Baykar Web Yazılım

Test Uzmanı Proje Dökümanı

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Version 1.0**

**İÇİNDEKİLER**

[1 Güncelleme Tarihçesi 4](#_Toc181568408)

[2 Giriş 5](#_Toc181568409)

[2.1 Manuel Testler: 5](#_Toc181568410)

[2.1.1 Test Senaryosu: Kariyer Sitesi Genel İşleyiş Testi 5](#_Toc181568411)

[2.1.2 Test Adımları 5](#_Toc181568412)

[2.2 Otomasyon Testleri 7](#_Toc181568413)

[2.3 Yük/ Performans Testleri: 10](#_Toc181568414)

[2.3.1 Python ve Locust araçları ile Baykar Kariyer(https://kariyer.baykartech.com/) web sitesi için yük ve performans testi senaryolarının yazılması. 10](#_Toc181568415)

[2.3.2 JMeter ile Baykar Kariyer(https://kariyer.baykartech.com/) web sitesi için yük ve performans test planı hazırlanması. 13](#_Toc181568416)

**ŞEKİLLER**

Şekil öğesi bulunamadı.

**TABLOLAR**

Tablosu öğesi bulunamadı.

# Güncelleme Tarihçesi

|  |  |  |  |
| --- | --- | --- | --- |
| Revizyon No | Değişiklik Sahibi | Değişiklik İçeriği | Tarih |
| 1.0 | Kamil İlhan | İlk Yazım | 02.11.2024 |

|  |  |  |
| --- | --- | --- |
| Revizyon No | Doküman Onayı | Tarih |
| 1.0 |  |  |

|  |  |  |
| --- | --- | --- |
| Revizyon No | Dağıtım | Tarih |
| 1.0 |  |  |

# Giriş

Bu doküman <https://kariyer.baykartech.com/> Web Sitesi içeriğinde yazılım test ve otomasyonun ilgili senaryolara istinaden hazırlanmıştır.

## Manuel Testler:

### Test Senaryosu: Kariyer Sitesi Genel İşleyiş Testi

Kariyer siteleri için hazırlanan test senaryoları, kullanıcıların kolaylıkla pozisyon arayabilmesi, başvuru yapabilmesi ve gerekli bilgileri hızlıca bulabilmesi için önemlidir. <https://kariyer.baykartech.com/> için genel bir test senaryosu:

### Test Adımları

##### Ana Sayfa Görüntüleme

• Adım: Kariyer sitesine ana sayfa URL’si ile erişim sağla.

• Beklenen Sonuç: Sayfa sorunsuz şekilde yüklenmeli, menüler ve görseller düzgün görünmeli.

• Kontrol Noktaları:

• Ana sayfa menüleri (Hakkında, Açık Pozisyonlar, İletişim, vb.) mevcut mu?

• “Açık Pozisyonlar” veya benzeri başlığa sahip bir sekme var mı?

##### Açık Pozisyonlar Sayfasına Erişim

• Adım: Ana sayfada bulunan “Açık Pozisyonlar” bağlantısına tıklayın.

• Beklenen Sonuç: Kullanıcı, tüm açık pozisyonların listelendiği sayfaya yönlendirilmeli.

• Kontrol Noktaları:

• Pozisyonların listesi düzgün görüntüleniyor mu?

• Filtreleme veya arama seçeneği var mı? (Varsa, düzgün çalıştığı doğrulanmalı)

##### Pozisyon Detay Görüntüleme

• Adım: Listelenen pozisyonlardan birine tıklayın.

• Beklenen Sonuç: Seçilen pozisyonun detayları (iş tanımı, gereklilikler, başvuru koşulları) eksiksiz şekilde gösterilmeli.

• Kontrol Noktaları:

• Pozisyon ismi, iş tanımı, gereklilikler gibi alanlar tam ve düzgün mü?

• Başvuru yap veya kaydet gibi butonlar görünür durumda mı?

##### Filtreleme ve Arama Fonksiyonu Testi

• Adım: Açık pozisyonlarda farklı filtre seçeneklerini kullanarak arama yapın (örneğin, lokasyon, departman, iş türü).

