T.C.
İSTANBUL ÜNİVERSİTESİ-CERRAHPAŞA
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

KURUMSAL KAYNAK PLANLAMA İŞ YAZILIMI SİSTEMLERİNDE UYGULAMA PROGRAMLAMA ARAYÜZÜ VE İSTEMCİ FARKLILIKLARININ PERFORMANSA ETKİSİ

Ali Burak ZEYTİNCİ

DANIŞMAN
Prof. Dr. Rüya ŞAMLI

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

İSTANBUL-2023

Bu çalışma, Tarih girmek için burayı tıklatın. tarihinde aşağıdaki jüri tarafından ....., ...nda ..... olarak kabul edilmiştir.

Tez Jürisi

Prof. Dr. Rüya ŞAMLI(Danışman)
İstanbul Üniversitesi-Cerrahpaşa
Mühendislik Fakültesi

Unvan Adı SOYADI
Üniversite
Fakülte

Unvan Adı SOYADI
Üniversite
Fakülte

Unvan Adı SOYADI
Üniversite
Fakülte

Unvan Adı SOYADI
Üniversite
Fakülte

20.04.2016 tarihli Resmi Gazete'de yayımlanan Lisansüstü Eğitim ve Öğretim Yönetmeliğinin 9/2 ve 22/2 maddeleri gereğince; Bu Lisansüstü teze, İstanbul Üniversitesi-Cerrahpaşa'nın abonesi olduğu intihal yazılım programı kullanılarak Lisansüstü Eğitim Enstitüsü'nün belirlemiş olduğu ölçütlere uygun rapor alınmıştır.

ÖNSÖZ

Tüm tez dönemi sürecimde benimle tecrübelerini paylaşan ve hiçbir desteği esirgemeyen değerli hocam Prof. Dr. Rüya ŞAMLI'ya teşekkürlerimi sunarım. Tez çalışmalarım boyunca bana destek olan; önce aileme, sonra okul arkadaşlarıma ve iş arkadaşlarımdan Serdar'a teşekkür ederim.

Mayıs 2023 Ali Burak ZEYTİNCİ

İÇİNDEKİLER

Sayfa No

ŞEKİL LİSTESİ

Sayfa No

TABLO LİSTESİ

SİMGE VE KISALTMA LİSTESİ

Kısaltmalar	Açıklama
API	: Application Programming Interface (Uygulama Programlama Arabirimi)
ERP	: Enterprise Resource Planning (Kurumsal Kaynak Planlaması)
HDD	: Hard Disk Drive (Sabit Disk Sürücüsü)
HTTP	: Hyper-Text Transfer Protocol (Hiper-Metin Transfer Protokolü)
JSON	: JavaScript Object Notation (JavaScript Nesnesi Gösterimi)
JS	: JavaScript
REST	: Representational State Transfer (Temsili Durum Aktarımı)
SOAP	: Simple Object Access Protocol (Basit Nesne Erişim Protokolü)
SAP	: Systemanalyse Programmentwicklung (Sistem Analizi ve Program Geliştirme)
WSDL	: Web Services Description Language (Web Hizmetleri Tanımlama Dili)
WWW	: World Wide Web (Dünya Çapında Ağ)
XML	: EXtensible Markup Language (Genişletilebilir İşaretleme Dili)

ÖZET

YÜKSEK LİSANS TEZİ

KURUMSAL KAYNAK PLANLAMA İŞ YAZILIMI SİSTEMLERİNDE UYGULAMA PROGRAMLAMA ARAYÜZÜ VE İSTEMCİ FARKLILIKLARININ PERFORMANSA ETKİSİ

Ali Burak ZEYTİNCİ

İstanbul Üniversitesi-Cerrahpaşa

Lisansüstü Eğitim Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı

Danışman : Prof. Dr. Rüya ŞAMLI

Bu tez kapsamında ERP (Enterprise Resource Planning - Kurumsal Kaynak Planlama) iş yazılımları içerisinde sıklıkla kullanılan SAP sistemlerinde uygulama programlama arayüzü (API – Application Programming Interface) ve istemci (client – consumer) farklılıklarının performansa etkisi incelenmiştir. Ayrıntılı olarak ifade etmek gerekirse, farklı uygulama programlama arayüzü kullanımlarının performansa etkisi, aynı web servislerden sağlanan verinin büyüdükçe farklı programlama dilleri tarafından yorumlanmasındaki performans farklılıkları, farklı uygulama programlama arayüzü kullanımının farklı programlama dillerinde oluşturabileceği performans farklılıkları ortaya çıkarılmıştır.
Bu amaçla, küçük ve büyük boyutlu verilerle SOAP (Simple Object Access Protocol - Basit Nesne Erişim Protokolü), REST (Representational State Transfer - Temsili Durum Transferi) JSON (JavaScript Object Notation – JavaScript Nesne Notasyonu) ve REST XML (Extensible Markup Language - Genişletilebilir İşaretleme Dili) çalışmaları gerçekleştirilmiş, her bir çalışmanın altında da C#, Java ve JavaScript(Node.js) programlama dilleri ve platformları ile çeşitli uygulamalar yapılmış ve bu uygulamaların performansları incelenmiştir.

Mayıs 2023, 184 sayfa.

Anahtar kelimeler: Kurumsal Kaynak Planlama, Uygulama Programlama Arayüzü, Performans Analizi

SUMMARY

Type of Thesis

EFFECTS OF API AND CONSUMER DIFFERENCES ON PERFORMANCE IN ERP BUSINESS SOFTWARE SYSTEMS

Ali Burak ZEYTİNCİ

Istanbul University-Cerrahpasa

Institute of Graduate Studies

Department of Computer Engineering

Supervisor : Prof. Dr. Rüya ŞAMLI

May 2023, 184 pages.

Within the scope of this thesis, ERP (Enterprise Resource Planning), the effects of application programming interface (API) and client (consumer) differences on performance in SAP systems, which are frequently used in business software, were examined. To put it in detail, the effect of using different application programming interfaces on performance, the performance differences in the interpretation of data provided from the same web services by different programming languages as they grow, and the performance differences that the use of different application programming interfaces can create in different programming languages have been revealed.

For this purpose, SOAP (Simple Object Access Protocol), REST (Representational State Transfer), JSON (JavaScript Object Notation) and REST XML (Extensible Markup Language) can be used with small and large data. Markup Language) studies were carried out, and under each study, various applications were made with C#, Java and JavaScript (Node.js) programming languages and platforms, and the performances of these applications were examined.

Keywords: Enterprise Resource Planning, Application Programming Interface, Performance Analysis

1. GİRİŞ

ERP yazılımları, bir işletmenin üretimden satışa, satın almadan muhasebeye dek uzanan çeşitli iş süreçlerinin ortak bir platformda toplandığı ve işletmelerin bu sayede farklı fonksiyonlarını daha kolay bir şekilde kontrol edebildiği yazılımlardır. İşletmeler bu yazılımlara farklı sebeplerden ötürü ihtiyaç duyabilmektedir. Diğer bir deyişle her işletmenin ERP yazılımlarına ihtiyaçları farklı olabilmektedir. Günümüzde piyasada birçok ERP yazılımı mevcuttur ve diğer tüm yazılım çeşitlerinde (işletim sistemleri, veritabanları, internet tarayıcılar vb) olduğu gibi ERP yazılımlarında da kurum için en uygun seçimin yapılması oldukça önemlidir. Uygun bir ERP yazılımı seçimi işletmeler için rekabet avantajı oluştururken, uygun olmayan bir ERP seçimi söz konusu projenin başarısız olmasına ve işletme performansı üzerinde olumsuz etki yapmasına neden olur. Uygun ERP yazılımı seçimi birçok kriter altında alternatiflerin değerlendirme skorlarına ve çoğunlukla yüksek, zayıf gibi dilsel terimlerle ifade edilen kriter ağırlıklarına bağlı olarak geliştirilen karmaşık bir süreçtir. Günümüzde, literatürde uygun ERP yazılımı seçimi için pek çok çalışma bulunmaktadır.

Genel olarak ERP seçimleri ile ilgilenen birçok akademik çalışma mevcuttur. Bayraktar ve Efe (2006) çalışmasında kurumsal kaynak planlaması ERP ve yazılım seçim süreci genel olarak ele alınmıştır. Dülgerler (2007) çalışmasında kurumsal kaynak planlaması ve web servisleri ile bir ERP uygulaması gerçekleştirilmiştir. Turan (2011) çalışmasında, KOBİ'lerin kurumsal kaynak planlama yazılımlarından beklentileri ve sektörel bazda yazılım geliştirilmesi ele alınmıştır. Kılıçaslan (2012) çalışmasında bir kurumsal kaynak planlama yazılımı uygulaması ve başarımı değerlendirilmiştir. Boztaş ve Özmızrak (2012) çalışmasında ERP yazılımları kurulum ve kullanım sürecinin bilgi yönetimi kavramıyla etkileşimi incelenmiştir. Çakır ve Bedük (2013) çalışmasında çalışanların kurumsal kaynak planlaması ERP değerlendirmeleri ve kurumsallaşma algıları incelenmiştir. Arslan (2015) çalışmasında ERP Microsoft Dynamıcs AX yazılımının şirketlere ve kamu kuruluşlarına uyarlanması gerçekleştirilmiştir. Tarhan (2017) çalışmasında, bir kurumsal kaynak planlama yazılımı ve akıllı karar destek sistemi araçları geliştirilmiştir.

Kimi zaman ERP'nin sadece bir bölgede ya da sadece bir konudaki uygulamalarının incelendiği çalışmalar görmek de mümkündür. Yeşildağ (2010) çalışmasında Muğla ilindeki KOBİ'lerde

ERP yazılımları kullanım düzeyi ve verimliliği araştırılırken, Topbaş (2019) çalışmasında yalın kurumsal kaynak planlaması yazılımının geliştirilmesi ve Kahramanmaraş'ta yalın üretim yapan işletmelerde uygulanması ele alınmıştır.

ERP yazılımı seçimi gerçekleştirilirken farklı yaklaşımların kullanıldığı da literatürde sıklıkla görülmektedir. Perçin ve Gök (2013) çalışmasında işletme problemlerinde kullanılan çok kriterli karar verme yöntemlerinden Analitik Ağ Süreci (AAS) ve TOPSIS yaklaşımlarının bir arada kullanılmasına yönelik bir metodoloji sunulmuştur. Sunulan iki aşamalı yaklaşımın uygulanabilirliğinin gösterilmesi amacıyla örnek bir uygulamaya yer verilerek, işletmeler için ERP yazılımı seçimi üzerine bir karar problemi ele alınmıştır. Vatansever ve Uluköy (2013) çalışmasında, üretim sektöründeki firmaların beklenti ve ihtiyaçlarına yönelik en uygun yazılım seçimi için bulanık AHP (Analitik Hiyerarşi Prosesi) ve bulanık MOORA (Oran Analizi Temeline Dayalı Çok Amaçlı Optimizasyon) yöntemleri bir arada kullanılarak yöneticilere karar desteği sağlanması amaçlanmaktadır. Yıldız ve Yıldız (2014) çalışmasında ERP yazılım seçimi için bulanık TOPSIS yönteminin nasıl uygulanacağını bütüncül bir yapı içinde göstermeyi hedeflemiştir. Bu yapı beş alternatifli on kriterli değişkenlere dayalı olarak bir firma için geliştirilmiştir. Değerlendirme sonucunda beşinci alternatif birinci sırada seçilmiş ve yazılım maliyetleri ile yazılımın süreçlere uyumluluğu da en önemli kriterler olarak belirlenmiştir. Başar ve Arslan (2017) çalışmasında, işletmeler için en etkin ERP yazılımı seçiminde çok kriterli karar analizi metotlarından VIKOR (En Uygun Uzlaşık Çözüm) yönteminden faydalanılmıştır. Polyester üretim sektöründe faaliyet gösteren bir işletmenin karar almada yetkili çalışanları ile ERP, seçim kriterleri ve muhtemel alternatifler hakkında görüşülmüştür. Ayrıca çalışmada ERP yazılımlarının işletmelere sağladığı faydalar ve seçim sürecine ilişkin önemli hatırlatmalar üzerinde durulmuştur. VIKOR yöntemi ile yapılan analizler sonucunda işletme için en etkin ERP yazılımları sırası belirlenmiş ve sonuçlar işletme ile paylaşılmıştır. Özkan Özen ve Koçak (2017) çalışmasında, fason imalat ve makine bakım konularında faaliyet gösteren bir imalat firmasında öncelikle sistem analizi çalışması yapılarak ihtiyaç listesi çıkarılmıştır. Sonrasında oluşan bu seçim kriterlerine göre, nihai olarak karar verilen iki yazılım firmasının uygulama yazılımlarında bulanık analitik hiyerarşi yöntemi kullanılarak seçim süreci gerçekleştirilmiştir. Buna ek olarak ERP seçimine konu olan seçim kriterlerinin etkileyen ve etkilenen ilişkileri bulanık DEMATEL yöntemi ile değerlendirilerek seçim sonrası ERP yazılımının kurulumunda rehber olacak stratejik bir yol haritası oluşturulmuştur. Bal (2020) çalışmasında, AHp ve Birliktelik Kuralları Analizi kullanılarak ERP seçimi yapılmış ve bu seçimler arasında performans analizleri gerçekleştirilmiştir.

Madencioğlu (2021) çalışmasında, dış kaynak kullanımıyla temin edilecek olan ERP yazılımı seçimi problemine, karar verme sürecine belirsizliğin dahil edildiği çok kriterli bir çözüm yaklaşımı önerilmiştir. Önerilen çözüm yaklaşımında alternatiflerin değerlendirmesinde kullanılacak olan kriter ağırlıklarının belirlenmesinde Shannon entropi yöntemi ve alternatiflerin sıralanmasında Bulanık TOPSİS, Bulanık GİA, Bulanık Maut, Bulanık Aras, Bulanık Waspas, Bulanık Copras, Bulanık Edas yöntemleri kullanılmıştır. Farklı çok kriterli karar verme yöntemlerinden elde edilen sıralamalar bütünleştirilerek alternatiflerin nihai sıralaması elde edilmiştir. Çalışmada önerilen çözüm yaklaşımı, mobilya sektöründe faaliyet gösteren bir işletmenin dış kaynak kullanımı ile işletmeye uygun ERP yazılımının seçim sürecine uygulanmış ve sonucunda işletmeye en uygun olan ERP yazılımı seçilmiştir. Dikmen ve Kavakci (2022) çalışmasında, 2020 yılı 1. ve 2. ISO 500 anket sonuçlarında yer alan ve Malatya İlinde faaliyet gösteren firmaların ERP yazılım seçenekleri arasından ölçütlerine en uygun olan ERP yazılımını belirlemektir. Bu amaçla ÇÖKV (Çok Ölçütlü Karar Verme) yöntemlerinden MACBETH, TOPSIS ve COPRAS yöntemleri bütünleşik olarak kullanılmış ve ERP yazılım seçimi gerçekleştirilmiştir.

Bu tez çalışmasında da bir ERP yazılımı olan SAP yazılımının performans incelemesi gerçekleştirilmiştir. Bu amaçla, küçük ve büyük boyutlu verilerle SOAP, REST JSON ve REST XML uygulamaları ve bu uygulamaların performans analizleri gerçekleştirilmiştir.

2. GENEL KISIMLAR

Bu bölümde tez çalışması kapsamında ele alınan ve ortaya çıkan analiz sonuçlarının kolayca anlaşılabilmesi için gerekli olan kavramlar açıklanmıştır. Bu kavramlar; kurumsal kaynak planlama sistemlerinin tanımı ve özellikleri, uygulama programlama arayüzlerinin tanımı, özellikleri ve çeşitleri, web servislerin tanımı, özellikleri ve çeşitleridir.

2.1. KURUMSAL KAYNAK PLANLAMA SİSTEMLERİ

Kurumsal kaynak planlama sistemleri, yazılımın ana iş süreçlerine entegre yönetimidir. Kurumsal kaynak planlama(ERP), genellikle bir kuruluşun birçok ticari faaliyetten veri toplamak, depolamak, yönetmek ve yorumlamak için kullanabileceği iş yönetimi entegre yazılım paketidir. ERP sistemleri yerel tabanlı veya bulut tabanlı olabilir. Bulut tabanlı uygulamalar, bilgilerin internet erişimi olan herhangi bir yerden kolayca erişilebilir olması ve kaynakların bölüşülebilir olmasından doğan uygun fiyat avantajları nedeniyle son yıllarda oldukça yaygınlaşmıştır.

