar. Bu sayede, JavaScript tarafından oluşturulan içeriği düzgün bir şekilde yükleyebilir ve etkileşimleri simüle edebilir.
- 3.Proxy desteği: undetected_chromedriver, HTTP veya SOCKS proxylerini destekler. Bu sayede, IP adresi gizlenebilir veya farklı coğrafi konumlardan web sitelerine erişilebilir.

4.Başlık ve diğer ayarlar: `undetected_chromedriver`, tarayıcının başlık, dil ayarları, tarayıcı boyutu gibi özelliklerin özelleştirilmesine olanak tanımaktadır. Böylece, botun farklı bir tarayıcı profili kullanması sağlanabilir.

3.2.4. Pymongo

Pymongo, MongoDB veri tabanı ile etkileşimde bulunmak için kullanılan bir kütüphanedir. MongoDB, belge tabanlı bir NoSQL veri tabanıdır ve Pymongo, Python uygulamalarının MongoDB veri tabanını okuma, yazma, güncelleme ve sorgulama gibi işlemleri gerçekleştirmesini sağlar.

Pymongonun temel amaçlarından bazıları şunlardır:

1.Veritabanı Bağlantısı: Pymongo, Python programlarının MongoDB veritabanına bağlanmasını sağlar. Bu sayede, Python uygulamaları MongoDB'ye erişebilmektedir.

2.Veritabanı Ekleme ve Güncelleme: Pymongo, Python programları aracılığıyla MongoDB'ye veritabanı eklemeyi ve var olan verileri güncellemeyi sağlar. Bu, belirli bir koleksiyona belge eklemek, var olan bir belgeyi güncellemek veya birden çok belgeyi toplu olarak güncellemek gibi işlemleri içerir.

3.Veritabanı Sorgulama: Pymongo, Python programlarına MongoDB veritabanında sorgular yapma yeteneği sağlar. Sorgular, belirli kriterlere göre verileri filtrelemek, sıralamak veya gruplandırmak için kullanılır. Bu, veritabanındaki belirli bir koleksiyondan belirli belgeleri almak veya belirli bir koşula uyan belgeleri sorgulamak gibi işlemleri içerir.

4.Veritabanı Silme: Pymongo, Python programlarıyla MongoDB'den belgeleri silme yeteneği sağlar. Belirli bir koşula uyan belgeleri veya tüm belgeleri silmek gibi işlemleri gerçekleştirebilir.

5.Veritabanı Yönetimi: Pymongo, veritabanı ve koleksiyon yönetimi için işlevler sağlar. Veritabanı ve koleksiyon oluşturma, yeniden adlandırma, silme gibi yönetim işlemlerini gerçekleştirmeyi sağlar.

3.2.5. Time

time kütüphanesi, Python programlarında zamanla ilgili işlemleri gerçekleştirmek için kullanılan bir kütüphanedir. Bu kütüphane, zamanı ölçmek, gecikmeleri kontrol etmek, zaman aralıklarını hesaplamak ve zamanla ilgili diğer işlemleri yapmak için fonksiyonlar içermektedir.

time kütüphanesi, aşağıdaki gibi bazı temel işlevlere sahiptir:

1.Zaman Bekletme: `time.sleep(seconds)` fonksiyonu, programın belirli bir süre boyunca uyku moduna geçip duraklamasını sağlar. Bu, programın belirli bir süre uyumasını veya belirli bir süre gecikmesini sağlar.

2.Zaman Ölçme: `time.time()` fonksiyonu, sistem saati üzerinden geçen zamanı verir. Bu, bir işlemin başlangıç ve bitiş zamanlarını ölçerek bir işlemin ne kadar sürede tamamlandığını bulmayı sağlar.

3.Zaman Formatlama: `time.strftime(format, time_struct)` fonksiyonu, belirli bir tarih ve saat yapısının formatlanması için kullanılır.

4.Tarih ve Saat İşlemleri: `time.localtime(seconds)` fonksiyonu, bir Unix zaman damgasını (epoch) yerel zaman dilimine dönüştürür. `time.gmtime(seconds)` fonksiyonu ise bir Unix zaman damgasını UTC zaman dilimine dönüştürür. Bu, zamanla ilgili hesaplamalar yapmak, tarih ve saat bilgilerini almak veya dönüştürmek için kullanılabilir.

3.2.6. Jinja

Jinja, Python programlama dili için geliştirilmiş bir şablon kütüphanesidir.

Jinja'nın temel amacı, dinamik içerik oluşturmak için şablonları kullanmaktır. Bir Jinja şablonu, statik metin ve değişkenlerin birleşiminden oluşur. Şablonlar, genellikle HTML, XML veya diğer metin tabanlı dosya biçimlerinde kullanılır. Kullanılmak istenen şablonun HTML'i "templates" klasörü içerisinde oluşturularak kullanılır.

Jinja, şablonlar içinde kontrol yapıları (if-else, döngüler), filtreler (verileri

biçimlendirme veya dönüştürme), makrolar (tekrar kullanılabilir işlevler) ve şablon mirasını (temel şablonları genişletmek) destekler. Bu özellikler sayesinde Jinja, şablonlar aracılığıyla dinamik içeriği yönetmek için güçlü bir araç haline gelir.

Jinja'nın popüler bir kullanım alanı, Python tabanlı web uygulamalarında HTML şablonlarının oluşturulmasıdır. Web framework'leri (örneğin Flask veya Django) genellikle Jinja'yı kullanarak dinamik web sayfaları oluştururlar. Bunun yanı sıra, Jinja, e-posta şablonları, raporlar, belgeler ve diğer metin tabanlı çıktılar oluşturmak için de kullanılabilir.

Jinja, Python ile entegre edilebilen bir kütüphanedir ve genellikle Python paket yöneticisi olan pip aracılığıyla yüklenir. pip install Jinja2 komutu ile jinja kütüphanesi projeye dahil edilip kullanılabilir.

Özellik değerine göre sınıflandırmayı öğrenir. Ağacı da sınıflandırır. Karar vermede yardım eder. Karar ağaçlarının anlaşılması ve yorumlanması kolaydır.

Karar ağaçlarındaki en büyük sorun aşırı sığdırmadır. Veri kümesinde performansları iyidir. Ama test veri kümesinde doğrulukları düşüktür. Karar ağaçlarının yarattığı bu sorunun üstesinden gelmek için, veri bilimcileri topluluk öğrenimini çıkardılar.

Topluluk öğrenimi bir dizi farklı modeli kullanarak tahminlerde bulunur. Bir dizi farklı modeli birleştirerek bir topluluk öğrenimi daha esnek ve daha az veriye duyarlı olma eğilimi gösterir. En popüler iki topluluk öğrenme yöntemleri: torbalama ve artırmadır. Torbalama, bir grup modeli paralel şekilde eğitir. Her model verilerin random bir alt kümesinden öğrenir. Güçlendirme, bir grup modeli sırayla eğitir. Her model bir önceki hatalarından çıkarım yapar.