• Beklenen Sonuç: Filtreler doğru şekilde uygulanmalı ve ilgili pozisyonlar listelenmeli.

• Kontrol Noktaları:

• Her filtreleme sonrası liste güncelleniyor mu?

• Filtrelenen sonuçlar, seçilen kritere uygun mu?

##### Başvuru Süreci Testi

• Adım: Pozisyon detay sayfasında “Başvur” veya “Başvuru Yap” düğmesine tıklayın.

• Beklenen Sonuç: Kullanıcı başvuru formuna yönlendirilmelidir.

• Kontrol Noktaları:

• Başvuru formu eksiksiz olarak açılıyor mu?

• Zorunlu alanlar belirtilmiş mi? Eksik bilgiyle formu gönderme denendiğinde uyarı veriyor mu?

• Dosya yükleme alanı varsa CV yüklenebiliyor mu?

• Form gönderildiğinde “Başvuru alındı” gibi bir onay mesajı görüntüleniyor mu?

##### İletişim Sayfasına Erişim ve Testi

• Adım: Ana sayfada bulunan “İletişim” bağlantısına tıklayın.

• Beklenen Sonuç: İletişim sayfası sorunsuz yüklenmeli ve gerekli bilgiler (telefon, e-posta, adres) yer almalı.

• Kontrol Noktaları:

• İletişim bilgileri tam mı? (telefon, e-posta adresi vb.)

• Harita bileşeni varsa doğru lokasyonu gösteriyor mu?

##### Mobil Uyumluluk Testi

• Adım: Siteye mobil cihaz veya tarayıcı üzerinde mobil görünümde erişim sağla.

• Beklenen Sonuç: Site mobil uyumlu olmalı, menüler ve pozisyon listesi mobil görünümde düzgün çalışmalı.

• Kontrol Noktaları:

• Sayfa menüleri ve görseller mobil cihazda düzgün görünüyor mu?

• Açık pozisyonlara, detaylarına ve başvuru formuna mobil cihaz üzerinden sorunsuz erişilebiliyor mu?

##### Site Performans ve Hız Testi

• Adım: Site performans testi aracı (örneğin, Google PageSpeed Insights veya GTmetrix) ile hız testi yapın.

• Beklenen Sonuç: Site hızlı bir şekilde yüklenmeli ve kabul edilebilir bir performans puanına sahip olmalı.

• Kontrol Noktaları:

• Ana sayfa ve pozisyonlar sayfası hız testi sonuçları tatmin edici mi?

• Sayfa yükleme süresi kullanıcı deneyimini olumsuz etkilemeyecek düzeyde mi?

Bu senaryoların tamamlanmasıyla kariyer sitesinin kullanıcı deneyimi, erişilebilirlik, işlevsellik ve performans açısından yeterli düzeyde olup olmadığı değerlendirilebilir.

## Otomasyon Testleri

1. Navbar Elementleri Testi

**Senaryo:** Ziyaretçi <https://kariyer.baykartech.com/> sitesindeki bütün navbar elementlerine tıklayabilmeli ve sayfa sorunsuz açılmalı.

**Amaç:** Navbar'daki tüm elementlere tıklanabilmesi ve tıklanan her elementin doğru sayfaya yönlendirdiğinin kontrol edilmesi.

**Dinamik Yapı:** Navbar otomasyonu, yeni eklenen elementler için otomatik güncellenmelidir.

using OpenQA.Selenium;

using OpenQA.Selenium.Chrome;

using NUnit.Framework;

public class NavbarTests

{

private IWebDriver driver;

[SetUp]

public void Setup()

{

driver = new ChromeDriver();

driver.Navigate().GoToUrl("https://www.baykartech.com");

}

[Test]

public void TestNavbarElements()

{

var navbarElements = driver.FindElements(By.CssSelector("nav a"));

foreach (var element in navbarElements)

{

string originalWindow = driver.CurrentWindowHandle;

element.Click();

// Sayfanın yüklenmesini bekleyin

Assert.IsTrue(driver.Url.Contains(element.Text), $"{element.Text} sayfasına yönlendirilemedi.");

// Geri dön

driver.Navigate().Back();

driver.SwitchTo().Window(originalWindow);

}

}

[TearDown]

public void Teardown()

{

driver.Quit();

}

}

B. Dil Değiştirme Testi

**Senaryo:** Ziyaretçi <https://kariyer.baykartech.com/> sitesinde sağlanan diller arasında geçiş yapabilmeli, doğru dil gösterilmeli.