2.2. UYGULAMA PROGRAMLAMA ARAYÜZLERİ

Uygulama programlama arayüzü (API), bir yazılımın fonksiyonlarının diğer bir yazılımda kullanılmasını sağlar. Bununla beraber bir veritabanına veya bir bilgisayar donanımına erişmeyi de sağlamaktadır.

Örneğin bir programlama dili ile yazılmış olan bir önyüz, bir veritabanından gelecek verilerle doldurulması gerektiğinde web servislere ihtiyaç duyarız. Web servisler bu verilere erişir ve bazı internet protokolleri ile veriyi dışarıya açar, örneğimizdeki önyüz programı veya birçok farklı alandan doğru istekleri yapan ve yetkileri olan herkes bu verilere erişir, işler, kullanır.

Bilgisayar donanımına erişmekte özellikle HDD (sabit disk sürücüleri) veya ekran kartlarına erişmekte kullanılıyor. API, grafiksel kullanıcı arayüzündeki (GUI) işi kolaylaştırıyor, CPU–GPU–uygulama arasındaki haberci olup uygulamanın çalışmasında rol oynuyor.

Örnek olarak; yeni bir Low-Level API olan Mantle, donanım–işletim sistemi–CPU–GPU arasında bir sürü katmandan oluşmayan bir yapı ile yapılacak işlemler için donanımla hızlıca iletişim kurabiliyor. GPU'ya iletişim hızlanınca CPU daha az yoruluyor ve GPU'yu daha çok besleyebiliyor. Böylelikle de yapılacak işlemler daha az yük gereken yollar ile yapılabiliyor.

2.3. WEB SERVİSLER

Bir web servisi; elektronik bir cihazın başka bir elektronik cihazla internet üzerinden konuşması ve iletişimde kalmasını sağlayan bir servis veya bir bilgisayarda çalışan, bir ağ üzerinden belirli bir bağlantı noktasındaki istekleri dinleyen, bu isteklere cevap sunan bir sunucudur.

Web servisleri, WWW protokollerini kullanarak internet üzerinde çalışan bilgisayarlardan başka bilgisayarlara veri aktaran servislerdir.

En yaygın web servis çeşitlerinden SOAP ve REST tanımları ve özelliklerine üçüncü bölümde detaylıca yer verilmiştir.

3. MALZEME VE YÖNTEM

Bu bölümde tez çalışması kapsamında kullanılan kurumsal kaynak planlama iş yazılımı olarak SAP, programlama dilleri, uygulama programlama arayüzleri ve web servisler detaylı şekilde açıklanıp kullanılan yöntemler ve uygulanan kriterlere yer verilmiştir.

3.1. SAP KURUMSAL KAYNAK PLANLAMA İŞ YAZILIMI

SAP, şirket içi organizasyonlar arasında veri işlemeyi ve bilgi akışını kolaylaştıran çözümler geliştiren bir kurumsal kaynak planlama iş yazılımıdır. İş süreçlerinin yönetimi için dünyanın önde gelen yazılım üreticilerinden biridir.

Kurumsal kaynak planlama iş yazılımının en önemli isimlerinden ve öncüsü IBM'den ayrılan mühendislerin 1972 yılında Almanya'da Systemanalyse Programmentwicklung ismiyle kurup zaman içerisinde SAP kısaltmasıyla adlandırdıkları bir iş yazılımıdır.

Bugün SAP, 230 milyondan fazla bulut kullanıcısına, tüm iş fonksiyonlarını kapsayan 100'den fazla çözüme ve tüm altyapı sağlayıcılarının en büyük bulut portföyüne sahiptir.

3.2. PROGRAMLAMA DİLLERİ

Bu tez çalışması JavaScript(Node.js), C# ve Java programlama dilleri kullanılarak gerçeklenmiş ve bu programlama dillerinin performans analizleri gerçekleştirilmiştir. Kurumsal kaynak planlama iş yazılımı olarak seçilen SAP ise kendi geliştirdiği ABAP dilini desteklemektedir. Bu dillerle geliştirilen uygulamaların yürütüldüğü ortamlara da yine bu bölümde yer verilmiştir.

3.2.1. ABAP

ABAP, 1980'lerde geliştirilmeye başlanan bir uygulamaya özel dördüncü nesil dillerden biridir. Başlangıçta, büyük şirketlerin malzeme yönetimi, finans ve muhasebesi için iş uygulamaları oluşturmasını sağlayan bir platform olan SAP R/2'nin rapor(program) diliydi.

ABAP, Jenerik Rapor Hazırlama İşlemcisi'nin(Allgemeiner Berichts-Aufbereitungs-Prozessor) kısaltmasıydı, ancak daha sonra Almanca yerine İngilizce'ye geçerek Advanced Business Application Programming olarak yeniden adlandırıldı.

Kısaca SAP sistemlerinde kullanılan uygulamaya özel bir programlama dilidir.

3.2.2. C#

C#, geliştiricilerin .NET üzerinde çalışan çeşitli, güvenli ve güvenilir uygulamalar oluşturmasını sağlayan üst düzey, sınıf tabanlı, nesne yönelimli bir programlama dilidir.

C#; web uygulamaları, konsol uygulamaları, masaüstü uygulamaları, mobil uygulamalar, oyunlar ve çok daha fazlasını geliştirmek için kullanılır. Microsoft tarafından geliştirilmektedir.

3.2.3. Java

Java, 1995 yılında Sun Microsystems tarafından ortaya çıkarılan bir programlama dili ve bilgi işlem platformudur. Java, olabildiğince az uygulama bağımlılığı olacak şekilde tasarlanmış, üst düzey, sınıf tabanlı, nesne yönelimli bir programlama dilidir. Programcıların bir kez yazıp her yerde çalıştırmasını (Write Once Run Anywhere – WORA) amaçlayan genel amaçlı bir programlama dilidir. Yani derlenmiş bir Java kodu, Java'yı destekleyen tüm platformlarda, yeniden derlemeye gerek kalmadan çalışabilir.

Java uygulamaları genellikle, temeldeki bilgisayar mimarisinden bağımsız olarak herhangi bir Java Sanal Makinesi'nde (JVM) çalışabilen bayt kodunda derlenir. Java'nın sözdizimi C ve C++'a benzer, ancak her ikisinden de daha az alt düzey olanaklara sahip daha üst düzey bir dildir.

3.2.4. JavaScript (Node.js)

JavaScript, World Wide Web'in HTML ve CSS ile birlikte en temel üç teknolojilerinden biridir. Günümüzde neredeyse tüm web siteleri bir veya birden fazla JavaScript kütüphanesi yardımıyla geliştiriliyor. Kullanılan tüm popüler web tarayıcılarında JavaScript ile geliştirilmiş bu kodları kullanıcının cihazında çalıştırmak için bir JavaScript motoru bulunuyor. Bu sayede cep telefonlarımızdan akıllı saatlerimize bilgisayarlarımızdan yeni nesil otomobillerimize birçok internete bağlanan ve web tarayıcısı olan cihazda JavaScript ile oluşturulmuş web sitelerini görüntüleyebiliyoruz.

JavaScript, ECMAScript standardına uyan, üst düzey, genellikle tam zamanında(just-in-time) derlenen, prototip tabanlı, nesne yönelimli bir programlama dilidir. JavaScript motorları ilk başta sadece web tarayıcılarında kullanılıyordu, ancak artık sunucuların ve uygulamaların temel bileşeni haline geldi. Bu alanda en popüler JavaScript Runtime ortamı Node.js'dir.

3.3. PROGRAMLAMA PLATFORMLARI

Bu tez çalışmasında ele alınan algoritmalar, farklı programlama dilleri kullanılarak gerçekleştirildiği için farklı programlama platformlarının da kullanılması gerekmiştir.

C# programlama dili ile kodlanan programlar Visual Studio platformunda, Java programlama dili ile kodlanan programlar IntelliJ IDEA platformunda, Node.js ile kodlanan programlar ise Visual Studio Code platformunda gerçeklenmiştir. Bu bölümde bu programlama platformları hakkında kısaca bilgi verilmiştir.

3.3.1. Visual Studio Code

Visual Studio Code (diğer bilinen adıyla VS Code veya VSC), Windows, Linux ve macOS için Microsoft tarafından Electron Framework ile yapılmış bir kaynak kodu düzenleyicisidir. Özellikler arasında hata ayıklama, sözdizimi vurgulama(syntax highlighting), akıllı kod tamamlama(IntelliSense), snippets, kod refactoring ve gömülü Git desteği yer alır. Kullanıcılar temayı, klavye kısayollarını, tercihleri değiştirebilir ve ek işlevler ekleyen uzantıları yükleyebilir. Uzantılar ile size sınırsız kolaylık, geliştirme ortamı ve programlama dili desteği sunmaktadır.

Visual Studio Code ile JavaScript ve TypeScript yetleşik olarak desteklenmektedir. Diğer dillere ise eklentiler ile destek vermektedir.

3.3.2. Visual Studio

Visual Studio, .NET ortamında C# dilinde web, masaüstü, konsol ve mobil uygulama geliştirmek için kullanılır. Microsoft tarafından geliştirilmektedir. Visual Studio ile yalnız .NET ortamında C# değil, birçok farklı programlama dillerinde geliştirme yapılabilir.

Visual Studio Community, giriş seviyesi sürüm olup ücretisz olarak kullanılabilmektedir. Visual Studio Community sürümünün sloganı “Öğrenciler, açık kaynak ve bireysel geliştiriciler için ücretsiz, tam özellikli IDE”dir.

3.3.3. IntelliJ IDEA

IntelliJ IDEA, JetBrains tarafından geliştirilen bir entegre geliştirme ortamıdır. 2001 yılında piyasaya sürülmüş olup Windows, macOS ve Linux işletim sistemlerinde kullanılabilir. Başlangıçta Java programlama diline odaklanmış olsa da yerleşik olarak veya eklentiler

aracılığıyla diğer programlama dillerini de desteklemektedir. Hem ücretsiz bir “Community” sürümü hem de ticari bir “Ultimate” sürümü mevcuttur. IntelliJ IDEA kaynak kodunu JetBrains açık kaynak Apache Lisansı 2.0 altında 2009 yılında yayınladı.

3.4. ERİŞİM PROTOKOLLERİ

Bu tez çalışmasında SAP tarafına yapılan istekler ve alınan cevaplarda farklı standartlar kullanılmaktadır. Bunlar SOAP ve REST protokolleridir.

3.4.1. SOAP

Basit Nesne Erişim Protokolü(SOAP), bir paketin farklı uygulamalarının iletişim kurmasını sağlayan bir mesajlaşma protokolüdür. SOAP, web ile ilgili HTTP dahil olmak üzere çeşitli standart protokoller üzerinden taşınabilir.

SOAP, farklı programlama dillerine sahip uygulamalar için bir ara dil olarak geliştirilmiş ve bu uygulamaların internet üzerinden birbirleriyle iletişim kurmasını sağlamıştır. SOAP, esnek ve bağımsızdır, bu da geliştiricilerin farklı dillerde SOAP uygulama programlama arabirimleri(API'ler) yazmasına ve aynı zamanda özellikler ve işlevler eklemesine olanak tanır.

SOAP, genellikle Genişletilebilir İşaretleme Dili (XML) ile web API'leri oluşturmak için kullanılan hafif bir protokoldür. HTTP, Basit Posta Aktarım Protokolü(SMTP) ve İletim Kontrol Protokolü(TCP) üzerinden çok çeşitli iletişim protokollerini destekler. SOAP yaklaşımı, bir SOAP mesajının nasıl işlendiğini, içerdiği özellikleri ve modülleri, desteklenen iletişim protokollerini ve SOAP mesajlarının yapısını tanımlar.

3.4.2. REST

REST, bir protokol veya standart değil, bir dizi mimari kısıtlamadır. API geliştiricileri, REST'i çeşitli şekillerde uygulayabilir. RESTful API, REST mimari stilinin kısıtlamalarına uyan ve RESTful web hizmetleriyle etkileşime izin veren bir uygulama programlama arayüzüdür(API). REST, temsili durum aktarımı(representational state transfer) anlamına gelir ve bilgisayar bilimcisi Roy Fielding tarafından oluşturulmuştur. RESTful API aracılığıyla bir alıcı isteği yapıldığında, kaynağın durumunun bir temsilini istek sahibine veya uç noktaya aktarır. Bu bilgi veya temsil, HTTP aracılığıyla birkaç formattan birinde iletilir: JSON, HTML, XML veya düz metin. JSON, genel olarak kullanılan en popüler dosya formatıdır çünkü ismine rağmen dilden bağımsızdır ve hem insanlar hem de makineler tarafından okunabilmektedir.

3.5. VERİ FORMATLARI

Bu tez çalışmasında SAP tarafına yapılan istekler ve alınan cevaplarda farklı standartlar kullanılmaktadır. Bunlar SOAP için WSDL iken REST için JSON ve XML'dir.

3.5.1. XML

Genişletilebilir İşaretleme Dili (XML) verileri depolamak ve farklı bilgisayarlar arasındaki iletişimde verileri iletmek için bir dosya formatı. Verileri hem makine hem de insan tarafından okunabilecek bir şekilde kodlanması için kuralları vardır. World Wide Web Konsorsiyumu'nun 1998 tarihli XML 1.0 Spesifikasyonu XML'i tanımlar. XML'in tasarım hedefleri; basitliği, genelliği ve internet genelinde kullanılabilirliği vurgular. Farklı insan dilleri için Unicode aracılığıyla desteği olan metinsel bir veri formatıdır.

3.5.2. WSDL

Web Hizmetleri Açıklama Dili (WSDL), 26 Haziran 2007 tarihli bir W3C(World Wide Web Konsorsiyumu) standartıdır. Bir web servisiyle yapılabilecek tüm işlemleri ve bu işlemler için kullanılacak mesajları ve veri türlerini tanımlayan bir XML formatıdır. Geçerli bir WSDL dosyası, bir web servisine istek göndermek için ihtiyacınız olan tüm bilgileri içerir.

3.5.3. JSON

JSON(JavaScript Object Notation), hafif(lightweight) bir veri değişim formatıdır. İnsanlar için okuma-yazması kolaydır. Makinelerin ayrıştırması ve oluşturması kolaydır. JSON, dilden tamamen bağımsızdır, ancak C dahil, C ailesi programcılarının aşina olduğu kuralları kullanan bir metin formatıdır. Bu, JSON'u ideal bir veri değişim dili yapar.

JSON iki yapı üzerine kuruludur:

- **isim:değer çiftlerinden oluşan bir yığın :** Çeşitli dillerde bu; bir nesne, kayıt, yapı, sözlük, karma tablo, anahtarlı liste veya ilişkisel dizi olarak gerçekleştirilir.
- **Sıralı değerler listesi:** Çoğunlukla bu bir dizi, liste, sıra veya vektör olarak dillerde yer alır.

Bunlar evrensel veri yapılarıdır. Neredeyse tüm modern diller bunları şu veya bu biçimde destekler. Programlama dilleriyle değiştirilebilen bir veri formatının da bu yapılara dayalı olması mantıklıdır.

4. BULGULAR

Bu başlıkta performans karşılaştırması için geliştirilen uygulamalar ve gerçeklenen testler açıklanmaktadır. Çalışmada; bir bilgisayar üzerinde çalıştırılan üç farklı programlama dili ile yazılmış olan programlardan SAP sunucusunun çalıştığı bir sunucuya küçük veya büyük veri cevabı döndürecek istekler atılmaktadır. Bu istekler C#, Java ve Node.js ile farklı sınıflar ve kütüphaneler kullanılarak SAP tarafındaki SOAP veya REST API'leri uyandırmakta ve sonrasında alınan cevaplar yorumlanmaktadır. Bu çalışma ile istemci tarafında kullanılan programlama dilleri ve bunların kendi içlerinde kullanılabilecek olan farklı kütüphane veya sınıfların SOAP ve RESTful API'lar karşısında küçük ve büyük verilerin farklı dosya formatları ile haberleştiği durumlarda ortaya çıkacak olan performans farklarını gözler önüne sermiş olacağız. Her bir sonuç, üç ana başlıktan oluşan bir döngünün ürünüdür; istek paketlerinin oluşturulması, gönderilmesi ve cevabın alınması, cevabın yorumlanması.