Torbalama uygulaması Random Forest'larda bulunur. Random Forest'larda karar ağaçlarının paralel bir kombinasyonudur. Her ağaç aynı verilerin random alt kümesi üzerinde eğitiliyor ve sınıflandırmayı yapmak için tüm ağaçlardan elde edilen sonuçların ortalaması alınıyor.

3.3. Projenin Class Yapısı

3.3.1. Templates Klasörü

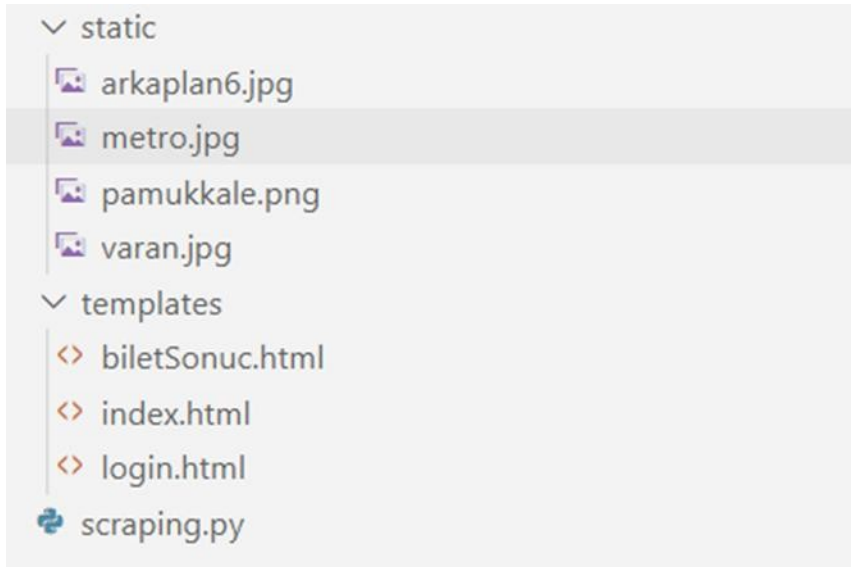
Templates klasörü projede kullanılacak olan HTML şablonlarını içerir. Bu şablonlar jinja kütüphanesi kullanılarak oluşturulmuştur.

3.3.2. Static Klasörü

Static klasörü, projede kullanılacak olan resimlerin kopyalarını içermektedir. Kullanılacak olan bu klasör içerisindeki resimler templates klasörü içindeki html şablonlarında kullanılmaktadır.

3.3.3. Scraping.py Classı

Scraping.py içerisinde login ve registration işlemleri, web kazıma işlemi, web kazıma sonrası elde edilen verilerin kullanıcıya sunulma işlemleri yapılmıştır. Belirtilen işlemlerle ilgili detaylı bilgiler ilerleyen sayfalarda anlatılacaktır.



Şekil 3-1 Dosya yapısı

3.4. MongoDB

MongoDB, bir belge tabanlı (document-based) NoSQL veritabanıdır. Geleneksel ilişkisel veritabanlarından farklı olarak, verileri belgeler halinde saklar. Bu belgeler genellikle JSON veya BSON formatında ifade edilen, anahtar-değer çiftleriyle temsil

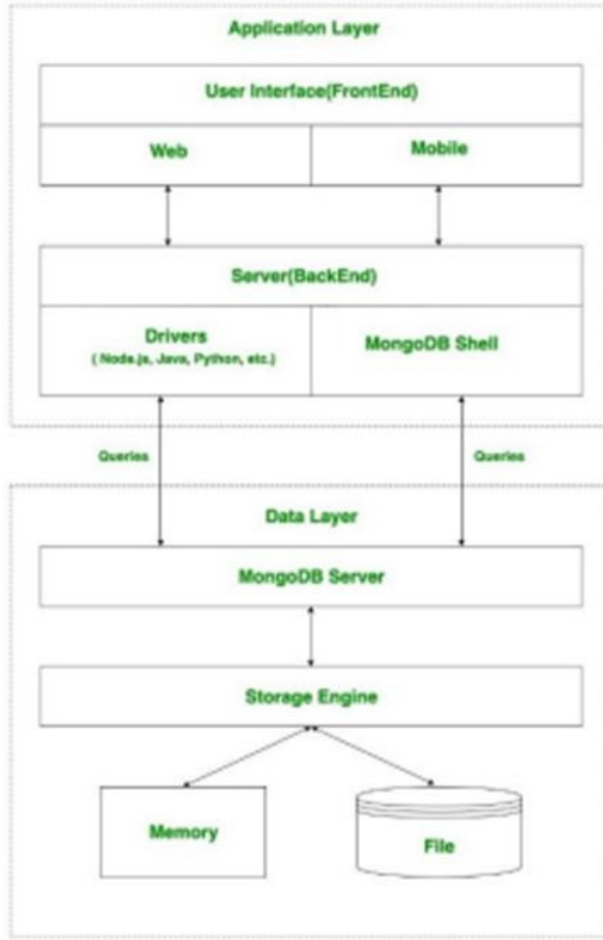
edilen veri yapısını kullanır.

MongoDB'nin bazı önemli özellikleri şunlardır:

- Esnek veri modeli: MongoDB, şema tabanlı bir veri modeline ihtiyaç duymaz. Veriler, belirli bir şema veya tabloya bağlamak zorunda kalmadan depolanır. Bu, veri modelinin değiştirilmesinde kolaylık sağlar.
- Yüksek performans: MongoDB, hızlı okuma ve yazma işlemleri sunar. Veriler, belgeler şeklinde depolandığından, işlemler genellikle daha hızlı gerçekleşir. Ayrıca, MongoDB, dağıtık sistemlerde ölçeklenebilirlik sağlayan yüksek kullanılabilirlik ve otomatik veri parçalama (sharding) özelliklerine sahiptir.
- Yüksek kullanılabilirlik: MongoDB, verilerin yedeklendiği ve gerektiğinde otomatik olarak geri alındığı yüksek kullanılabilirlik özelliklerine sahiptir. Bu sayede sistemde oluşabilecek kesinti veya hata durumlarında veri kaybı yaşanmaz.
- Veritabanı sorgulama dili (query language): MongoDB, zengin bir sorgulama diline sahiptir. Bu dil, belgeler arasında karmaşık sorgular yapmanızı ve verileri esnek bir şekilde filtrelemenizi sağlar.

MongoDB'nin yaygın kullanım alanları arasında web uygulamaları, büyük veri analitiği, gerçek zamanlı analiz, içerik yönetimi sistemleri ve mobil uygulamalar bulunur.

Projede mongoDB ile bağlantılı işlemler pymongo kütüphanesi kullanılarak yapılmıştır.