**Amaç:** Kullanıcı dil değiştirdiğinde sayfanın doğru dilde yüklenip yüklenmediğini kontrol etmek.

**Statik Yapı:** Desteklenen diller önceden tanımlanmalıdır.

using OpenQA.Selenium;

using OpenQA.Selenium.Chrome;

using NUnit.Framework;

public class LanguageSwitchTests

{

private IWebDriver driver;

[SetUp]

public void Setup()

{

driver = new ChromeDriver();

driver.Navigate().GoToUrl("https://www.baykartech.com");

}

[Test]

public void TestLanguageSwitch()

{

var supportedLanguages = new[] { "English", "Türkçe" }; // Desteklenen diller

foreach (var language in supportedLanguages)

{

var languageElement = driver.FindElement(By.XPath($"//a[contains(text(), '{language}')]"));

languageElement.Click();

// Dil değişikliğini kontrol et

Assert.IsTrue(driver.PageSource.Contains(language), $"{language} diline geçiş yapılamadı.");

}

}

[TearDown]

public void Teardown()

{

driver.Quit();

}

}

C. Pozisyon Arama ve Filtreleme Testi

**Senaryo:** Ziyaretçi/Kullanıcı <https://kariyer.baykartech.com/> sitesinde açık pozisyonlarda birim filtreleme ve pozisyon arama yapabilmeli.

**Amaç:** Kullanıcı, pozisyonlar üzerinde filtreleme yapabilmeli ve arama sonuçları ile ilgili verilerin eşleştiğini doğrulamalıdır.

**Dinamik Yapı:** Mevcut pozisyonlar dinamik olarak alınmalıdır.

using OpenQA.Selenium;

using OpenQA.Selenium.Chrome;

using NUnit.Framework;

public class JobSearchTests

{

private IWebDriver driver;

[SetUp]

public void Setup()

{

driver = new ChromeDriver();

driver.Navigate().GoToUrl("https://kariyer.baykartech.com");

}

[Test]

public void TestJobFilteringAndSearch()

{

var jobTitles = new[] { "Software Engineer", "Test Engineer" }; // Örnek pozisyon isimleri

foreach (var jobTitle in jobTitles)

{

var searchBox = driver.FindElement(By.Id("jobSearchInput"));

searchBox.Clear();

searchBox.SendKeys(jobTitle);

searchBox.Submit();

// Arama sonuçlarını kontrol et

var results = driver.FindElements(By.CssSelector(".job-title"));

Assert.IsTrue(results.Any(r => r.Text.Contains(jobTitle)), $"{jobTitle} için sonuç bulunamadı.");

}

}

[TearDown]

public void Teardown()

{

driver.Quit();

}

}

**Sonuç**

Yukarıdaki test senaryoları, <https://kariyer.baykartech.com/> web sitesinin farklı özelliklerini otomatikleştirmek için geliştirilmiştir. Her bir senaryo, belirlenen yapı ve prensiplere uygun olarak yazılmıştır.

## Yük/ Performans Testleri:

Aşağıda, Baykar Kariyer web sitesi için temel bir yük ve performans testi senaryosu adım adım açıklanmıştır.

### Python ve Locust araçları ile Baykar Kariyer(https://kariyer.baykartech.com/) web sitesi için yük ve performans testi senaryolarının yazılması.

##### Gereksinimler

* Python yüklü olmalı (Python 3.6 veya üzeri).
* Locust kütüphanesi kurulmalı. Terminalden aşağıdaki komutla kurulabilir: pip install locust

##### Test Senaryolarını Belirleme

Öncelikle test edilmesi gereken senaryoları belirleyelim. Bu senaryoları örnek olarak aşağıda listeledik:

* Ana Sayfa Erişimi: Kullanıcı ana sayfayı ziyaret eder.
* Açık Pozisyonlar Sayfasına Erişim: Kullanıcı, açık pozisyonların bulunduğu sayfaya erişir.
* Pozisyon Detayı Görüntüleme: Kullanıcı, belirli bir pozisyonun detay sayfasını açar.
* Başvuru Yapma (Form Gönderme): Başvuru formuna erişim ve formun gönderilmesi işlemlerini simüle etme.