İstemci tarafında isteklerin yollandığı bilgisayar özellikleri:

- AMD Ryzen 5 4600H CPU @ 3.00 GHz 6C/12T
- 16 GB RAM
- Windows 10 Home 22H2

Sunucu tarafında isteklerin cevaplandığı yerel sunucunun (LOCAL) özellikleri:

- 6 Çekirdek AMD İşlemci
- 12,75 GB RAM
- SUSE Linux Enterprise Server 15 SP4

Sunucu tarafında isteklerin cevaplandığı canlı sunucunun (S4) özellikleri:

- 16 Çekirdek Intel İşlemci
- 364 GB RAM
- SUSE Linux Enterprise Server 15 SP3

Sunucu tarafında isteklerin cevaplandığı demo sunucusunun (IDES) özellikleri:

- 8 Çekirdek Intel İşlemci
- 32 GB RAM
- SUSE Linux Enterprise Server 15 SP2

Test senaryolarını çeşitlendirerek elde edilen sonuçların daha güvenilir olması ve sunucuların donanım ve yazılım özelliklerinin ve farklı lokasyonlardan birbirleriyle haberleşen sistemlerin test sonuçlarına etkisini gösterebilmek amacıyla farklı SAP sunucuları bu tez kapsamında kullanılmıştır.

Bu çalışmada her programlama dili için geliştirici topluluğunun en uygun gördüğü bütünleşmiş geliştirme ortamları tercih edilmiştir. C# dili, Visual Studio 17.5.3 ortamında ve .NET 7.0.202 platformunda gerçeklenmiştir. Java dili, IntelliJ IDEA 2022.3.3 ortamında ve JDK 19.0.1 sürümünde gerçeklenmiştir. JavaScript dili, Visual Studio Code 1.77.1 ortamında ve Node.js 18.15.0 sürümünde gerçeklenmiştir. Bütün çalışmalar, bahsedilen geliştirme ortamlarının release build ayarları ile gerçekleştirilmiştir.

4.1. UYGULAMALAR

4.1.1. Küçük Veriyle SOAP Uygulamaları

Bu tez çalışmasında ilk olarak küçük verilerle SOAP API çalışılmıştır. Test çeşitliliği ve sonuçların doğru yorumlanabilmesi için üç farklı sunucuyla çalışılmış ve sunuculardaki veri büyüklükleri sırasıyla IDES ile LOCAL arasında yaklaşık 100 kat, LOCAL ile S4 arasında yaklaşık 5 kat büyüklük farkı olacak şeklindedir. Üç farklı programlama dili kullanılıp bunlar sırasıyla C#, Java ve JavaScript(Node.js) dilleri olmuştur. Bu programlama dillerinin de kendi içlerinde farklı kütüphaneler ve sınıflar kullanılarak test sonuçları her programlama dili içerisinde çeşitlendirilmiştir. Her bir program uygun olduğu geliştirme ortamında geliştirilip derlenmiş ve çalıştırılmıştır. Her bir çalıştırma döngüsü kendi içerisinde üç ana başlıktan oluşmaktadır; istek paketlerinin oluşturulması, gönderilmesi ve cevabın alınması, cevabın yorumlanması. Bu üç başlığın toplam çalışma süresi milisaniye(ms) biriminden kaydedilmiş ve tez çalışması kapsamında yorumlanmıştır. Bu tez çalışması kapsamında yorumlanan veriler; programların toplam çalışma süreleri değil, belirtilen bu üç ana başlıkta geçirilen sürelerdir.

İstek atılan SAP sunucusunun SOAP WSDL tanımlamasına EK 1.1'de yer verilmiştir.

SAP sunucumuza gönderilen SOAP istek mektubuna EK 1.2'de yer verilmiştir.

SAP sunucumuzun bu isteğe cevabına EK 1.3'te yer verilmiştir.

Kullanılan farklı diller ve bu dillerde kullanılan farklı kütüphaneler veya sınıflar SOAP istek mektubunu ve HTTP 1.1 isteğini farklı yöntemlerle oluşturup, gönderip, EK 1.3'te yer verilen

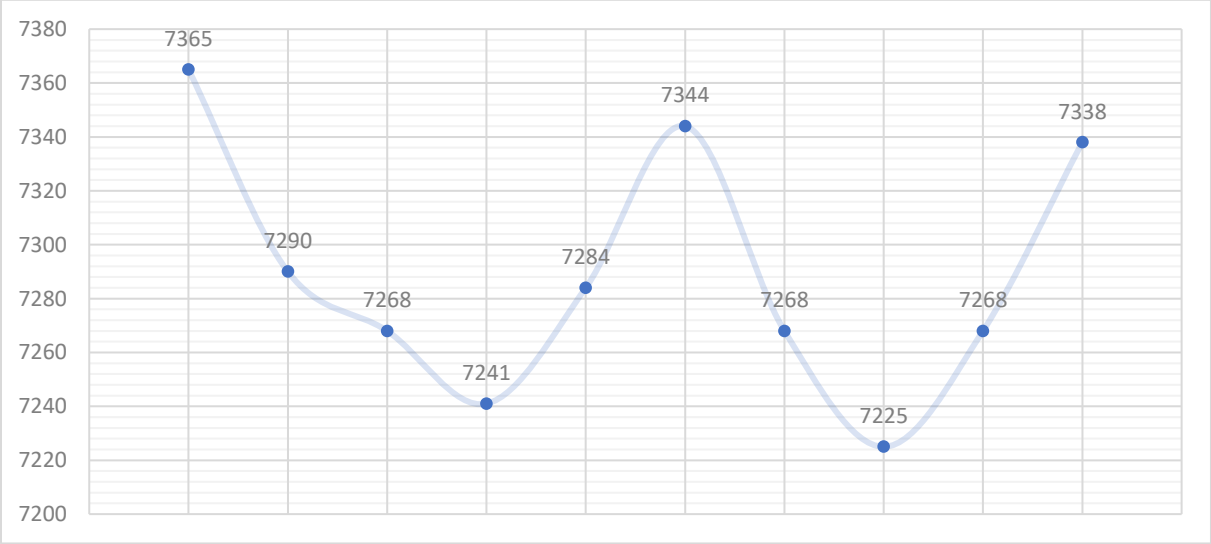
aynı cevabı almıştır. Bu cevap; SAP tarafında veritabanının farklı alanlarından SQL sorguları ve ABAP fonksiyonlarıyla çekilmiş, paketlenmiş ve gönderilmiştir. SAP ABAP program koduna EK 4.1'de yer verilmiştir.

C# HttpClient Çalışması

Bu çalışma C# programlama dilinde .NET 7.0 platformunda HttpClient sınıfı kullanılarak Visual Studio ortamında gerçeklenmiş ve Tablo 4.1'deki ve Şekil 4.1-Şekil 4.3 arasındaki sonuçlar elde edilmiştir. Program koduna EK 1.4'te yer verilmiştir.

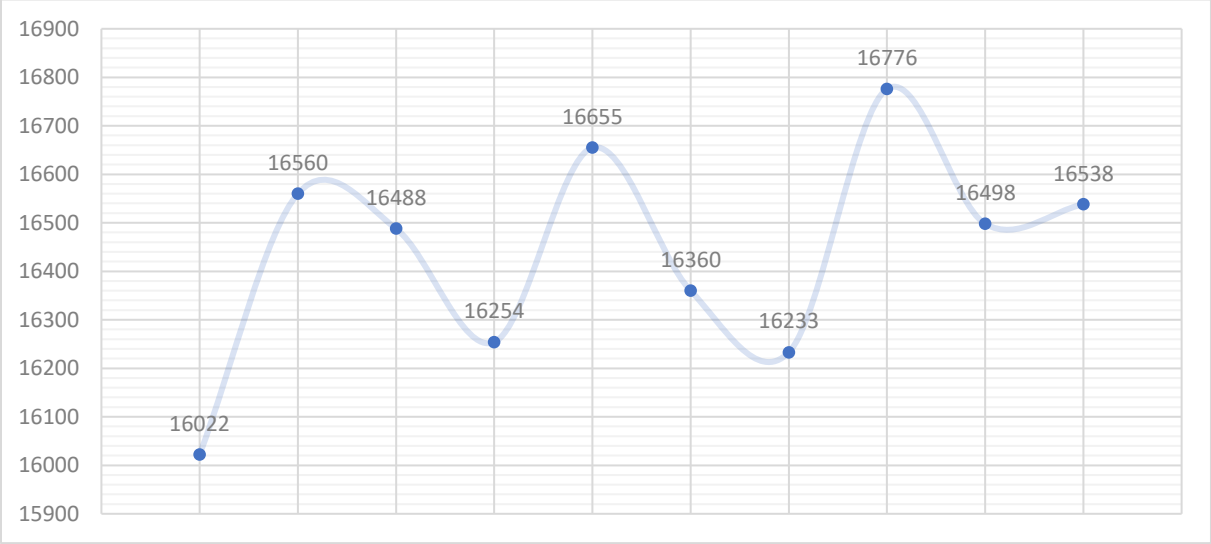
Tablo 4.1 SOAP Küçük Veri C# HttpClient

Tekrar	LOCAL	S4	IDES
1	7365	16022	238
2	7290	16560	223
3	7268	16488	228
4	7241	16254	227
5	7284	16655	225
6	7344	16360	245
7	7268	16233	222
8	7225	16776	246
9	7268	16498	256
10	7338	16538	207
Ortalama	**7289**	**16438**	**232**



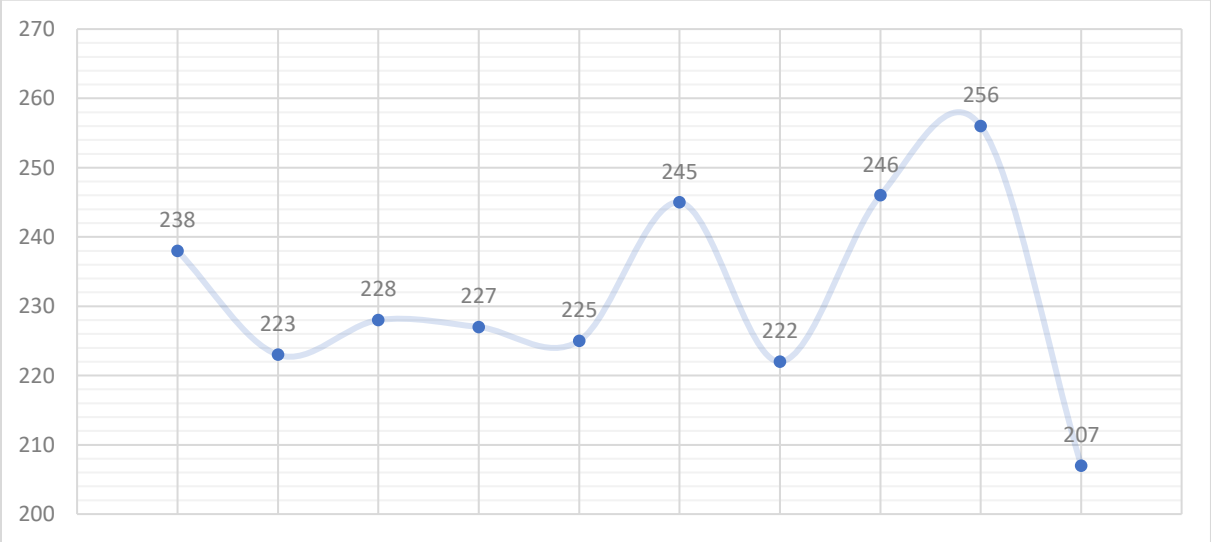
Şekil 4.1 LOCAL SOAP Küçük Veri C# HttpClient

Test ortalaması 7289 ms olup, açıklığın ortalamaya sapması %1,9 ve çeyrekler açıklığının ortalamaya sapması %0,8 olduğu kaydedilmiştir.



Şekil 4.2 S4 SOAP Küçük Veri C# HttpClient

Test ortalaması 16438 ms olup, açıklığın ortalamaya sapması %4,6 ve çeyrekler açıklığının ortalamaya sapması %1,7 olduğu kaydedilmiştir.



Şekil 4.3 IDES SOAP Küçük Veri C# HttpClient

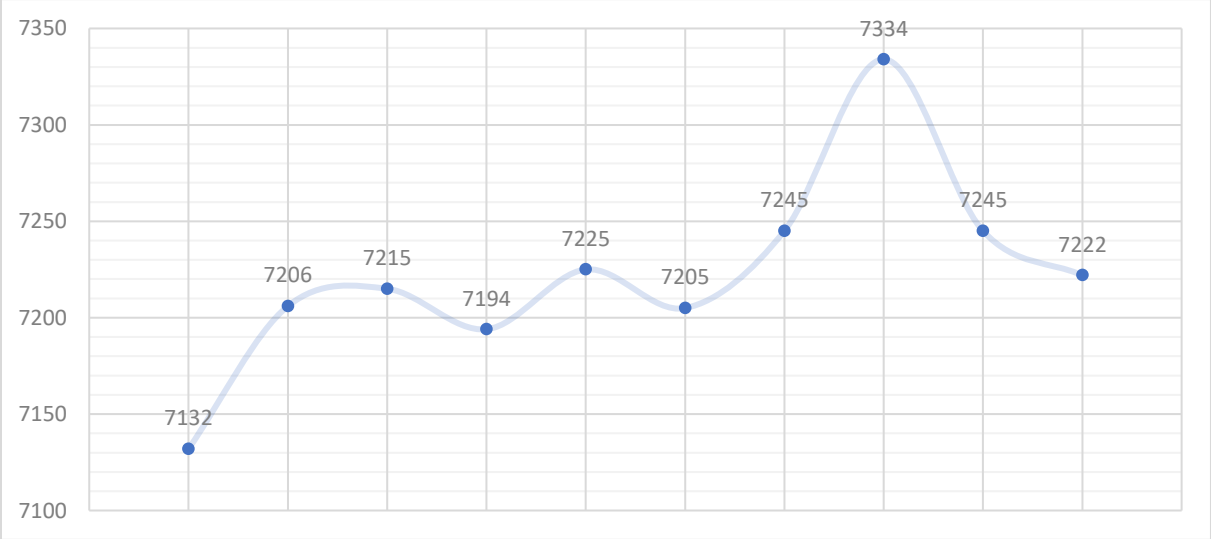
Test ortalaması 232 ms olup, açıklığın ortalamaya sapması %21,1 ve çeyrekler açıklığının ortalamaya sapması %8,5 olduğu kaydedilmiştir.

C# WebClient Çalışması

Bu çalışma C# programlama dilinde .NET 7.0 platformunda WebClient sınıfı kullanılarak Visual Studio ortamında gerçeklenmiş ve Tablo 4.2'deki ve Şekil 4.4-Şekil 4.6 arasındaki sonuçlar elde edilmiştir. Program koduna EK 1.5'te yer verilmiştir.