Şekil 3-2 MongoDB çalışma mantığı[18]

3.5. MongoDB Compass

MongoDB Compass, MongoDB'nin resmi görsel kullanıcı arayüzü (GUI) istemcisidir. MongoDB veritabanı oluşturmak, veri tabanını keşfetmek, sorgulamak, veri yönetimi yapmak ve veri analizi yapmak için kullanılan bir araçtır.[19]

MongoDB Compass, kullanıcıların MongoDB veritabanlarına grafiksel bir arayüz üzerinden erişmelerini sağlar. Aşağıdaki özelliklere sahiptir:

- Keşfetme: MongoDB Compass, veritabanındaki koleksiyonları ve dokümanları listelemektedir. Koleksiyonlar, şemaları ve ilişkileriyle birlikte kullanıcıya sunulmaktadır.

- Sorgulama: MongoDB Compass, veritabanındaki verileri sorgulamak için kullanılabilecek bir sorgu editörü sunar. Bu editör, kolay kullanım için sorgu oluşturma ve düzenleme özelliklerine sahiptir.
- Veri düzenleme: Compass, veri tabanındaki dokümanların düzenlenmesini sağlar. Dokümanlar görüntülenebilmekte, düzenleme yapılabilmekte ve silinebilmektedir.
- İndeksleme: MongoDB Compass, veritabanındaki indekslerin görüntülenmesini ve yeni indeksler oluşturulabilmesini sağlar. Bu, veritabanının performansını artırmak için önemli bir özelliktir.
- Veri analizi: Compass, veritabanındaki verileri görselleştirmek ve analiz etmek için grafiksel araçlar sunar. Grafikler, tablolar ve haritalar kullanarak verilerin daha iyi anlaşılmasını sağlar.

3.6. Csv Uzantılı Dosya

CSV (Comma-Separated Values), virgülle ayrılan değerler anlamına gelen bir dosya biçimidir. CSV dosyaları, tablo benzeri verileri saklamak için kullanılır ve metin tabanlıdır. Her satır, virgül veya başka bir ayırıcı karakterle ayrılmış sütunlardan oluşur.[20]

CSV dosyaları genellikle veri alışverişi veya veri depolama amacıyla kullanılır. Aşağıda CSV dosyalarının bazı yaygın kullanım alanları bulunmaktadır:

- Veri aktarımı: CSV dosyaları, farklı sistemler veya uygulamalar arasında veri aktarımında kullanılır. Örneğin, bir veri tabanından diğerine veri taşımak veya bir tablodan başka bir uygulamaya veri aktarmak için CSV dosyaları kullanılabilir. Bu dosya biçimi, veri kaynaklarının farklı formatlarda olması durumunda kolaylıkla kullanılabilir.
- Veri analizi ve raporlama: CSV dosyaları, veri analizi ve raporlama süreçlerinde yaygın olarak kullanılır. Verilerin belirli sütunlarında bulunan değerler üzerinde hesaplamalar yapmak, filtrelemeler uygulamak veya grafikler oluşturmak için CSV dosyaları kullanılabilir.

- Web uygulamaları: CSV dosyaları, web uygulamalarıyla veri alışverişinde kullanılabilir. Örneğin, kullanıcıların bir web sitesine yüklediği veya indirdiği veriler CSV formatında olabilir. Bu veriler daha sonra web uygulaması tarafından işlenerek gösterilebilir veya başka bir formata dönüştürülebilir.

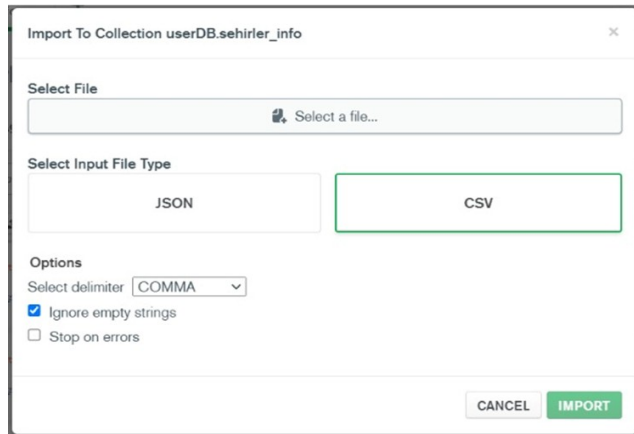
- Veri depolama: Verilerin depolanması veya geçici olarak saklanması için CSV dosyaları kullanılabilir. CSV dosyaları basit ve hızlı bir veri depolama yöntemi sunar. CSV dosyaları, metin tabanlı olduklarından kolayca okunabilir ve düzenlenebilirler.

3.7. Şehirler.CSV Dosyasının Oluşturulması

ŞehirlerBolgeler.xlsx [21] dosyası içerisinde ŞehirlerBolgeler, Şehirler, Bolgeler olmak üzere 3 sheet bulunmaktadır. Projede sadece ŞehirlerBolgeler sheeti kullanılmak üzere düzenlenmiştir. ŞehirlerBolgeler sheetinde Id, BolgeId, ŞehirAd, Nufus kolonları bulunmaktadır. ŞehirlerBolgeler sheeti sadece id ve şehirAd kolonlarını içerecek şekilde düzenlenmiştir. Xlsx uzantılı dosya düzenlendikten sonra .csv uzantılı dosyaya dönüştürülmüştür.

3.7.1. Şehirler.csv Dosyasının MongoDB Veri tabanına Import Edilmesi

.csv uzantılı olarak dönüştürülen ŞehirlerBolgeler dosyası mongoDB Compass arayüzü kullanılarak import edilmiştir. Dosya türü CSV olarak seçilmiştir. Kolon ayırıcı olarak ‘,’ seçeneği tercih edilmiştir. Boş gelen stringlerin göz ardı edilmesi seçeneği seçilmiştir.



Şekil 3-3 MongoDB Compass import ekranı

Projede mongoDB veritabanı bağlantısı pymongo kütüphanesi kullanılarak sağlanmıştır. pymongo modülü içe aktarılır ve MongoClient sınıfı kullanmak için gerekli bir bağlantı oluşturulur. MongoDB içerisinde bulunan veritabanlarına bağlanmak için myclient nesnesi kullanılır. Bağlantıyı kullanmak için ilgili veritabanlarının mevcut olması gerekmektedir. Eğer ilgili isimlerdeki veritabanları mevcut değil ise, belirtilen isimde yeni bir veritabanı oluşturulur. Belirli bir veritabanı içindeki koleksiyonlara erişmek için bu isimler kullanılmaktadır.

```
myclient = pymongo.MongoClient("mongodb://localhost:27017/")
mydb_login = myclient["userDB"]
mydb_sefer = myclient["busDb"]
mycollection_login = mydb_login["user_info"]
mycollection_sefer = mydb_sefer["tickets"]
mycollection_sehirler = mydb_login["sehirler_info"]
...
```

Şekil 3-4 MongoDB bağlantısı

3.8. Kullanılan Algoritmalar

3.8.1. Login ve Registration

Login ve registration ekranı, bir web sitesi veya mobil uygulamada kullanıcıların hesap oluşturabildiği ve mevcut hesaplarıyla giriş yapabileceği arayüzlerdir. Login ekranı, kullanıcı adı ve şifre gibi kimlik bilgilerini girmek için tasarlanmıştır. Bu bilgiler doğru ise, kullanıcı sisteme giriş yapabilmektedir. Registration ekranı ise yeni kullanıcıların kaydolabileceği bir form veya süreç sunmaktadır. Kullanıcılar, ad, e-posta, şifre gibi bilgileri girerek yeni bir hesap oluşturabilirler. Bu ekranlar, kullanıcı kimlik doğrulama ve yetkilendirme işlemlerini gerçekleştirmek için kullanılır.