Bu senaryolar, Locust ile tanımlanarak belirli bir sayıda kullanıcı tarafından gerçekleştirilmek üzere ayarlanabilir.

##### Locust Test Senaryosu Yazımı

Aşağıda, yukarıdaki senaryoları Locust ile yazılmış bir Python kodu olarak bulabilirsiniz:

from locust import HttpUser, TaskSet, task, between

class BaykarKariyerUserBehavior(TaskSet):

@task(1)

def load\_homepage(self):

"""Ana sayfa yükleme testi"""

self.client.get("/")

@task(2)

def load\_positions\_page(self):

"""Açık pozisyonlar sayfası yükleme testi"""

self.client.get("/positions") # Açık pozisyonlar URL’sini kontrol edin

@task(3)

def load\_position\_detail(self):

"""Pozisyon detay sayfası yükleme testi"""

# Örnek bir pozisyon ID'si ile bir sayfa testi yapılabilir

self.client.get("/positions/detail/123") # Belirli bir pozisyon ID

@task(4)

def submit\_application\_form(self):

"""Başvuru formunu gönderme testi"""

# Örnek verilerle bir form gönderme simülasyonu

form\_data = {

"name": "Test User",

"email": "testuser@example.com",

"phone": "5551234567",

"resume": "path/to/sample\_resume.pdf" # Formun yapısına göre uyarlayın

}

self.client.post("/apply", data=form\_data)

class BaykarKariyerUser(HttpUser):

tasks = [BaykarKariyerUserBehavior]

wait\_time = between(1, 5) # Kullanıcılar arası bekleme süresi (1-5 saniye)

host = "https://kariyer.baykartech.com" # Hedef site URL

##### Test Senaryosu Açıklamaları

• @task Dekoratörü: Her bir işleme bir @task dekoratörü eklenmiştir. Bu dekoratör, belirtilen işin test sürecinde hangi sıklıkta çalıştırılacağını ifade eder. Örneğin @task(2) ifadesi, diğer görevlere göre iki kat fazla çalıştırılacağı anlamına gelir.

• Ana Sayfa Yükleme (load\_homepage): Site ana sayfasının düzgün bir şekilde yüklendiğini kontrol eder.

• Açık Pozisyonlar Sayfası (load\_positions\_page): Siteye gelen kullanıcıların açık pozisyonlara göz atma senaryosunu test eder.

• Pozisyon Detay Sayfası (load\_position\_detail): Belirli bir pozisyonun detay sayfasına erişim işlemini simüle eder

• Başvuru Formu Gönderme (submit\_application\_form): Adayların başvuru formunu doldurup gönderme işlemini test eder. (Form elemanları Baykar kariyer sitesine göre uyarlanmalıdır.)

##### Testi Çalıştırma

Yukarıdaki kodu bir locustfile.py dosyasına kaydedin ve terminalden çalıştırın:

locust -f locustfile.py

Ardından Locust web arayüzüne gitmek için tarayıcıda http://localhost:8089 adresine gidin.

##### Yük Testi Parametreleri

Locust arayüzünde aşağıdaki parametreleri belirleyebilirsiniz:

• Total number of users to simulate (simüle edilecek kullanıcı sayısı): Test sırasında simüle edilecek toplam kullanıcı sayısını belirler.

• Spawn rate (users per second): Saniyede kaç kullanıcının teste ekleneceğini ifade eder.

Bu ayarları girerek “Start Swarming” butonuna tıkladığınızda test başlar ve sonuçları gerçek zamanlı olarak takip edebilirsiniz.

##### Test Sonuçlarının Analizi

Locust arayüzü aşağıdaki performans ölçümlerini sağlar:

• Requests per second (RPS): Saniye başına yapılan istek sayısı.

• Average response time: Ortalama yanıt süresi.

• Failure rate: Hata oranı.

• Percentile response times (50%, 90%, 99%): Belirli yüzdelik dilimlerdeki yanıt süreleri.