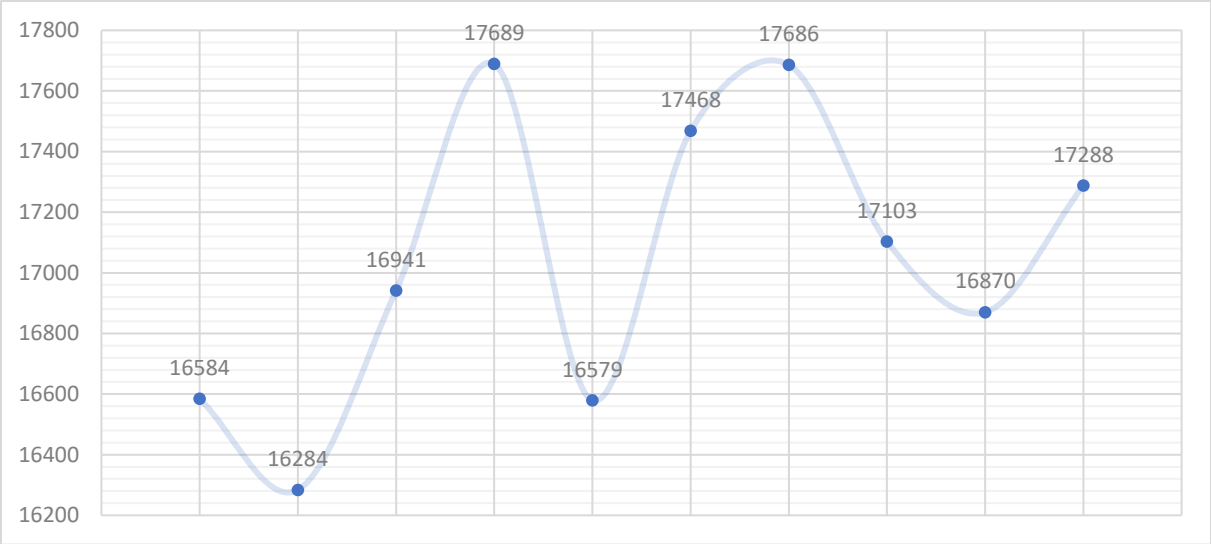
Tablo 4.2 SOAP Küçük Veri C# WebClient

Tekrar	LOCAL	S4	IDES
1	7132	16584	216
2	7206	16284	228
3	7215	16941	207
4	7194	17689	227
5	7225	16579	197
6	7205	17468	222
7	7245	17686	243
8	7334	17103	206
9	7245	16870	206
10	7222	17288	268
Ortalama	**7222**	**17049**	**222**



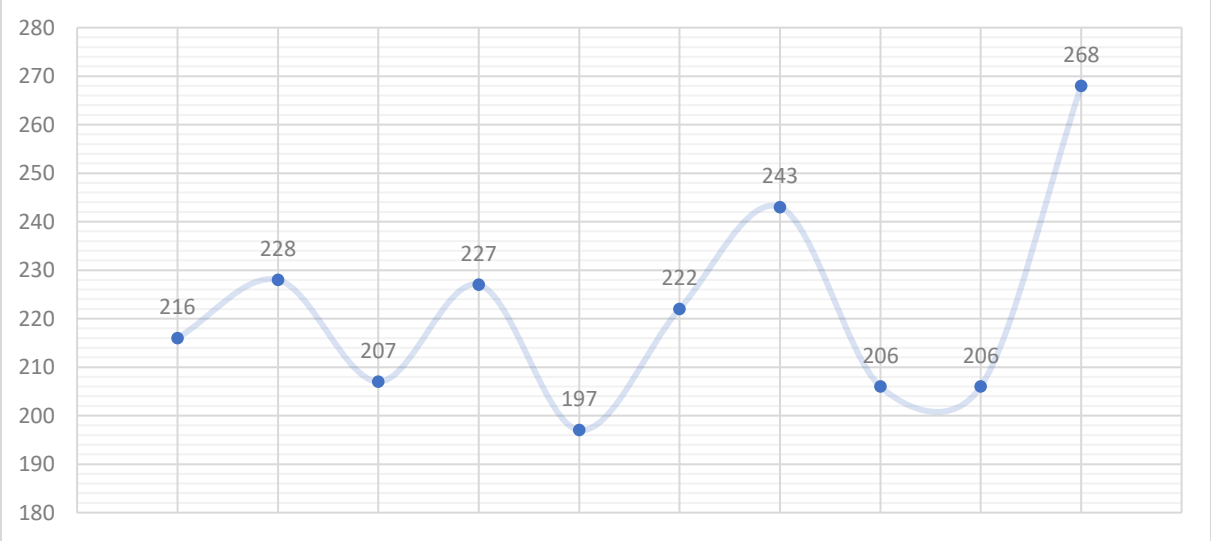
Şekil 4.4 LOCAL SOAP Küçük Veri C# WebClient

Test ortalaması 7222 ms olup, açıklığın ortalamaya sapması %2,8 ve çeyrekler açıklığının ortalamaya sapması %0,5 olduğu kaydedilmiştir.



Şekil 4.4 S4 SOAP Küçük Veri C# WebClient

Test ortalaması 17049 ms olup, açıklığın ortalamaya sapması %8,2 ve çeyrekler açıklığının ortalamaya sapması %4,5 olduğu kaydedilmiştir.



Şekil 4.6 IDES SOAP Küçük Veri C# WebClient

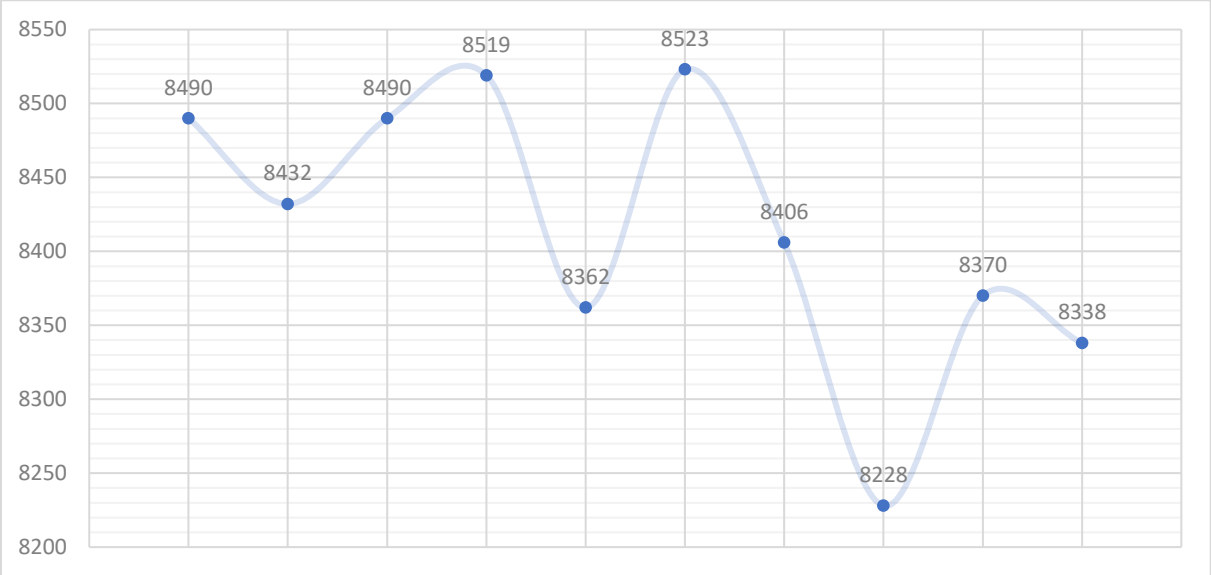
Test ortalaması 222 ms olup, açıklığın ortalamaya sapması %32 ve çeyrekler açıklığının ortalamaya sapması %9,7 olduğu kaydedilmiştir.

Java HttpURLConnection Çalışması

Bu çalışma Java programlama dilinde JDK 19 sürümünde HttpURLConnection sınıfı kullanılarak IntelliJ IDEA ortamında gerçeklenmiş ve Tablo 4.3'teki ve Şekil 4.7-Şekil 4.9 arasındaki sonuçlar elde edilmiştir. Program koduna EK 1.6'da yer verilmiştir.

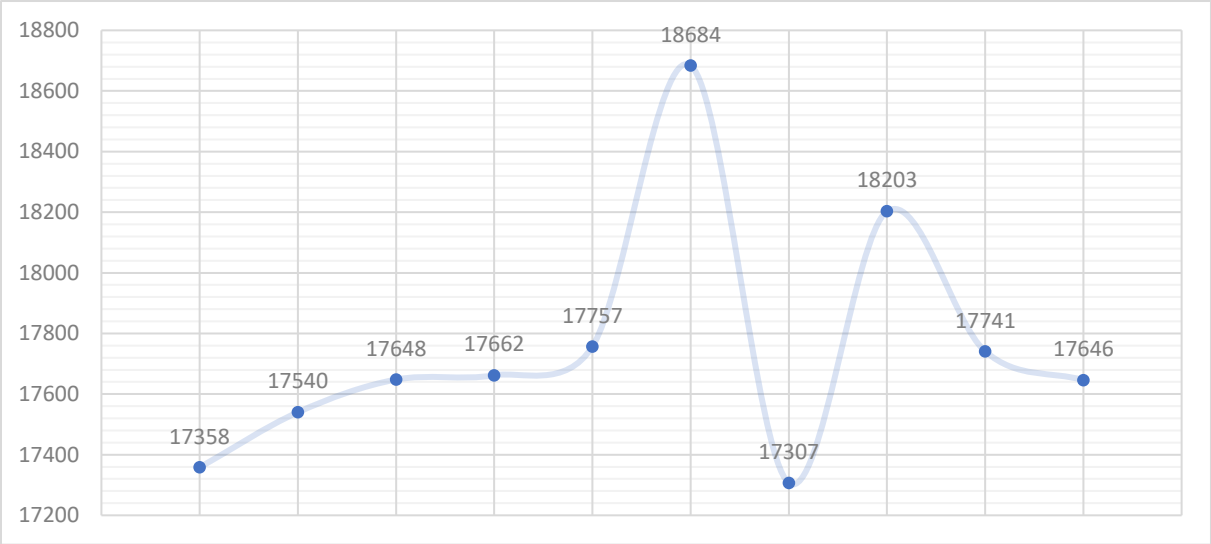
Tablo 4.3 SOAP Küçük Veri Java HttpURLConnection

Tekrar	LOCAL	S4	IDES
1	8490	17358	201
2	8432	17540	167
3	8490	17648	174
4	8519	17662	176
5	8362	17757	200
6	8523	18684	165
7	8406	17307	176
8	8228	18203	187
9	8370	17741	184
10	8338	17646	169
Ortalama	**8416**	**17755**	**180**



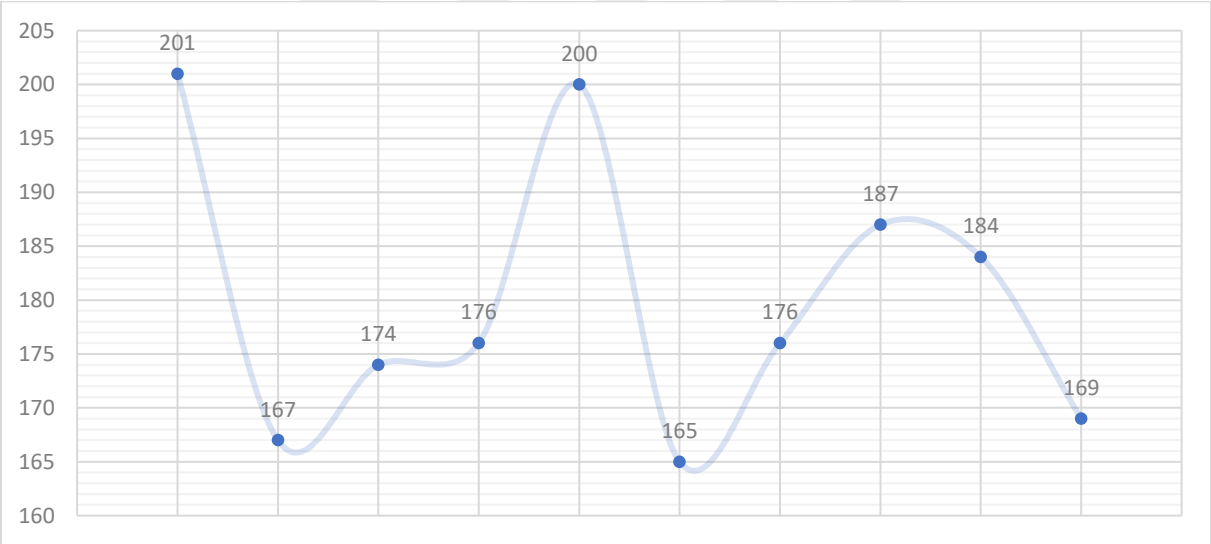
Şekil 4.7 LOCAL SOAP Küçük Veri Java HttpURLConnection

Test ortalaması 8416 ms olup, açıklığın ortalamaya sapması %3,5 ve çeyrekler açıklığının ortalamaya sapması %1,5 olduğu kaydedilmiştir.



Şekil 4.8 S4 SOAP Küçük Veri Java HttpURLConnection

Test ortalaması 17755 ms olup, açıklığın ortalamaya sapması %7,8 ve çeyrekler açıklığının ortalamaya sapması %1,1 olduğu kaydedilmiştir.



Şekil 4.9 IDES SOAP Küçük Veri Java HttpURLConnection

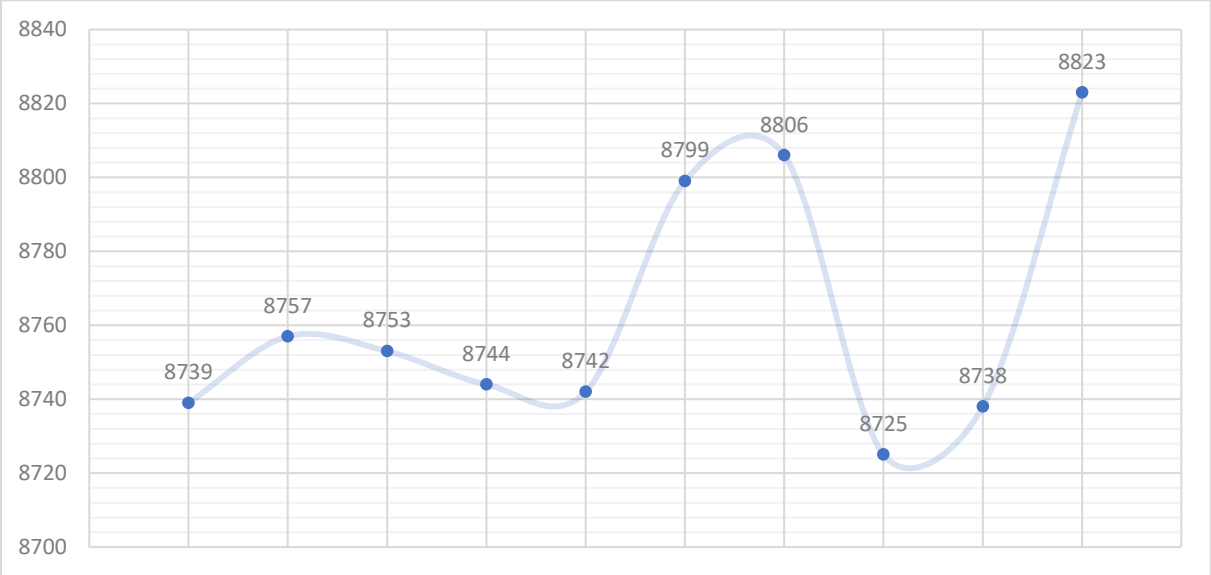
Test ortalaması 180 ms olup, açıklığın ortalamaya sapması %20 ve çeyrekler açıklığının ortalamaya sapması %8,9 olduğu kaydedilmiştir.

Java HttpClient Çalışması

Bu çalışma Java programlama dilinde JDK 19 sürümünde HttpClient sınıfı kullanılarak IntelliJ IDEA ortamında gerçeklenmiş ve Tablo 4.4'teki ve Şekil 4.10-Şekil 4.12 arasındaki sonuçlar elde edilmiştir. Program koduna EK 1.7'de yer verilmiştir.