3.8.1.1. Login ve Registration Ekranları Tasarımı

Projede templates klasörü içerisinde bulunan jinja kütüphanesi kullanılarak oluşturulan login.html dosyası login ve registration sayfalarının şablonunu içermektedir.

Login.html dosyası içerisinde html kodları, sayfa dizaynı için css kodları ve hareketli giriş sistemi için login ve registration ekranları arasında kullanıcı tercihiine göre geçiş

sağlayacak javascript kodları yer almaktadır.

Projede hem login hem registration ekranı tek bir sayfada gösterildiği için kullanıcının hangi tercihte bulunacağına göre kodun o form için çalışması gerekmektedir. Kodun hangi form için çalışacağı html içerisinde `<form action= “{{url_for(‘form adı ’)}}”>` kodu ile belirtilmiştir. Aşağıdaki kod örneği register işlemi için gerçekleştirilmiştir. Aynı formatta login işlemi için de routing işlemi `<form action= “{{url_for(‘form adı ’)}}”>` kodu ile gerçekleştirilmiştir.

```
<div class="form-container register-container">
  <form action="{{ url_for('register') }}" method="POST" name="form_kaydol">
    <h1>KAYDOLUN</h1>
    <input type="text" placeholder="İsim" value="isim" name="isim">
    <input type="email" placeholder="Email" value="email" name="email">
    <input type="password" placeholder="Şifre" value="sifre" name="sifre">
    <button type="submit">KAYDOL</button>
  </form>
</div>
```

Şekil 3-5 Register html kodu

Projede kullanıcının login ve registration ekranları arasında tercih yapabilmesi için ekranlar arası geçiş sağlayan javascript kodu yazılmıştır.

```
<script>
const registerButton = document.getElementById("register");
const loginButton = document.getElementById("login");
const container = document.getElementById("container");

registerButton.addEventListener("click", () => {
  container.classList.add("right-panel-active");
});

loginButton.addEventListener("click", () => {
  container.classList.remove("right-panel-active");
});
</script>
```

Şekil 3-6 Geçiş sağlayan javascript kodu

Kullanıcı registration ekranında bulunan formu doldurup kaydol butonuna tıkladığında web uygulamasında post isteği alınır. Form üzerinden `request.form['alınacak_değer']` kod parçacığıyla çekilen email ve şifre değerleri alınır. Mail adresinin veri tabanında olup olmadığının kontrolü

mycollection_login.find_one({'email': email}) kod parçacığı ile sağlanır. Girilen mail adresinin veri tabanında ekli olmaması durumunda formdan alınan mail ve şifre değerleri veri tabanına kaydedilir.

```
mycollection_login.insert_one({ 'email': email, 'sifre': sifre })
```

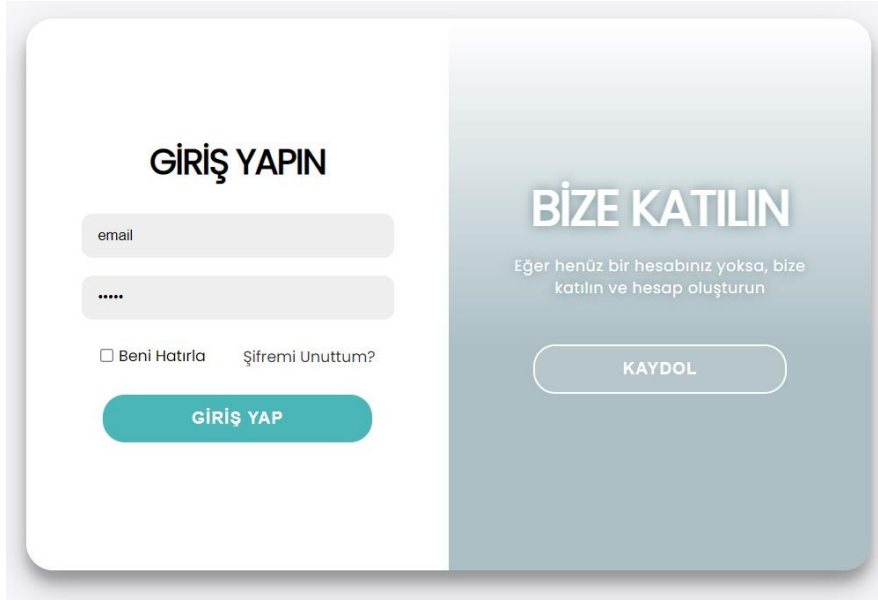
Şekil 3-7 MongoDB’ye kaydetme kodu

Kullanıcı login ekranında bulunan formu doldurup giriş yap butonuna tıkladığında web uygulamasında post isteği alınır. Form üzerinden request.form[‘girilen_deger’] kod parçacığıyla girilen email ve şifre değerleri alınır. Girilen email ve şifre değerlerinin veri tabanında olup olmadığının kontrolü aşağıdaki kod parçacığı ile sağlanır.

```
mycollection_login.find_one({ 'email': email, 'sifre': sifre })
```

Şekil 3-8 MongoDB’den veri arama kodu

Girilen mail adresi ve şifrenin veri tabanında ekli olması durumunda formda girilen değerler ile kullanıcı girişi sağlanır ve kullanıcı bilet arama ekranına yönlendirilir.

The image shows a web form with two main sections. The left section is titled 'GİRİŞ YAPIN' (Login) and contains an 'email' input field, a password input field with masked characters '.....', a checkbox labeled 'Beni Hatırla' (Remember me), a link 'Şifremi Unuttum?' (Forgot my password?), and a teal 'GİRİŞ YAP' (Login) button. The right section is titled 'BİZE KATILIN' (Join Us) and contains the text 'Eğer henüz bir hesabınız yoksa, bize katılın ve hesap oluşturun' (If you don't have an account yet, join us and create an account) and a 'KAYDOL' (Sign Up) button.

Şekil 3-9 Login ekranı

3.8.2. Bilet Seçim Ekranı

Kullanıcı girişi sağlandıktan sonra kullanıcı, bilet seçim ekranına yönlendirilir. Bilet seçim ekranında sefer kalkış yeri, varış yeri ve sefer tarihi bilgilerinin seçilebileceği dropdownlar bulunmaktadır. Kullanıcı bu ekranda kalkış yerini, varış yerini ve seyahat edeceği tarihi seçerek biletleri aratmaktadır.