Bu ölçümler, sitenin performansını değerlendirmenizi sağlar. Yük altında, sayfaların kabul edilebilir bir hızda yüklenip yüklenmediğini veya kullanıcıların başvuru formunu sorunsuz gönderebilip gönderemediğini görebilirsiniz.

##### Ekstra Ayarlar ve İyileştirmeler

• Daha Karmaşık Senaryolar: Kullanıcı davranışını daha doğru simüle etmek için senaryoları daha ayrıntılı hale getirebilir ve kullanıcıların farklı yolları izlemesini sağlayabilirsiniz.

• Hedef URL’lerin Güncellenmesi: Web sitesindeki URL yapısına bağlı olarak, test URL’leri güncellenebilir.

• Sonuçların Raporlanması: Test sonuçlarını CSV veya HTML olarak dışa aktarmak için Locust’un --csv parametresini kullanabilirsiniz.

Bu test senaryolarıyla <https://kariyer.baykartech.com/> web sitesinin performansını ve yük dayanıklılığını analiz edebilirsiniz. Test sırasında, yanıt süresindeki artışları veya hata oranlarını gözlemleyerek sitenin iyileştirilmesi gereken alanlarını belirleyebilirsiniz.

### JMeter ile Baykar Kariyer(https://kariyer.baykartech.com/) web sitesi için yük ve performans test planı hazırlanması.

<https://kariyer.baykartech.com> web sitesi için JMeter ile bir yük ve performans test planı hazırlamak, sitenin kullanıcı trafiğini karşılayacak performansa sahip olup olmadığını anlamak için oldukça faydalıdır. Aşağıda, JMeter kullanarak bu test planını hazırlamanız için adım adım talimatlar bulunmaktadır.

##### JMeter Kurulumu

* Apache JMeter’i resmi sitesinden indirin ve sisteminize kurun.
* JMeter’ı çalıştırmak için jmeter.bat (Windows) veya jmeter.sh (Linux/Mac) dosyasını çalıştırın.

##### Test Planı Oluşturma

JMeter’ı açtıktan sonra yeni bir Test Planı oluşturun ve bu planı <https://kariyer.baykartech.com/> web sitesine uygun olarak yapılandırın.

• Adım 1: File > New menüsünden yeni bir Test Planı açın.

• Adım 2: Test planına sağ tıklayıp, Add > Thread (Users) > Thread Group seçeneğiyle bir Thread Group ekleyin. Thread Group, simüle edilecek sanal kullanıcıları ve bu kullanıcıların siteye erişim hızlarını ayarlamak için kullanılır.

Thread Group Ayarları

Thread Group içerisinde aşağıdaki temel ayarları yapın:

• Number of Threads (users): Simüle edilecek kullanıcı sayısını belirtir. (Örneğin, 50 kullanıcı)

• Ramp-Up Period (in seconds): Kullanıcıların ne kadar süre içinde teste dahil olacağını belirtir. (Örneğin, 50 saniye: Her saniye 1 kullanıcı eklenir)

• Loop Count: Her bir kullanıcının testi kaç kez tekrar edeceğini belirtir. Belirli bir süre boyunca döngü yaptırmak istiyorsanız, “Forever” seçeneğini işaretleyebilirsiniz.

##### HTTP Request Sampler Eklemek

Her sayfaya veya fonksiyona yönelik HTTP isteklerini tanımlamak için HTTP Request Sampler ekleyin.

* Thread Group’a sağ tıklayın ve Add > Sampler > HTTP Request seçeneğini seçin.
* İstek ayrıntılarını doldurun:

• Name: Bu isteğe anlamlı bir isim verin (Örneğin, “Ana Sayfa Yükleme”).

• Server Name or IP: kariyer.baykartech.com (URL’deki https:// ve / karakterleri hariç).

• Protocol: https olarak ayarlayın.

• Path: İstek yapacağınız URL yolunu girin. Örneğin, ana sayfa için /, pozisyonlar sayfası için /positions gibi.