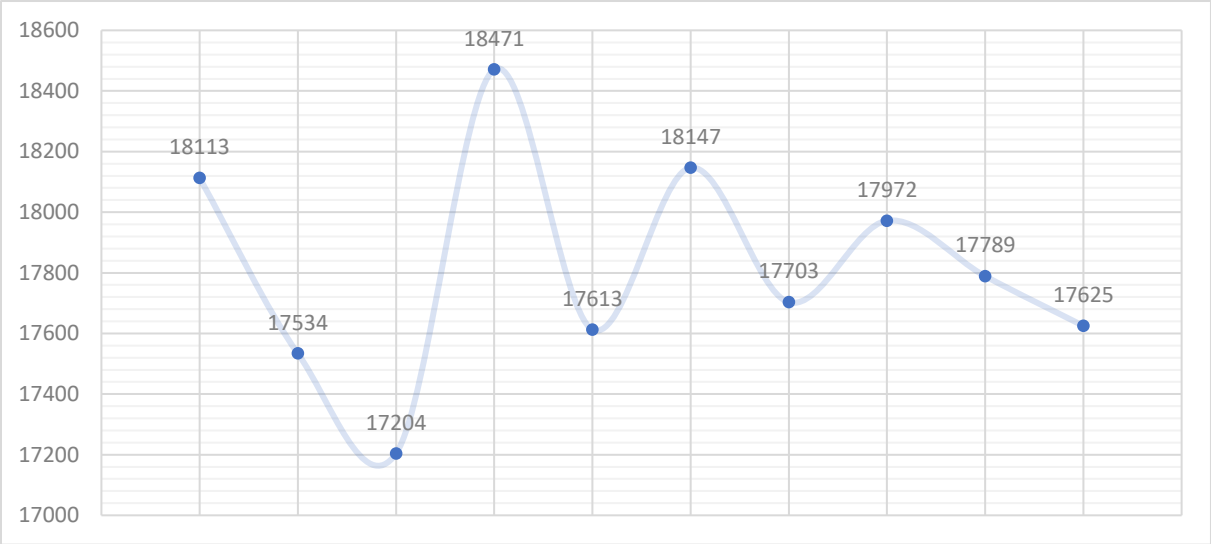
Tablo 4.4 SOAP Küçük Veri Java HttpClient

Tekrar	LOCAL	S4	IDES
1	8739	18113	423
2	8757	17534	428
3	8753	17204	437
4	8744	18471	424
5	8742	17613	504
6	8799	18147	461
7	8806	17703	485
8	8725	17972	452
9	8738	17789	444
10	8823	17625	440
Ortalama	**8763**	**17817**	**450**



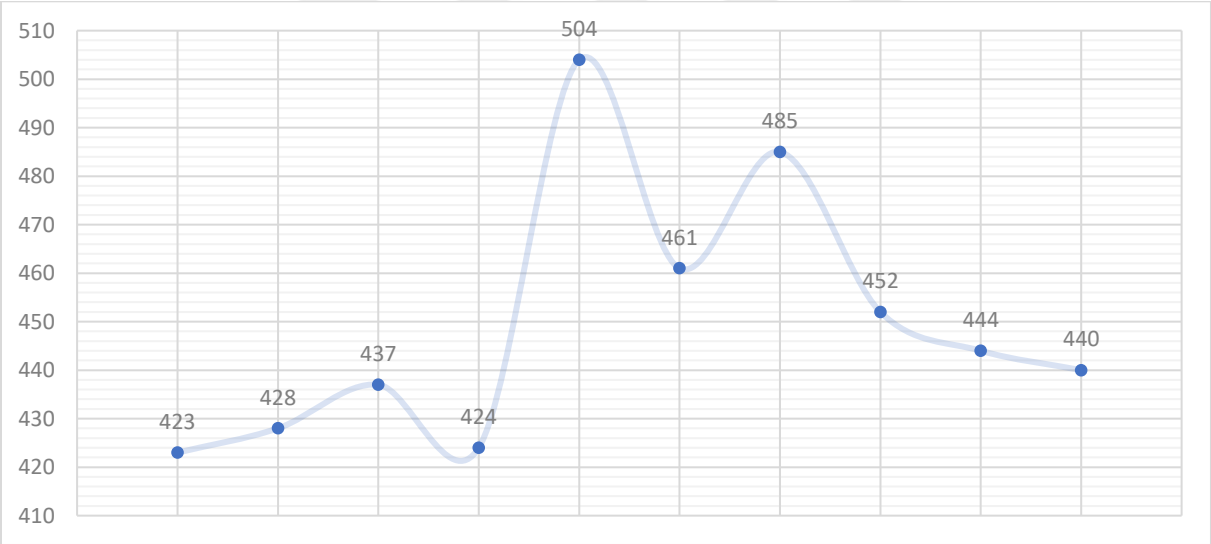
Şekil 4.10 LOCAL SOAP Küçük Veri Java HttpClient

Test ortalaması 8763 ms olup, açıklığın ortalamaya sapması %1,1 ve çeyrekler açıklığının ortalamaya sapması %0,6 olduğu kaydedilmiştir.



Şekil 4.11 S4 SOAP Küçük Veri Java HttpClient

Test ortalaması 17817 ms olup, açıklığın ortalamaya sapması %7,1 ve çeyrekler açıklığının ortalamaya sapması %2,6 olduğu kaydedilmiştir.



Şekil 4.12 IDES SOAP Küçük Veri Java HttpClient

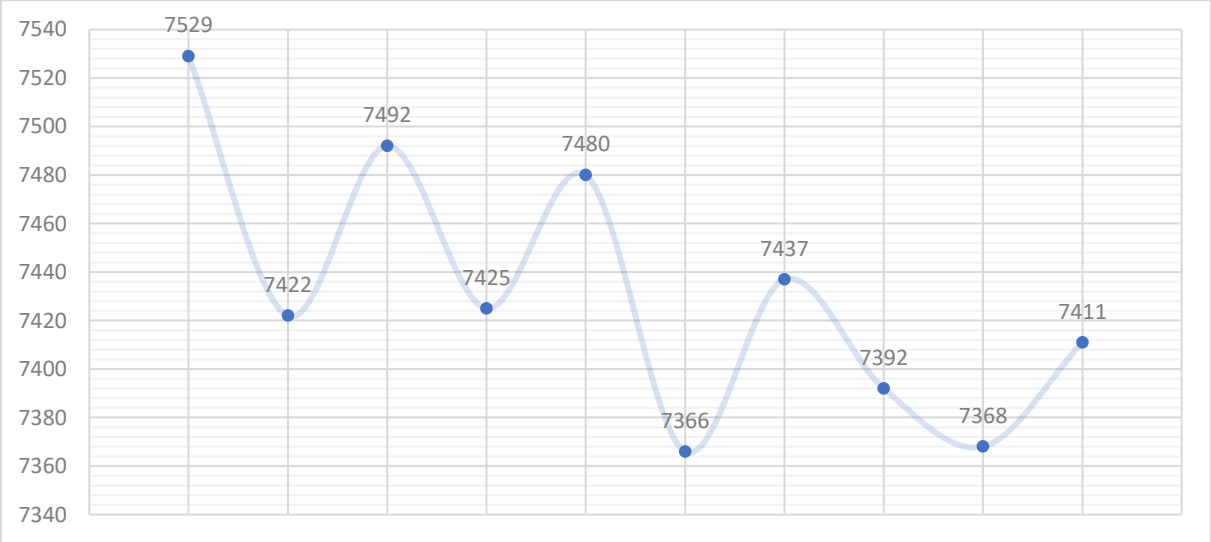
Test ortalaması 450 ms olup, açıklığın ortalamaya sapması %18 ve çeyrekler açıklığının ortalamaya sapması %6,3 olduğu kaydedilmiştir.

Node.js soap Çalışması

Bu çalışma JavaScript programlama dilinde Node.js v18 sürümünde soap kütüphanesi kullanılarak Visual Studio Code ortamında gerçeklenmiş ve Tablo 4.5'teki ve Şekil 4.13-Şekil 4.15 arasındaki sonuçlar elde edilmiştir. Program koduna EK 1.8'de yer verilmiştir.

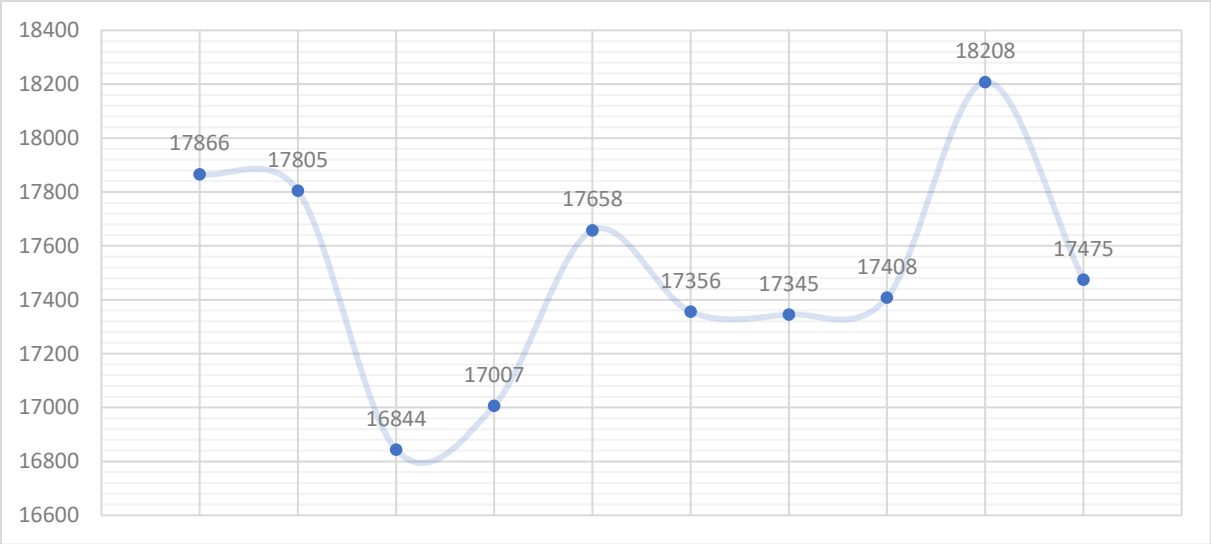
Tablo 4.5 SOAP Küçük Veri Node.js soap

Tekrar	LOCAL	S4	IDES
1	7529	17866	293
2	7422	17805	256
3	7492	16844	270
4	7425	17007	282
5	7480	17658	269
6	7366	17356	274
7	7437	17345	279
8	7392	17408	282
9	7368	18208	287
10	7411	17475	272
Ortalama	**7432**	**17497**	**276**



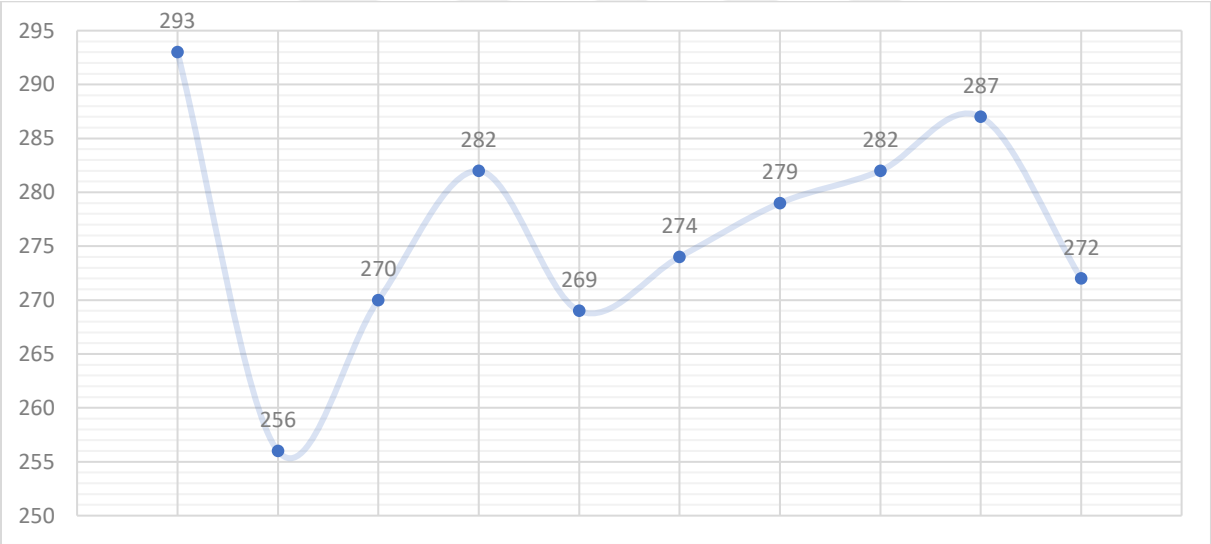
Şekil 4.13 LOCAL SOAP Küçük Veri Node.js soap

Test ortalaması 7432 ms olup, açıklığın ortalamaya sapması %2,2 ve çeyrekler açıklığının ortalamaya sapması %1,0 olduğu kaydedilmiştir.



Şekil 4.14 S4 SOAP Küçük Veri Node.js soap

Test ortalaması 17497 ms olup, açıklığın ortalamaya sapması %7,8 ve çeyrekler açıklığının ortalamaya sapması %2,4 olduğu kaydedilmiştir.



Şekil 4.15 IDES SOAP Küçük Veri Node.js soap

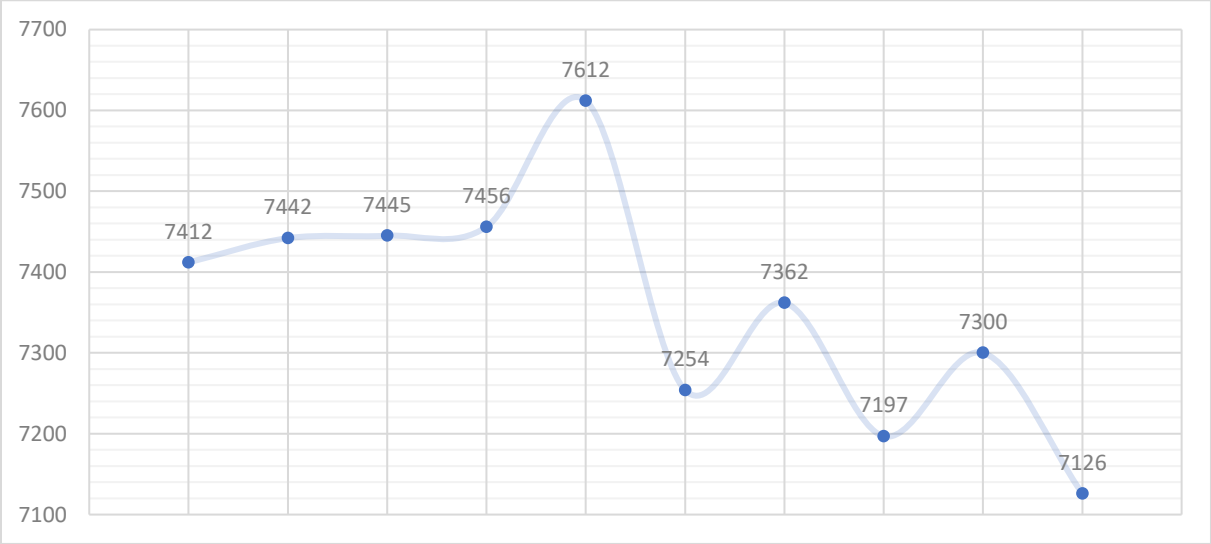
Test ortalaması 276 ms olup, açıklığın ortalamaya sapması %13,4 ve çeyrekler açıklığının ortalamaya sapması %4,2 olduğu kaydedilmiştir.

Node.js strong-soap Çalışması

Bu çalışma JavaScript programlama dilinde Node.js v18 sürümünde strong-soap kütüphanesi kullanılarak Visual Studio Code ortamında gerçeklenmiş ve Tablo 4.6'daki ve Şekil 4.16-Şekil 4.18 arasındaki sonuçlar elde edilmiştir. Program koduna EK 1.9'da yer verilmiştir.