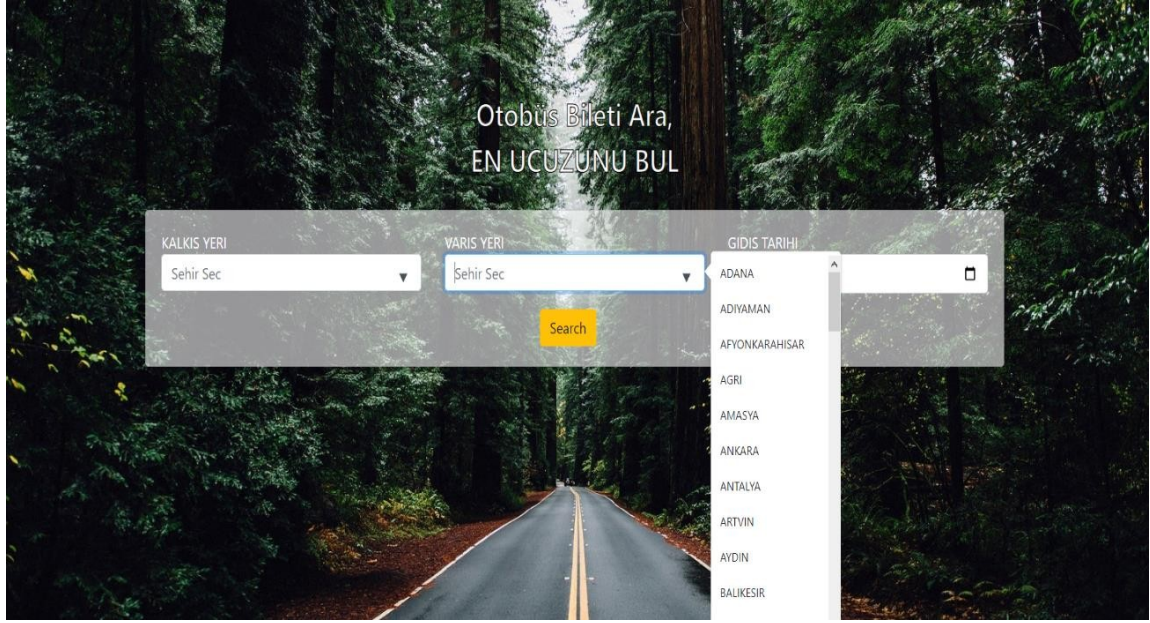
3.8.2.1. Dropdownlara Veri Çekilmesi

Daha önce ŞehirlerBolgeler.csv dosyası mongoDB veri tabanına import edilmişti. Import edilen bu veriler şehir isimlerinden oluşmaktadır. Kalkış yeri ve varış yeri için oluşturulan dropdownlara buradaki veriler GET_SEHIRLER() fonksiyonu ile şehirler=list(mycollection_sehirler.find({}, {'_id': 0,'Id':0})) kod parçasığı ile veri tabanından çekilmektedir. Çekilen veriler templates klasörü altında bulunan index.html dosyasına aşağıdaki kod parçasığı ile dropdownlara eklenmektedir.

```
<datalist id="datalistOptions">
  {% for i in range(sehirlerLen) %}
  <option value={{sehirler[i]}} SELECTED>{{sehirler[i]}}</option>
  {% endfor %}
```

Şekil 3-10 Şehirlerin aktarıldığı html kodu

Kullanıcı arama yaptıktan sonra tekrar arama yapmak istediğinde geri butonu ile tekrar bilet seçim ekranına gelebilmektedir. Index.html sayfası her yüklendiğinde veri tabanında bulunan veriler sıfırlanmaktadır. Böylelikle web sitesinin dinamik olarak anlık alınan veriler ile çalışması sağlanmış olmaktadır. Kullanıcı arama butonuna tıkladığında post metodu çalışarak kullanıcının girdiği bilgiler ile arka planda web kazıma işlemleri gerçekleştirilmektedir. Web kazıma işlemi ilerleyen sayfalarda detaylandırılacaktır. Web kazıma işlemi sonrası seyahat firmalarından çekilen veriler biletsonuc.html sayfasında kullanıcıya sunulacaktır.



Şekil 3-11 Bilet arama ekranı

3.8.2.2. Web Kazıma İşlemleri

Web kazıma (web scraping), web sitelerinden veri toplama işlemidir. Kullanıcının aradığı verilere uygun bileti diğer seyahat sitelerinden arayarak biletlere dair bilgilerin toplanması sağlanmaktadır. Toplanan veriler ile farklı sitelerdeki biletlerin kullanıcıya bir web sitesinde gösterilmesi sağlanmaktadır. Web kazıma işlemi için selenium kütüphanesi kullanılmaktadır. Farklı web sitelerinden veri çekerken web sitelerine sürekli request atıldığı için ban yeme problemi ile karşılaşılması için undetected_chromedriver kütüphanesi kullanılmaktadır. Flask ve undetected_chromedriver aynı anda çalıştığında iki farklı serverda başlatıldığından dolayı server hatası verecektir. Bu problemle karşılaşmamak için undetected_chromedriver ın seçeneklerini değiştirmek gerekmektedir. Driverın seçeneklerini değiştirmek için selenium.webdriver.chrome.options kütüphanesi kullanılmaktadır. Veri çekilecek olan her bir web sitesinin html şablonu ve verileri tutuş şekli birbirinden farklı olduğu için çekilecek her bir web sitesi için farklı bir web kazıma algoritması geliştirilmiştir. Geliştirilen algoritmaların temelinde bilet arama ekranından alınan kalkış yeri, varış yeri ve tarih verileri ile bir url oluşturulması ve bu url üzerinden url'e ait olan web sitesinin html şablonundan biletlere ait verilerin çekilmesi vardır. Algoritmalarla dair detaylı bilgi ilerleyen sayfalarda verilecektir.

3.8.2.3. Driverın Özelliklerinin Değiştirilmesi

Universal Chrome(uc) tarayıcısı kullanılmaktadır. Chrome tarayıcısı için bazı yapılandırma seçeneklerinde ayarlamalar yapılmıştır. uc.ChromeOptions() komutu ile Chrome tarayıcısı için yapılandırma seçeneklerini ayarlamak için bir ChromeOptions nesnesi oluşturulur. uc.Chrome(options=chrome_options) komutu, yapılandırma seçeneklerini kullanarak Universal Chrome tarayıcısı üzerinde otomasyon yapmak için bir Chrome nesnesi oluşturur. Bu nesne, web kazıma görevlerini gerçekleştirmek için kullanılmaktadır.