Test Edilecek Sayfalar

Aşağıdaki gibi her sayfa için bir HTTP Request Sampler ekleyin:

• Ana Sayfa: Path = /

• Açık Pozisyonlar Sayfası: Path = /positions

• Pozisyon Detay Sayfası: Path = /positions/detail/{id} (Örnek bir pozisyon ID’si kullanarak)

• Başvuru Formu Gönderme: Path = /apply (Form elemanları ile POST isteği yapılması gerektiğinde uygun olarak yapılandırın)

##### HTTP Header Manager Ekleme

Web sitesine yapılan HTTP isteklerine uygun başlık bilgileri eklemek için bir HTTP Header Manager ekleyin:

* Thread Group’a sağ tıklayın, Add > Config Element > HTTP Header Manager seçeneğini seçin.
* HTTP Header Manager penceresinde User-Agent, Accept-Language gibi başlıkları ekleyin. Örnek başlık:

• User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Safari/537.36

• Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,/;q=0.8

##### HTTP Request Defaults Ekleme

Bu adımda temel ayarları tekrar yazmak yerine varsayılan istek ayarlarını yapabilirsiniz.

* Thread Group’a sağ tıklayın, Add > Config Element > HTTP Request Defaults seçeneğini seçin.
* Server Name or IP kısmına kariyer.baykartech.com girin, Protocol kısmına https girin.
* Bu ayarları yaptıktan sonra her bir HTTP Request’te sunucu adı ve protokol kısmını boş bırakabilirsiniz.

##### Yanıtları Kontrol Etmek için Assertion Ekleme

Her istekte dönen yanıtları kontrol etmek için Assertion ekleyerek doğrulama yapabilirsiniz.

* Her bir HTTP Request’e sağ tıklayın, Add > Assertions > Response Assertion seçeneğini seçin.
* Assertion ayarlarını yaparak belirli bir yanıtı veya metni doğrulayabilirsiniz. Örneğin, ana sayfa yüklendiğinde yanıt içinde “Baykar” ifadesi bulunmalı.

##### Listener (Dinleyici) Ekleme

Test sonuçlarını görüntülemek için Listener ekleyin. Listener, test sırasında toplanan verileri grafiksel veya tablo halinde görmenizi sağlar.

* Thread Group’a sağ tıklayın ve Add > Listener menüsünden ihtiyacınıza göre bir dinleyici seçin.
* Aşağıdaki dinleyicileri ekleyebilirsiniz:

• View Results in Table: Sonuçları tablo olarak gösterir.

• View Results Tree: Her isteğin detayını gösterir.

• Summary Report: Özet rapor sunar; istek sayısı, hata sayısı, ortalama yanıt süresi gibi bilgiler içerir.

• Aggregate Report: İstek başına en uzun süre, en kısa süre ve ortalama süre gibi toplu istatistikleri sunar.

##### Test Planını Kaydetme ve Çalıştırma

* Test planınızı File > Save menüsünden kaydedin.
* Test planını çalıştırmak için üst menüdeki Start (Yeşil ok) butonuna tıklayın.

##### Raporlama ve Analiz

Test bittikten sonra sonuçları inceleyin. Aşağıdaki metriklere dikkat edebilirsiniz:

• Throughput: Saniye başına tamamlanan istek sayısı.

• Average Response Time: Ortalama yanıt süresi.

• Error %: Hata oranı.

• 95% Line ve 99% Line: İsteklerin %95’inin ve %99’unun tamamlandığı süre.

Bu metrikler sayesinde sistemin mevcut yükü karşılayıp karşılamadığına dair genel bir fikir sahibi olabilirsiniz. Özellikle Average Response Time ve Error % oranları kritik öneme sahiptir.

##### Ekstra Yapılandırmalar ve İyileştirmeler

• Daha Karmaşık Kullanıcı Senaryoları: Aynı test planında birden fazla Thread Group oluşturarak farklı kullanıcı senaryolarını eş zamanlı test edebilirsiniz.

• Distributed Testing (Dağıtık Test): JMeter’ın dağıtık test özelliği ile yük testi için birden fazla bilgisayarı kullanabilirsiniz.

• CSV Data Set Config: Birden fazla kullanıcı adı ve şifre ile oturum açma gibi işlemleri test etmek için CSV Data Set Config öğesini kullanarak dinamik veri sağlayabilirsiniz.