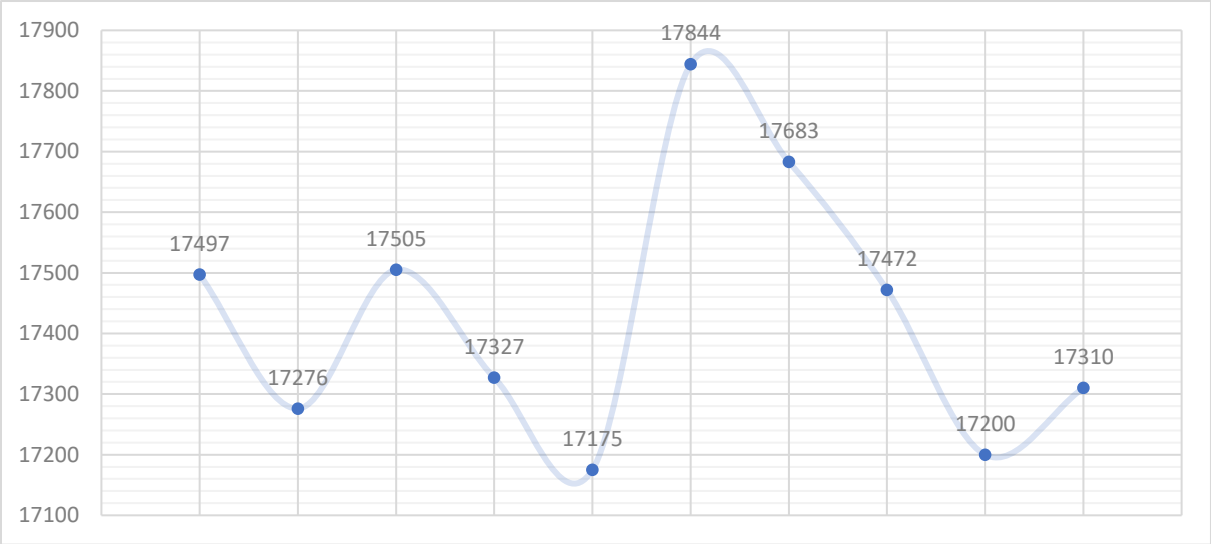
Tablo 4.6 SOAP Küçük Veri Node.js strong-soap

Tekrar	LOCAL	S4	IDES
1	7412	17497	297
2	7442	17276	285
3	7445	17505	288
4	7456	17327	282
5	7612	17175	315
6	7254	17844	292
7	7362	17683	293
8	7197	17472	292
9	7300	17200	320
10	7126	17310	324
Ortalama	**7361**	**17429**	**299**



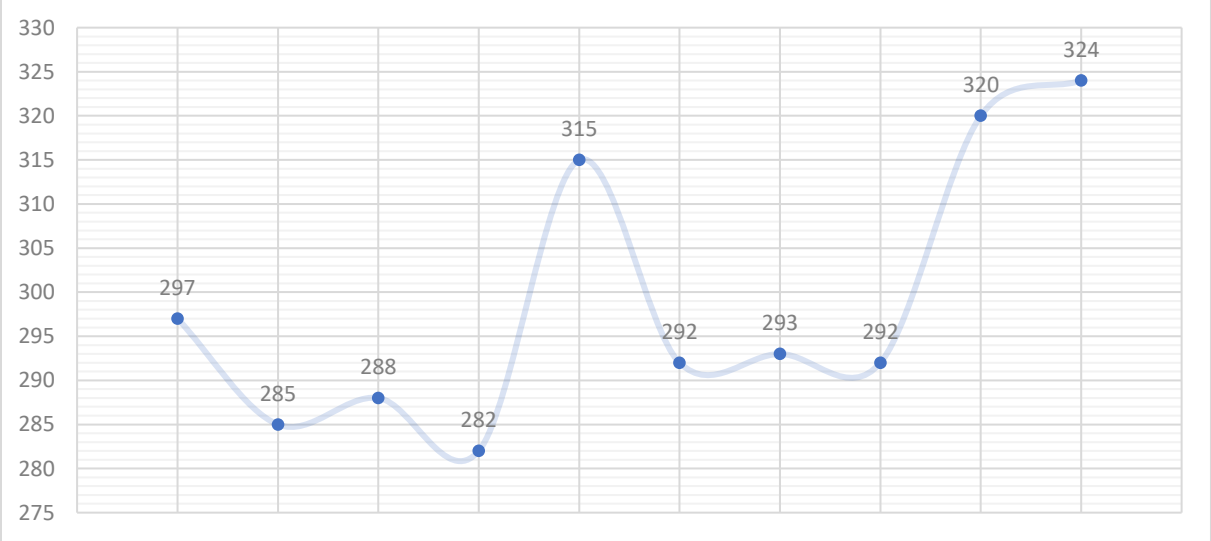
Şekil 4.16 LOCAL SOAP Küçük Veri Node.js strong-soap

Test ortalaması 7361 ms olup, açıklığın ortalamaya sapması %6,6 ve çeyrekler açıklığının ortalamaya sapması %2,4 olduğu kaydedilmiştir.



Şekil 4.17 S4 SOAP Küçük Veri Node.js strong-soap

Test ortalaması 17429 ms olup, açıklığın ortalamaya sapması %3,8 ve çeyrekler açıklığının ortalamaya sapması %1,3 olduğu kaydedilmiştir.



Şekil 4.18 IDES SOAP Küçük Veri Node.js strong-soap

Test ortalaması 299 ms olup, açıklığın ortalamaya sapması %14,1 ve çeyrekler açıklığının ortalamaya sapması %7,2 olduğu kaydedilmiştir.

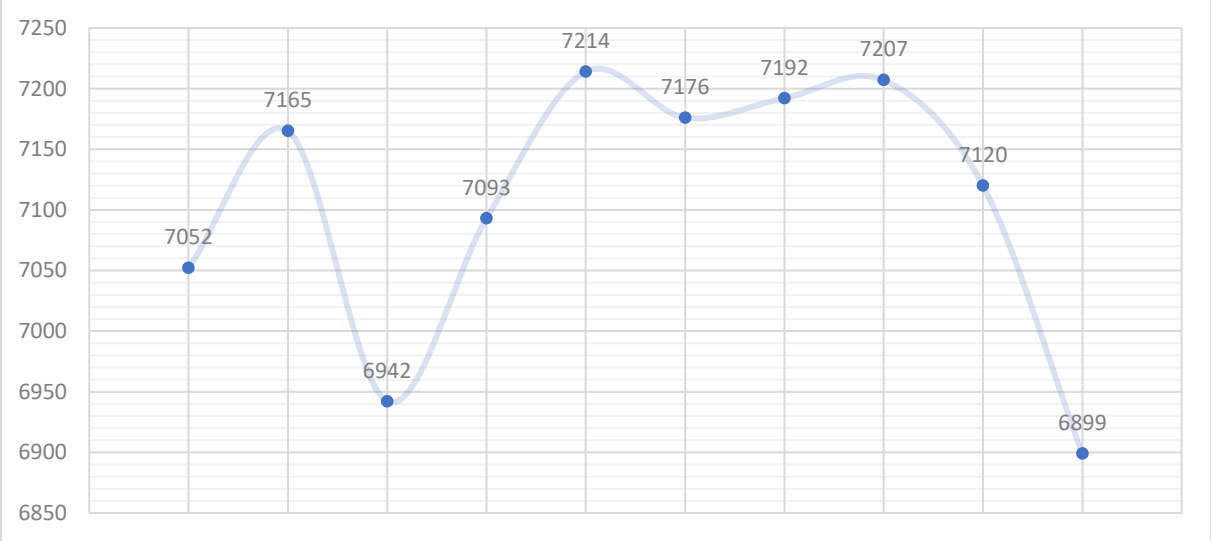
Node.js easy-soap-request Çalışması

Bu çalışma JavaScript programlama dilinde Node.js v18 sürümünde easy-soap-request kütüphanesi kullanılarak Visual Studio Code ortamında gerçeklenmiş ve Tablo 4.7'deki ve

Şekil 4.19-Şekil 4.21 arasındaki sonuçlar elde edilmiştir. Program koduna EK 1.10'da yer verilmiştir.

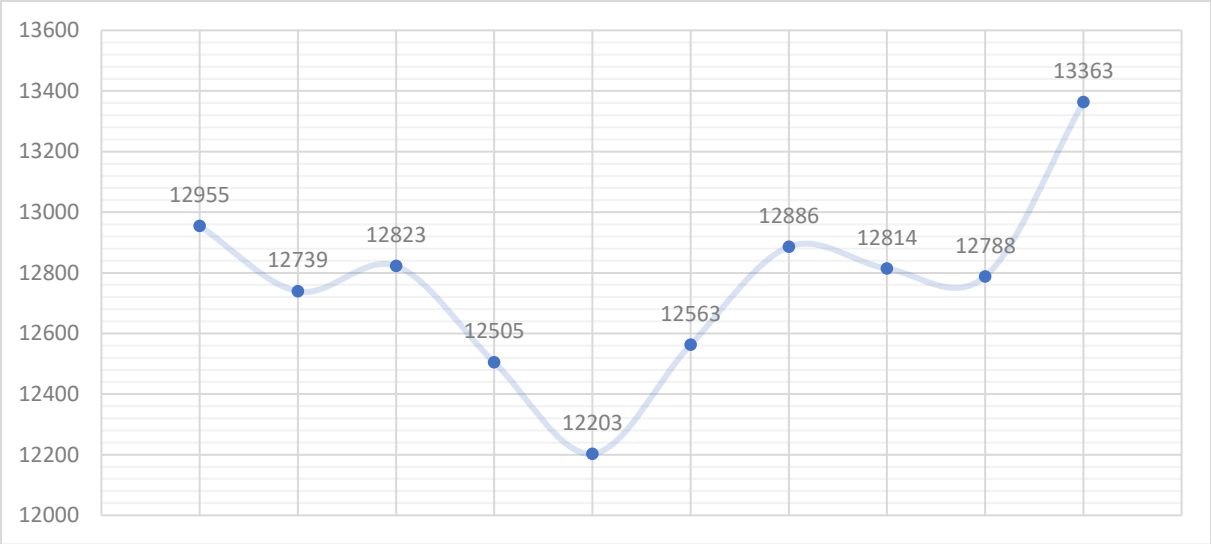
Tablo 4.7 SOAP Küçük Veri Node.js easy-soap-request

Tekrar	LOCAL	S4	IDES
1	7052	12955	166
2	7165	12739	164
3	6942	12823	170
4	7093	12505	156
5	7214	12203	153
6	7176	12563	167
7	7192	12886	183
8	7207	12814	198
9	7120	12788	200
10	6899	13363	202
Ortalama	**7106**	**12764**	**176**



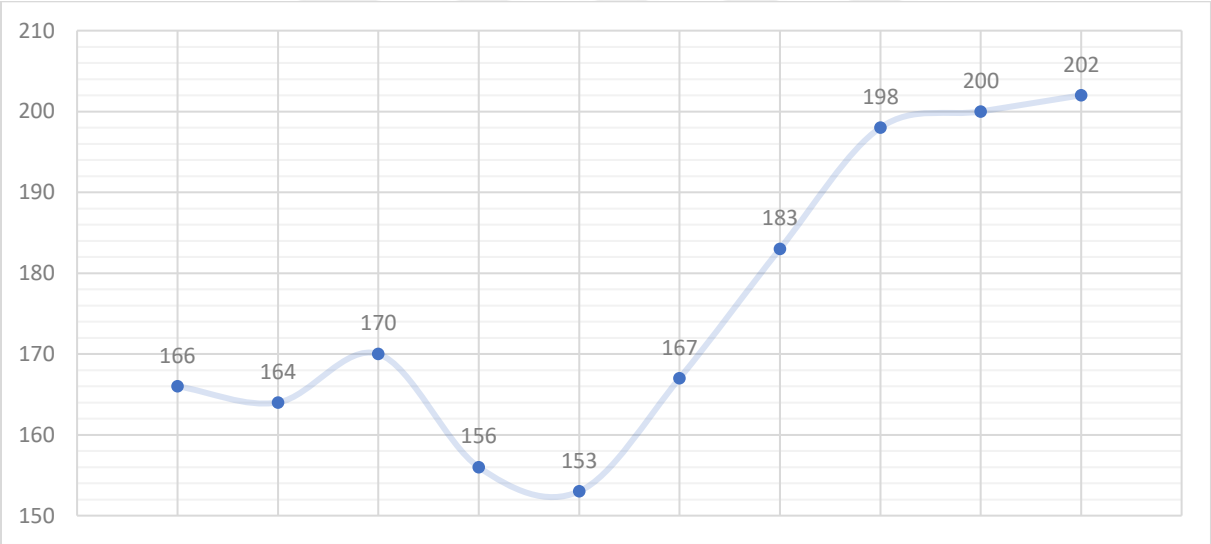
Şekil 4.19 LOCAL SOAP Küçük Veri Node.js easy-soap-request

Test ortalaması 7106 ms olup, açıklığın ortalamaya sapması %4,4 ve çeyrekler açıklığının ortalamaya sapması %1,8 olduğu kaydedilmiştir.



Şekil 4.20 S4 SOAP Küçük Veri Node.js easy-soap-request

Test ortalaması 12764 ms olup, açıklığın ortalamaya sapması %9,1 ve çeyrekler açıklığının ortalamaya sapması %2,1 olduğu kaydedilmiştir.



Şekil 4.21 IDES SOAP Küçük Veri Node.js easy-soap-request

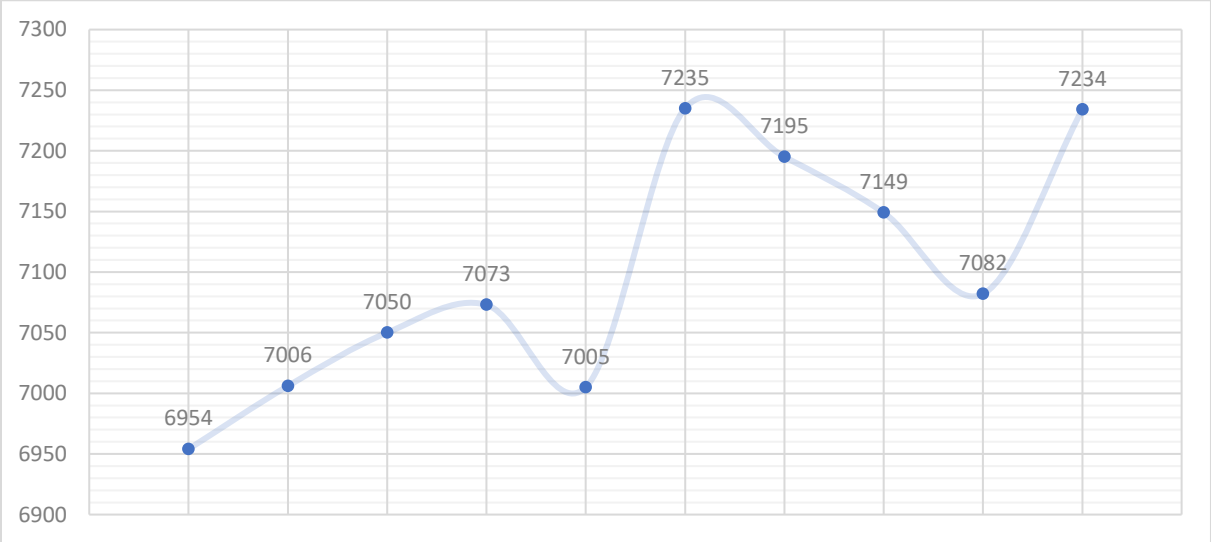
Test ortalaması 176 ms olup, açıklığın ortalamaya sapması %27,9 ve çeyrekler açıklığının ortalamaya sapması %16,9 olduğu kaydedilmiştir.

Node.js axios Çalışması

Bu çalışma JavaScript programlama dilinde Node.js v18 sürümünde axios kütüphanesi kullanılarak Visual Studio Code ortamında gerçeklenmiş ve Tablo 4.8'deki ve Şekil 4.22-Şekil 4.24 arasındaki sonuçlar elde edilmiştir. Program koduna EK 1.11'de yer verilmiştir.