- chrome_options.add_argument("--no-sandbox") komutu, Chrome tarayıcısının güvenlik önlemlerini kısmen devre dışı bırakmaktadır. Kullanıcı izinleriyle ilgili sorunlara neden olabilecek bazı sınırlamaları kaldırır.
- chrome_options.add_argument("--disable-dev-shm-usage") komutu, tarayıcının paylaşılan bellek bölgesini devre dışı bırakır ve bellek kullanımını azaltır. Bu, tarayıcının daha hızlı çalışmasını sağlamaktadır.
- chrome_options.add_argument("--start-maximized") komutu, tarayıcının başlangıçta tam ekran olarak açılmasını sağlar. Ekranın tam ekran olarak açılması tarama yapılan sitelerin daha detaylı görünmesini sağlamaktadır.
- chrome_options.add_argument('--disable-popup-blocking') komutu, bir web sitesine girildiğinde çıkan açılır pencerelerin (pop-up) engellenmeden açılmasını sağlamaktadır. Böylelikle açılan herhangi bir açılır pencereye engel olmamaktadır.

```
chrome_options = uc.ChromeOptions()
chrome_options.add_argument("--no-sandbox")
chrome_options.add_argument("--disable-dev-shm-usage")
chrome_options.add_argument("--start-maximized")
chrome_options.add_argument('--disable-popup-blocking')
driver = uc.Chrome(options=chrome_options)
```

Şekil 3-12 Driverın optionslarının değiştirilme kodu

3.8.2.4. Web Kazıma İle Veri Çekme

Chrome tarayıcısı için bazı yapılandırma seçeneklerinde ayarlamalar yapılmıştır.

Firmalara ait genel bir url alınmıştır. Bu url’de kalkış yeri, varış yeri ve bilet tarihi gibi değişkenler bilet arama ekranından alınacak şekilde parametrik olarak düzenlenmiştir. Url Universal Chrome tarayıcısında açılmaktadır ve web sitesine yönlendirilmektedir. Web sitesinin tamamen yüklenmesi ve gereken verilerin eksiksiz bir şekilde alınabilmesi için bir süre kodun uyku moduna geçmesi sağlanmıştır.

Metro Firmasından Web Kazıma İle Veri Çekme

Parametrik olan düzenlenen url kodu aşağıdaki gibidir.

```
url = 'https://www.metroturizm.com.tr/otobus-bileti/{kalkisyeri}-{varisyeri}?date={tarihsaat}&tw=0'
```

Şekil 3-13 Metro firması url’i

Metro firmasına ait web sitesinin sayfa kaynağı görüntülenerek çekilecek olan verilerin html şablonunda hangi class altında olduğu incelenmiştir. Çekilecek olan veriye ait xpath bulunmuştur. Bulunan xpath ifadesine göre html elementlerinin bulunması sağlanmaktadır.

```
metro_kalkis = driver.find_elements("xpath", '//*[@class="start ng-binding"]')
```

Şekil 3-14 Metro firması kalkış yeri çekme kodu

find_elements yöntemi kullanılarak XPath ifadesine göre HTML elementleri bulunmaktadır. Bu kod parçacığı, elementleri arasında start sınıfına ve ng-binding özniteliğine sahip olan verileri seçmektedir. Bu xpath ifadesi metro firmasının sayfası incelendiğinde web sitesindeki biletlerin kalkış yeri değişkenine denk gelmektedir. Çekilen veriler kullanıcıya sunulmak üzere lst_kalkis_yeri isimli bir liste içinde tutulmaktadır.

Yukarıda kullanılan yöntem kullanıcıya sunulmak istenen her bir değişken için kullanılmaktadır. Aşağıdaki xpath ifadesi metro firmasının sayfası incelendiğinde web sitesindeki biletlerin kalkış ve varış saatleri değişkenine denk gelmektedir. Çekilen veriler kullanıcıya sunulmak üzere lst_kalkis_saati isimli bir liste içinde tutulmaktadır. Kalkış ve varış saatleri aynı xpath ile çekildiği için çekilen verilerin sadece kalkış saati olacak şekilde düzenlenmesi gerekmektedir. Bu düzenleme için

aşağıdaki kod parçacığı kullanılmıştır. Bu kod parçacığı ile liste dilimleme yöntemi kullanılarak [::2] ifadesi, listenin 0. indeksten başlayarak her iki adımda bir eleman alması sağlanmaktadır. Yani, listenin 0, 2, 4, 6, ... gibi çift indeksli elemanları yeni bir liste içine atılarak sadece kalkış saatlerini içeren bir liste elde edilmiştir.

```
metro_saat = driver.find_elements("xpath", '//*[@class="journey-item-hour ng-binding"]')
for saatIndex in metro_saat:
    if(saatIndex.text!=""):
        lst_kalkis_saati.append(saatIndex.text)
        lst_metro_saat2=lst_kalkis_saati[::2]
```

Şekil 3-15 Metro firması kalkış saati çekme kodu

Aşağıdaki xpath ifadesi metro firmasının sayfası incelendiğinde web sitesindeki biletlerin varış yeri değişkenine denk gelmektedir. Çekilen veriler kullanıcıya sunulmak üzere lst_varis_saati isimli bir liste içinde tutulmaktadır.

```
metro_varis = driver.find_elements("xpath", '//*[@class="end ng-binding"]')
```

Şekil 3-16 Metro firması varış yeri çekme kodu

Aşağıdaki xpath ifadesi metro firmasının sayfası incelendiğinde web sitesindeki biletlerin bilet fiyat değişkenine denk gelmektedir. Çekilen veriler kullanıcıya sunulmak üzere lst_fiyat isimli bir liste içinde tutulmaktadır.

```
metro_fiyat = driver.find_elements("xpath", '//*[@class="price ng-binding"]')
```

Şekil 3-17 Metro firması bilet fiyatı çekme kodu

Verilerin kullanıcıya sunulurken html şablonuna aktarımının daha kolay olması ve verinin daha düzenli bir formatta olması için tüm listeler sefer_lst isimli bir liste içerisinde toplanmıştır. Web kazıma ile elde edilen tüm veriler mongoDB veritabanına kaydedilmiştir.