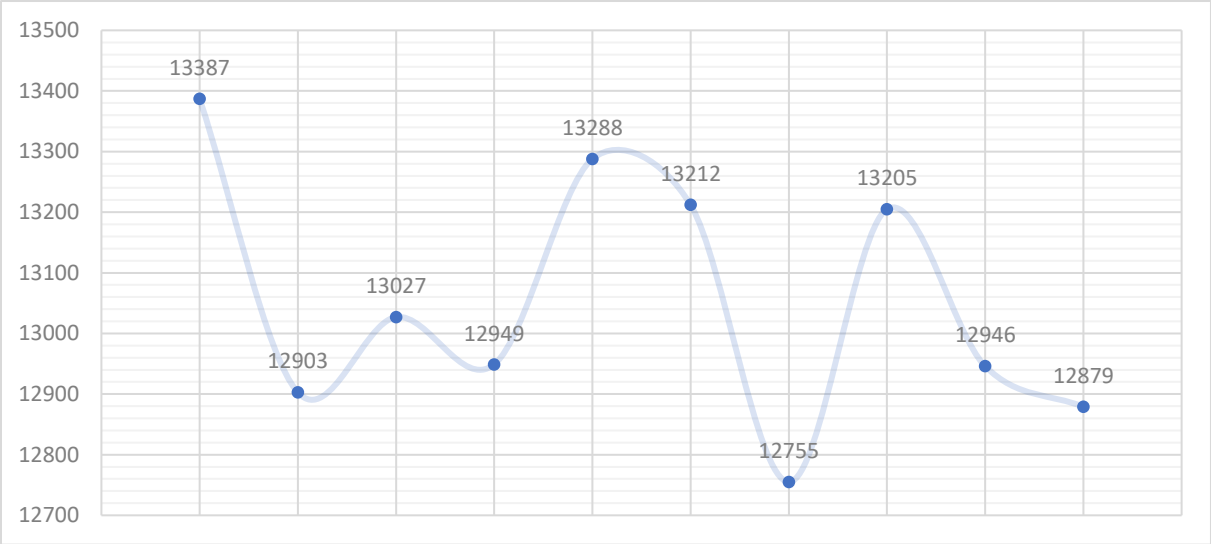
Tablo 4.8 SOAP Küçük Veri Node.js axios

Tekrar	LOCAL	S4	IDES
1	6954	13387	178
2	7006	12903	154
3	7050	13027	155
4	7073	12949	146
5	7005	13288	144
6	7235	13212	167
7	7195	12755	161
8	7149	13205	171
9	7082	12946	182
10	7234	12879	218
Ortalama	**7098**	**13055**	**168**



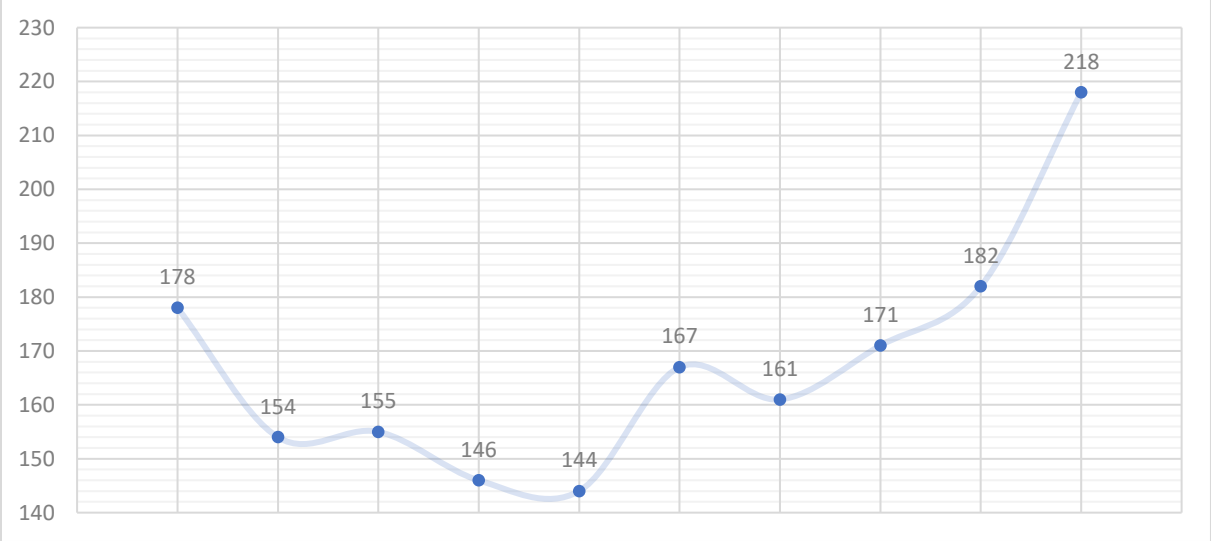
Şekil 4.22 LOCAL SOAP Küçük Veri Node.js axios

Test ortalaması 7098 ms olup, açıklığın ortalamaya sapması %4,0 ve çeyrekler açıklığının ortalamaya sapması %2,3 olduğu kaydedilmiştir.



Şekil 4.23 S4 SOAP Küçük Veri Node.js axios

Test ortalaması 13055 ms olup, açıklığın ortalamaya sapması %4,8 ve çeyrekler açıklığının ortalamaya sapması %2,3 olduğu kaydedilmiştir.



Şekil 4.24 IDES SOAP Küçük Veri Node.js axios

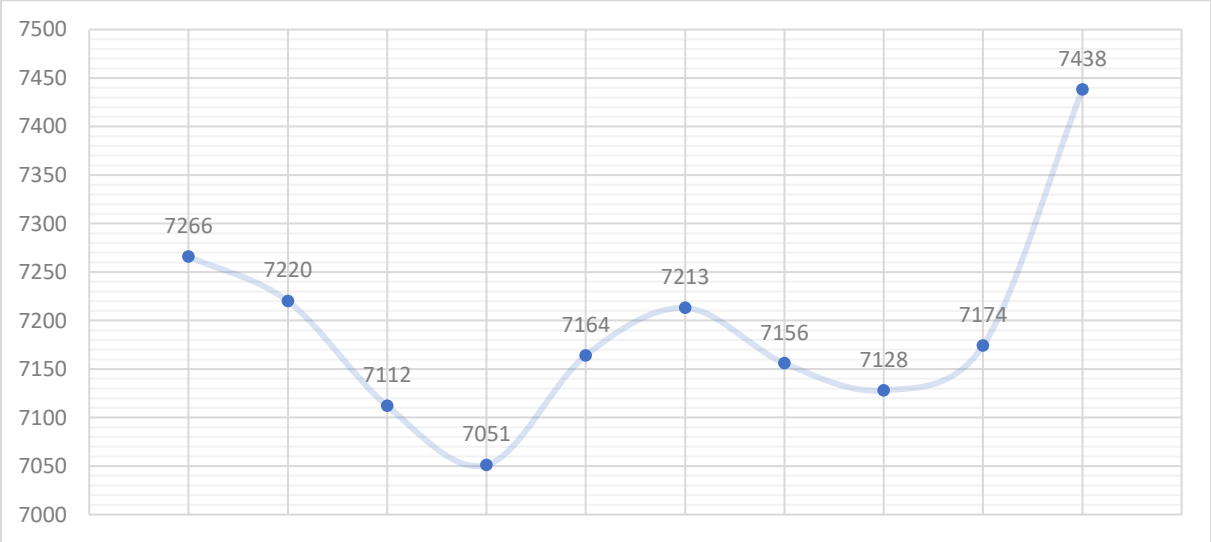
Test ortalaması 168 ms olup, açıklığın ortalamaya sapması %44,2 ve çeyrekler açıklığının ortalamaya sapması %13,1 olduğu kaydedilmiştir.

Node.js request Çalışması

Bu çalışma JavaScript programlama dilinde Node.js v18 sürümünde request kütüphanesi kullanılarak Visual Studio Code ortamında gerçeklenmiş ve Tablo 4.9'daki ve Şekil 4.25-Şekil 4.27 arasındaki sonuçlar elde edilmiştir. Program koduna EK 1.12'de yer verilmiştir.

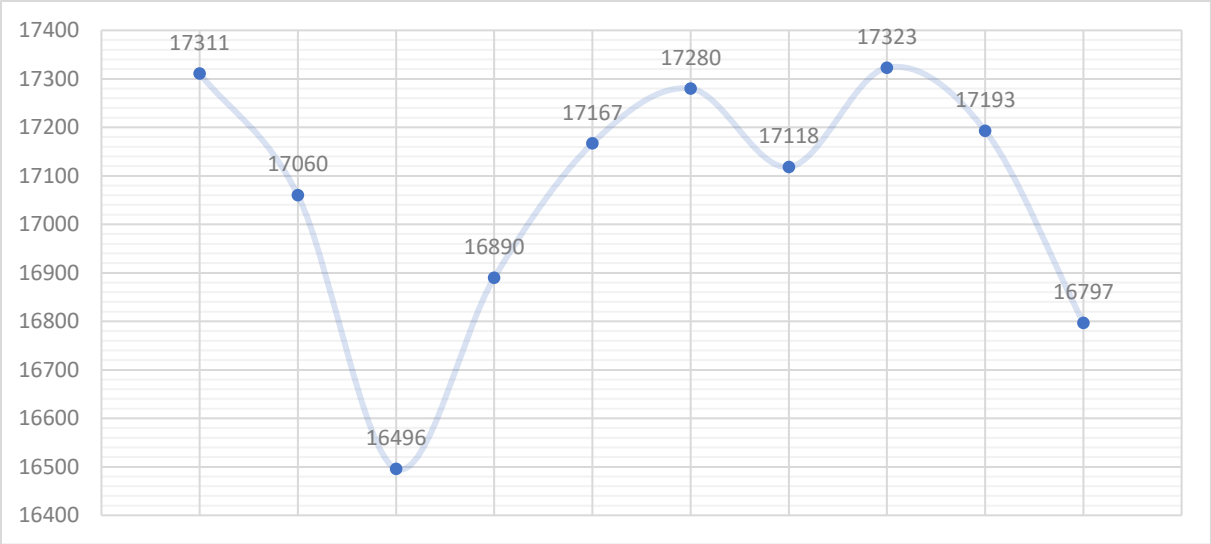
Tablo 4.9 SOAP Küçük Veri Node.js request

Tekrar	LOCAL	S4	IDES
1	7266	17311	148
2	7220	17060	162
3	7112	16496	139
4	7051	16890	152
5	7164	17167	142
6	7213	17280	154
7	7156	17118	151
8	7128	17323	150
9	7174	17193	145
10	7438	16797	153
Ortalama	**7192**	**17064**	**150**



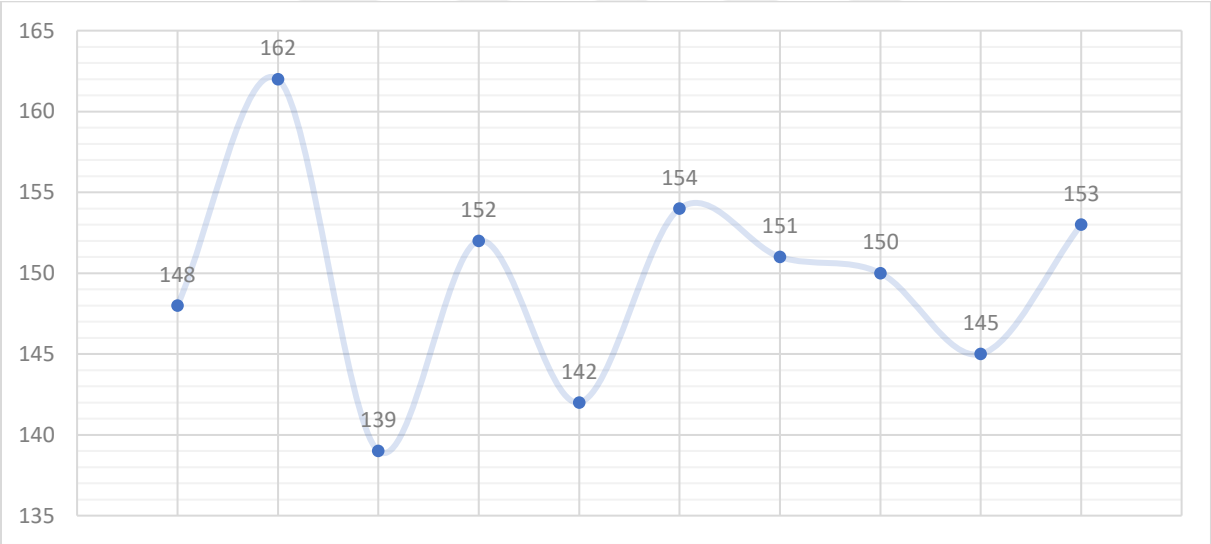
Şekil 4.25 LOCAL SOAP Küçük Veri Node.js request

Test ortalaması 7192 ms olup, açıklığın ortalamaya sapması %5,4 ve çeyrekler açıklığının ortalamaya sapması %1,2 olduğu kaydedilmiştir.



Şekil 4.26 S4 LOCAL SOAP Küçük Veri Node.js request

Test ortalaması 17064 ms olup, açıklığın ortalamaya sapması %4,8 ve çeyrekler açıklığının ortalamaya sapması %1,9 olduğu kaydedilmiştir.



Şekil 4.27 IDES LOCAL SOAP Küçük Veri Node.js request

Test ortalaması 150 ms olup, açıklığın ortalamaya sapması %15,4 ve çeyrekler açıklığının ortalamaya sapması %4,7 olduğu kaydedilmiştir.

4.1.2. Küçük Veriyle REST JSON Uygulamaları

Bu tez çalışmasında ikinci olarak JSON formatındaki küçük verilerle REST API çalışılmıştır. Test çeşitliliği ve sonuçların doğru yorumlanabilmesi için üç farklı sunucuyla çalışılmış ve

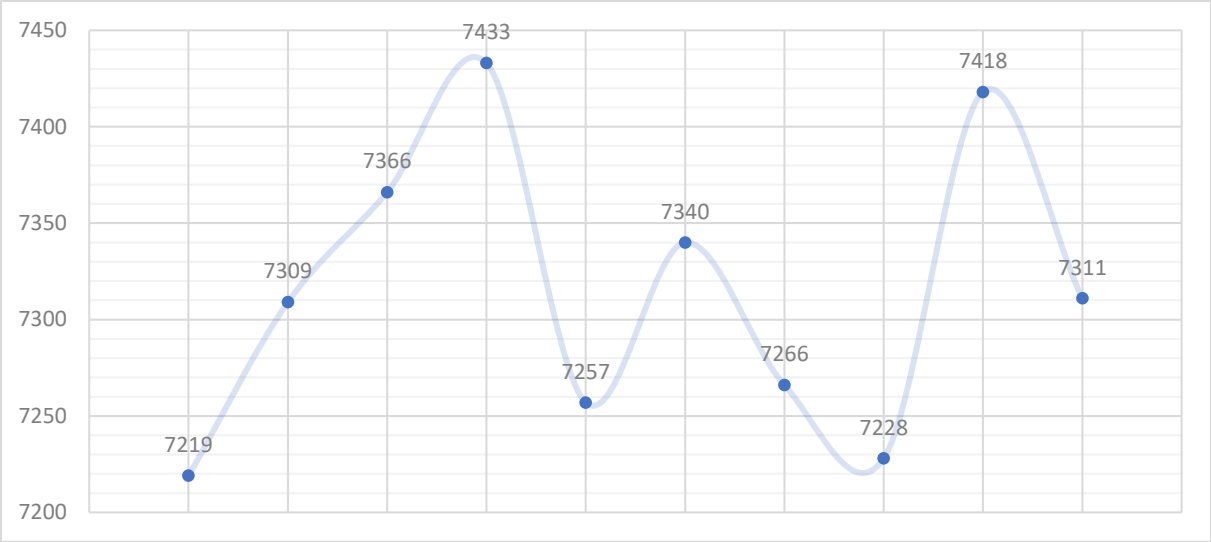
sunuculardaki veri büyüklükleri sırasıyla IDES ile LOCAL arasında yaklaşık 100 kat, LOCAL ile S4 arasında yaklaşık 5 kat büyüklük farkı olacak şeklindedir. Üç farklı programlama dili kullanılıp bunlar sırasıyla C#, Java ve JavaScript(Node.js) dilleri olmuştur. Bu programlama dillerinin de kendi içlerinde farklı kütüphaneler ve sınıflar kullanılarak test sonuçları her programlama dili içerisinde çeşitlendirilmiştir. Her bir program uygun olduğu geliştirme ortamında geliştirilip derlenmiş ve çalıştırılmıştır. Her bir çalıştırma döngüsü kendi içerisinde üç ana başlıktan oluşmaktadır; istek paketlerinin oluşturulması, gönderilmesi ve cevabın alınması, cevabın yorumlanması. Bu üç başlığın toplam çalışma süresi milisaniye(ms) biriminden kaydedilmiş ve tez çalışması kapsamında yorumlanmıştır. Bu tez çalışması kapsamında yorumlanan veriler; programların toplam çalışma süreleri değil, belirtilen bu üç ana başlıkta geçirilen sürelerdir. SAP sunucusunun REST API ile JSON formatında döndürdüğü cevaba EK 2.1'de yer verilmiştir. Kullanılan farklı diller ve bu dillerde kullanılan farklı kütüphaneler veya sınıflar REST isteklerini farklı yöntemlerle oluşturup, gönderip, EK 2.1'de yer verilen aynı cevabı almıştır. Bu cevap; SAP tarafında veritabanının farklı alanlarından SQL sorguları ve ABAP fonksiyonlarıyla çekilmiş, paketlenmiş ve gönderilmiştir. SAP ABAP program koduna EK 4.2'de yer verilmiştir.

C# HttpClient Çalışması

Bu çalışma C# programlama dilinde .NET 7.0 platformunda HttpClient sınıfı kullanılarak Visual Studio ortamında gerçeklenmiş ve Tablo 4.10'daki ve Şekil 4.28-Şekil 4.30 arasındaki sonuçlar elde edilmiştir. Program koduna EK 2.2'de yer verilmiştir.