Varan Firmasından Web Kazıma İle Veri Çekme

Parametrik olarak düzenlenen url kodu aşağıdaki gibidir.

```
url = "https://www.varan.com.tr/otobus-bilet1/{KalkisYeri}/{VarisYeri}/{Tarihsaat}?rezultat={rezultat}"
```

Şekil 3-18 Varan firması url'i

Metro firmasına ait web sitesinin sayfa kaynağı görüntülenerek çekilecek olan verilerin html şablonunda hangi class altında olduğu incelenmiştir. Çekilecek olan veriye ait xpath bulunmuştur. Bulunan xpath ifadesine göre html elementlerinin bulunması sağlanmaktadır. Metro turizm için yapılmakta olan tüm adımlar Varan firması için de yapılmaktadır. Tek farkı Varan firmasının web sitesinden kalkış saati tek seferde çekilebilmektedir. Bu yüzden aşağıdaki kod parçasığı bu işlem için yeterlidir.

```
varan_saat = driver.find_elements("xpath", '//*[@class="time"]')
for saatIndex in varan_saat:
    if(saatIndex.text!=""):
        lst_kalkis_saati.append(saatIndex.text)
```

Şekil 3-19 Varan firması kalkış saati çekme kodu

Bilet Sonuç Ekranı

Kullanıcının bilet arama sayfasında tercih ettiği seferlere ait biletlerin kullanıcıya sunulduğu ekrandır. Bir tablo şeklinde biletler sıralanmaktadır. Biletlere ait firmaların logoları, bilete ait kalkış yeri, varış yeri, kalkış saati, tarihi ve bilet fiyatı bilgileri kullanıcıya sunulmaktadır. Kullanıcının isteğine göre fiyata göre artan veya fiyata göre azalan bir sıralama tercih etmesi durumu için bir filtreleme seçeneği oluşturulmuştur. Kullanıcı sıralama tercihi yapmadığı sürece listelenen biletler random olarak kullanıcıya sunulmaktadır. Kullanıcı bir filtreleme işlemi yapar ise tercih ettiği filtreye göre veriler mongoDB den alınıp sıralanmaktadır. Kullanıcı fiyata göre artan seçeneğini tercih ederse aşağıdaki kod parçasığı ile sıralama işlemi gerçekleştirilmektedir.

```
sort=list(mycollection_sefer.find({}, {'_id': 0}).sort("fiyat",1))
```

Şekil 3-20 Bilet fiyatlarının artan olarak sıralanma kodu

Kullanıcı fiyata göre azalan seçeneğini tercih ederse aşağıdaki kod parçasığı ile

sıralama işlemi gerçekleştirilmektedir.

```
sort=list(mycollection_sefer.find({}, {'_id': 0}).sort("fiyat",-1))
```

Şekil 3-21 Bilet fiyatlarının azalan olarak sıralanma kodu

Veriler sefer_lst içerisine atılarak template klasörü içerisindeki biletSonuc.html şablonuna render_template() ile gönderilmektedir. Bu işlemlere ait örnek bir kod parçası aşağıda verilmiştir.

```
filtre = request.form['filtre']
if filtre == "artan":
    sefer_lst.clear()

    sort=list(mycollection_sefer.find({}, {'_id': 0}).sort("fiyat",1))
    for i in sort:
        sefer = list(i.values()) # Sadece değerleri al
        for value in sefer:
            sefer_lst.append(value)
    return render_template("biletSonuc.html",sefer_lst=sefer_lst,sortdatalen=len(sefer_lst))
```

Şekil 3-22 Bilet sonuç ekranına yönlendirme kodu

BiletSonuc.html sayfasında backend'ten yollanan sefer bilgilerini içeren sefer_lst ve bu listenin uzunluğunu veren sortdatalen değişkenini kullanarak sefer sonuçlarını kullanıcıya gösteren bir tablo oluşturulmuştur. For döngüsünün sekizer artarak liste uzunluğu kadar dönmesi sağlanmıştır. Sekizer artış olmasının sebebi tek döngüde biletin ait olduğu firmanın logosu, firma adı, kalkış yeri, varış yeri, bilet tarihi, bilet saati, bilet fiyatı ve satın alma butonu olmak üzere sekiz değişken bir bilet için tek satırda ekranda gösterilmektedir. Bilete ait firmanın logosu static klasörü içerisinde bulunmaktadır. Jpg, png vb. uzantılı resimlerin html şablonu içerisinde gösterilebilmesi için static isimli klasör içerisinde bulunması gerekmektedir.
 <div class="info">
 <label> </label>
 <label>{{sefer_lst[i+1]}}</label>
 <label>{{sefer_lst[i+2]}}</label>
 <label>{{sefer_lst[i+3]}}</label>
 <label>{{sefer_lst[i+4]}}</label>
 <label>{{sefer_lst[i+5]}}</label>
 <label>{{sefer_lst[i+6]}}</label>
 </div>
 Satın Al
</div>
</section>
{% endfor %}
```








Şekil 3-23 Bilet sonuçlarının html'e aktarılma kodu

Listelenen biletlerin satın alınabilmesi işlemi için her bir bilet için satın al butonu oluşturulmuştur. Satın alınmak istenen bilete ait satın al butonuna tıklanıldığında kullanıcı bilete ait firmanın bilet satış sitesine yönlendirilmektedir.

Bilet Arama Sonuçları							
	FİRMA	KALKIŞ	VARIŞ	TARİH	SAAT	FİYAT	
	pamukkale	İZMİR	BURSA	2023-05-25	04:00	200 TL	<a href="#">Satın Al</a>
	pamukkale	İZMİR	BURSA	2023-05-25	07:00	210 TL	<a href="#">Satın Al</a>
	pamukkale	İZMİR	BURSA	2023-05-25	09:00	200 TL	<a href="#">Satın Al</a>
	pamukkale	İZMİR	BURSA	2023-05-25	10:00	210 TL	<a href="#">Satın Al</a>
	pamukkale	İZMİR	BURSA	2023-05-25	11:00	210 TL	<a href="#">Satın Al</a>
	pamukkale	İZMİR	BURSA	2023-05-25	11:30	200 TL	<a href="#">Satın Al</a>
	pamukkale	İZMİR	BURSA	2023-05-25	12:00	200 TL	<a href="#">Satın Al</a>



Şekil 3-24 Bilet sonuç ekranı

Bir filtreleme seçeneği tercih edildiğinde kullanıcıya sunulacak olan örnek bilet sonuç ekranı aşağıdaki gibidir.

	varan	İzmir	Bursa	2023-05-25	23:00	160₺ Web'e Özel Fiyat	<a href="#">Satın Al</a>
	varan	İzmir	Bursa	2023-05-25	23:30	160₺ Web'e Özel Fiyat	<a href="#">Satın Al</a>
	varan	İzmir	Bursa	2023-05-25	23:59	160₺ Web'e Özel Fiyat	<a href="#">Satın Al</a>
	varan	İzmir	Bursa	2023-05-25	01:00	160₺ Web'e Özel Fiyat	<a href="#">Satın Al</a>
	metro	İZMİR	BURSA	2023-05-25	10:00	170	<a href="#">Satın Al</a>
	metro	İZMİR	BURSA	2023-05-25	11:00	170	<a href="#">Satın Al</a>
	metro	İZMİR	BURSA	2023-05-25	12:00	170	<a href="#">Satın Al</a>

Şekil 3-25 Bilet sonuç ekranı artan fiyata göre sıralanması

Yukarıda listelenen biletlerden herhangi birinin satın al butonuna tıklandığında o bilet firmasına ait web sitesine yönlendirilmesinin örnek durumu aşağıdaki gibidir.


İzmir > Bursa
25 Mayıs Perşembe 2023


25.05.2023 Perşembe

26.05.2023 Cuma

27.05.2023 Cumartesi

09:00	İzmir Bursa	2+1	160₺ Web'e Özel Fiyat	<a href="#">KOLTUK SEÇ</a>
11:00	İzmir Bursa	2+1	160₺ Web'e Özel Fiyat	<a href="#">KOLTUK SEÇ</a>
12:00	İzmir Bursa	2+1	160₺ Web'e Özel Fiyat	<a href="#">KOLTUK SEÇ</a>
13:00	İzmir Bursa	2+1	190₺ Web'e Özel Fiyat	<a href="#">KOLTUK SEÇ</a>

Şekil 3-26 Bilet al seçeneğine tıklandıktan sonra yönlendirilen ekran

#### 4.Sonuçlar ve Öneriler

Bir yere seyahat edileceğinde gitmek istenilen şehire otobüs seferi düzenleyen birçok bilet firması bulunmaktadır. Her bir bilet firmasının web sitesine girip tek tek gidilecek yere ait seferleri aratmak oldukça zordur. Bu işlem kullanıcının vaktini olması gerekenden daha uzun süre almaktadır. Harcanan süreye kıyasla en ideal biletin bulunmasının da garantisi yoktur. Dolayısıyla bu yöntem optimize bir çözüm değildir. Bu sorunu çözebilmek ve kullanıcının bilet ararken kaybettiği zamanı geri kazandırmak için birçok bilet firmasına ait seferlerin tek bir web sitesi kullanılarak kullanıcıya sunulması çözümünü üreten bir web sitesi tasarlanmıştır. Kullanıcının aratmak istediği seferleri seçebileceği bir responsive ekran ile kullanıcının bilet araması yapması sağlanmış ve aratılan bilgilere ait seferler birkaç web sitesinden web kazıma yöntemi ile çekilerek tek bir sayfada kullanıcıya sunulmuştur. Kullanıcının yüzlerce bilet arasından en ucuzunu aramasının da bir vakit kaybı yaratacağı düşünülerek kullanıcının biletleri ucuzdan pahalıya veya pahalıdan ucuza olacak şekilde sıralayabileceği bir filtreleme yöntemi geliştirilmiştir. Geliştirilen algoritmalar sonucunda kullanıcı aradığı sefere ait biletleri tek bir sayfada görebilmekte ve fiyatına göre filtreleyebilmektedir. Bu algoritmaların geliştirilmesi sonucunda ortaya çıkardığımız web sitesi kullanıcı dostu bir site olarak ortaya çıkmıştır. Kullanıcının vakit kaybı yaşaması engellenmiş ve kullanıcıya en uygun biletin sunulması sağlanmıştır.

## Kaynakça

- [1] URL-1: <https://unalarif.com/yazi/online-alisveris-nedir/>,  
(Ziyaret tarihi: 14 Ekim 2022).
- [2] URL-2: <https://www.metroturizm.com.tr/>, (14 Ekim 2022)
- [3] URL-3: [https://www.metroturizm.com.tr/otobus-bileti/ISTANBUL\\_AND-ANKARA?date=26.05.2023&tw=0](https://www.metroturizm.com.tr/otobus-bileti/ISTANBUL_AND-ANKARA?date=26.05.2023&tw=0), (Ziyaret tarihi: 14 Ekim 2022)
- [4] URL-4: <https://www.obilet.com/>, (Ziyaret tarihi: 14 Ekim 2022)
- [5] URL-5: <https://www.obilet.com/seferler/349-383/2023-05-27>,  
(Ziyaret tarihi: 14 Ekim 2022)
- [6] URL-6: <https://www.mysoft.com.tr/buyuk-veri-big-data>,  
(Ziyaret tarihi: 27 Aralık 2022)
- [7] URL-7: <https://www.webharvy.com/articles/what-is-web-scraping.html>,  
(Ziyaret tarihi: 23 Şubat 2023)
- [8] URL-8: <https://docs.python-requests.org/en/latest/>, (Ziyaret tarihi: 23 Şubat 2023)
- [9] URL-9: <https://www.crummy.com/software/BeautifulSoup/bs4/doc/>,  
(Ziyaret tarihi: 24 Şubat 2023)
- [10] URL-10: <https://docs.scrapy.org/>, (Ziyaret tarihi: 25 Şubat 2023)
- [11] URL-11: <https://docs.scrapy.org/en/latest/topics/selectors.html>,  
(Ziyaret tarihi: 25 Şubat 2023)
- [12] URL-12: <https://www.turkhackteam.org/konular/osint-spiderfoot.2030781/>, (Ziyaret tarihi: 02 Mart 2023)
- [13] URL-13: <https://developers.google.com/web/updates/2017/04/headless-chrome>,  
(Ziyaret tarihi: 19 Mart 2023)
- [14] URL-14: <https://www.selenium.dev/selenium/docs/api/py/>,  
(Ziyaret tarihi: 28 Mart 2023)
- [15] URL-15: <https://selenium-python.readthedocs.io/>, (Ziyaret tarihi: 28 Mart 2023)
- [16] URL-16: <https://www.zesty.io/mindshare/marketing-technology/dynamic-vs-static-websites/>, (Ziyaret tarihi: 28 Mart 2023)
- [17] URL-17: [https://en.wikipedia.org/wiki/Flask\\_\(web\\_framework\)](https://en.wikipedia.org/wiki/Flask_(web_framework)),  
(Ziyaret tarihi: 28 Mart 2023)



- [18] URL-18: <https://selenium-python.readthedocs.io/>, (Ziyaret tarihi: 28 Mart 2023)
- [19] URL-19: <https://docs.mongodb.com/compass/current/>,  
(Ziyaret tarihi: 04 Nisan 2023)
- [20] URL-20: [https://en.wikipedia.org/wiki/Comma-separated\\_values](https://en.wikipedia.org/wiki/Comma-separated_values),  
(Ziyaret tarihi: 04 Nisan 2023)
- [21] URL-21: <https://github.com/yigith/TurkiyeSehirlerBolgeler/blob/master/excel/SehirlerBolgeler.xlsx>, (Ziyaret tarihi: 10 Nisan 2023)

## ÖZGEÇMİŞ

Sena Öktem, 2000 yılında İstanbul’da doğdu. Lise öğrenimini Gökkuşığı Kolejinde tamamladı. 2019 yılında girdiği Kocaeli Üniversitesi Bilgisayar Mühendisliği Bölümü’nde halen 4. sınıfta okumaktadır.

Mehmet Aygün, 2000 yılında Adana’da doğdu. Lise öğrenimini Çukurova Kolejinde tamamladı. 2018 yılında Kocaeli Üniversitesi Bilgisayar Mühendisliği Bölümü’nü kazandı. Şu an halen Kocaeli Üniversitesi Bilgisayar Mühendisliği Bölümü’nde 4. sınıfta okumaktadır.