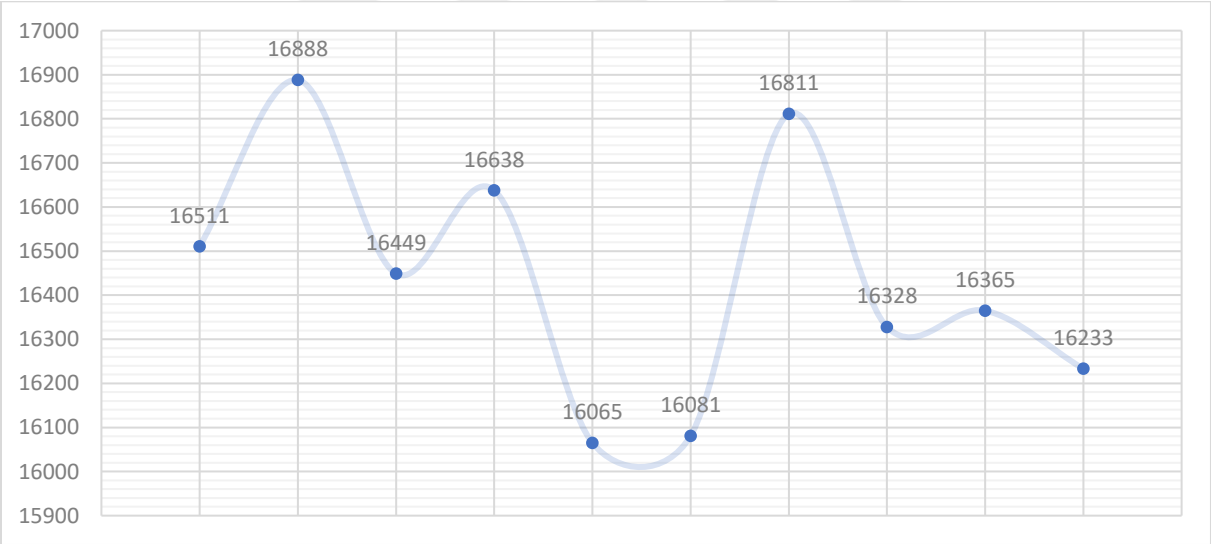
Tablo 4.10 REST JSON Küçük Veri C# HttpClient

Tekrar	LOCAL	S4	IDES
1	7219	16511	255
2	7309	16888	244
3	7366	16449	224
4	7433	16638	230
5	7257	16065	293
6	7340	16081	319
7	7266	16811	280
8	7228	16328	292
9	7418	16365	347
10	7311	16233	364
Ortalama	**7315**	**16437**	**285**



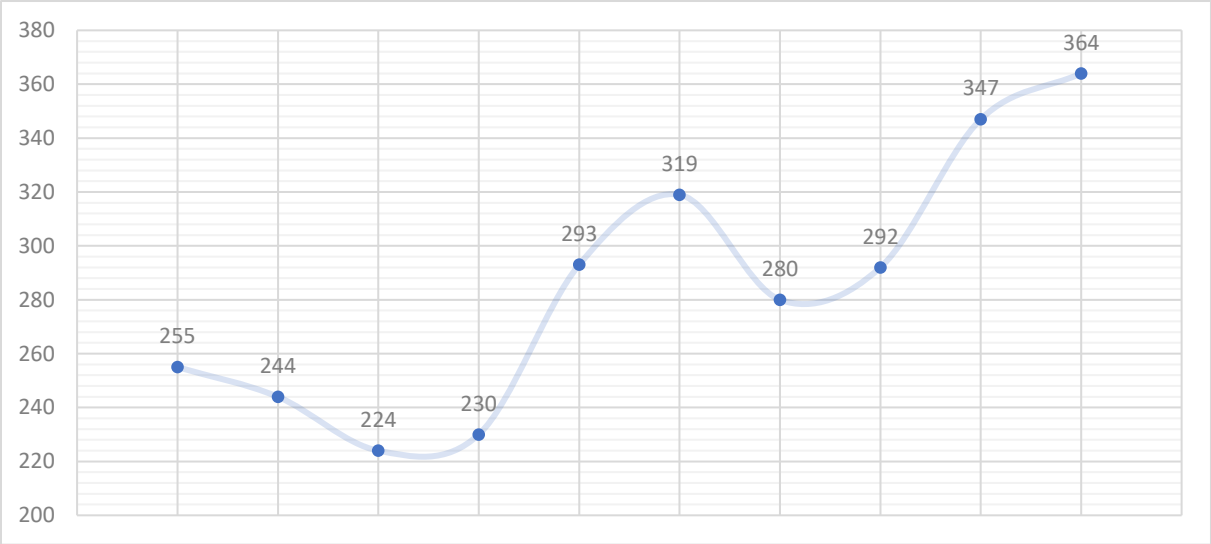
Şekil 4.28 LOCAL REST JSON Küçük Veri C# HttpClient

Test ortalaması 7315 ms olup, açıklığın ortalamaya sapması %2,9 ve çeyrekler açıklığının ortalamaya sapması %1,4 olduğu kaydedilmiştir.



Şekil 4.29 S4 REST JSON Küçük Veri C# HttpClient

Test ortalaması 16437 ms olup, açıklığın ortalamaya sapması %5 ve çeyrekler açıklığının ortalamaya sapması %2,1 olduğu kaydedilmiştir.



Şekil 4.30 IDES REST JSON Küçük Veri C# HttpClient

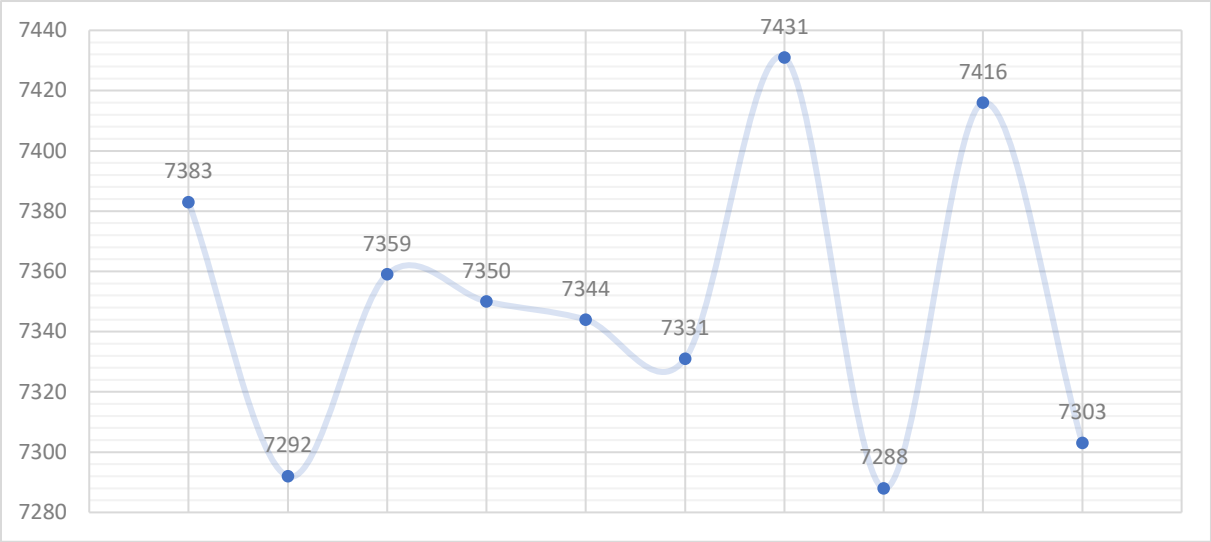
Test ortalaması 285 ms olup, açıklığın ortalamaya sapması %49,2 ve çeyrekler açıklığının ortalamaya sapması %23,1 olduğu kaydedilmiştir.

C# RestSharp Çalışması

Bu çalışma C# programlama dilinde .NET 7.0 platformunda RestSharp sınıfı kullanılarak Visual Studio ortamında gerçeklenmiş ve Tablo 4.11'deki ve Şekil 4.31-Şekil 4.33 arasındaki sonuçlar elde edilmiştir. Program koduna EK 2.3'te yer verilmiştir.

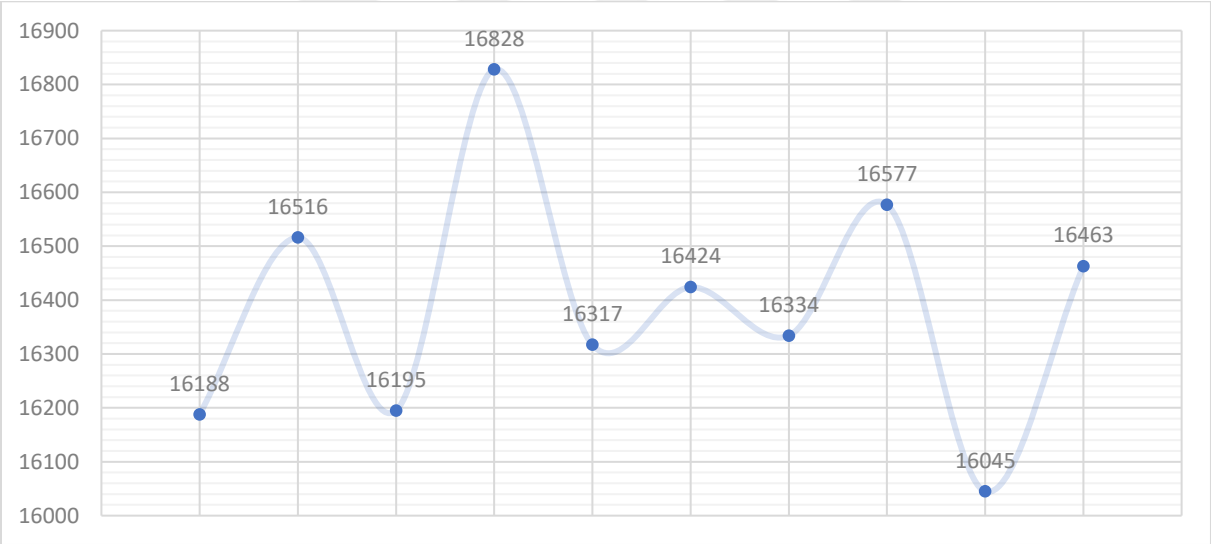
Tablo 4.11 REST JSON Küçük Veri C# RestSharp

Tekrar	LOCAL	S4	IDES
1	7383	16188	300
2	7292	16516	359
3	7359	16195	290
4	7350	16828	303
5	7344	16317	293
6	7331	16424	288
7	7431	16334	286
8	7288	16577	265
9	7416	16045	275
10	7303	16463	288
Ortalama	**7350**	**16389**	**295**



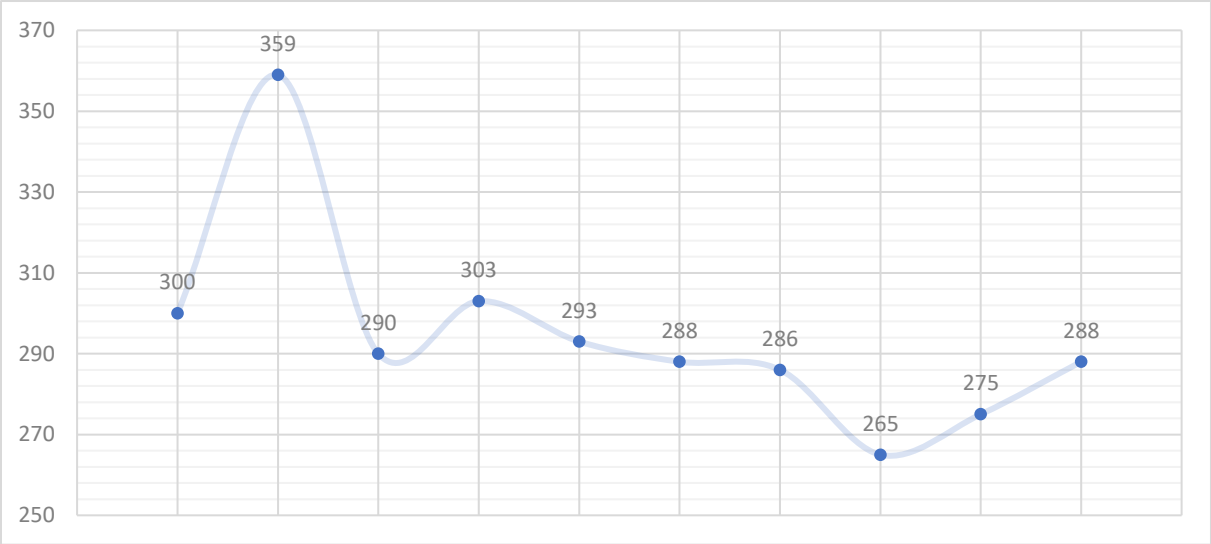
Şekil 4.31 LOCAL REST JSON Küçük Veri C# RestSharp

Test ortalaması 7350 ms olup, açıklığın ortalamaya sapması %1,9 ve çeyrekler açıklığının ortalamaya sapması %0,9 olduğu kaydedilmiştir.



Şekil 4.32 S4 REST JSON Küçük Veri C# RestSharp

Test ortalaması 16389 ms olup, açıklığın ortalamaya sapması %4,8 ve çeyrekler açıklığının ortalamaya sapması %1,7 olduğu kaydedilmiştir.



Şekil 4.33 IDES REST JSON Küçük Veri C# RestSharp

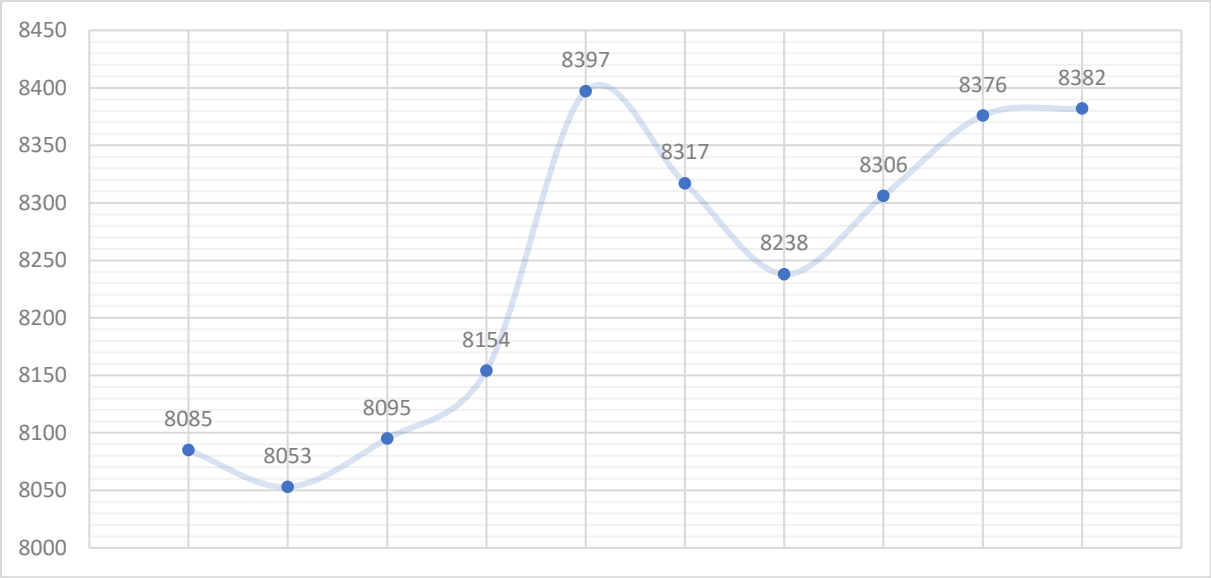
Test ortalaması 295 ms olup, açıklığın ortalamaya sapması %31,9 ve çeyrekler açıklığının ortalamaya sapması %4 olduğu kaydedilmiştir.

Java HttpURLConnection Çalışması

Bu çalışma Java programlama dilinde JDK 19 sürümünde HttpURLConnection sınıfı kullanılarak IntelliJ IDEA ortamında gerçeklenmiş ve Tablo 4.12'deki ve Şekil 4.34-Şekil 4.36 arasındaki sonuçlar elde edilmiştir. Program koduna EK 2.4'te yer verilmiştir.

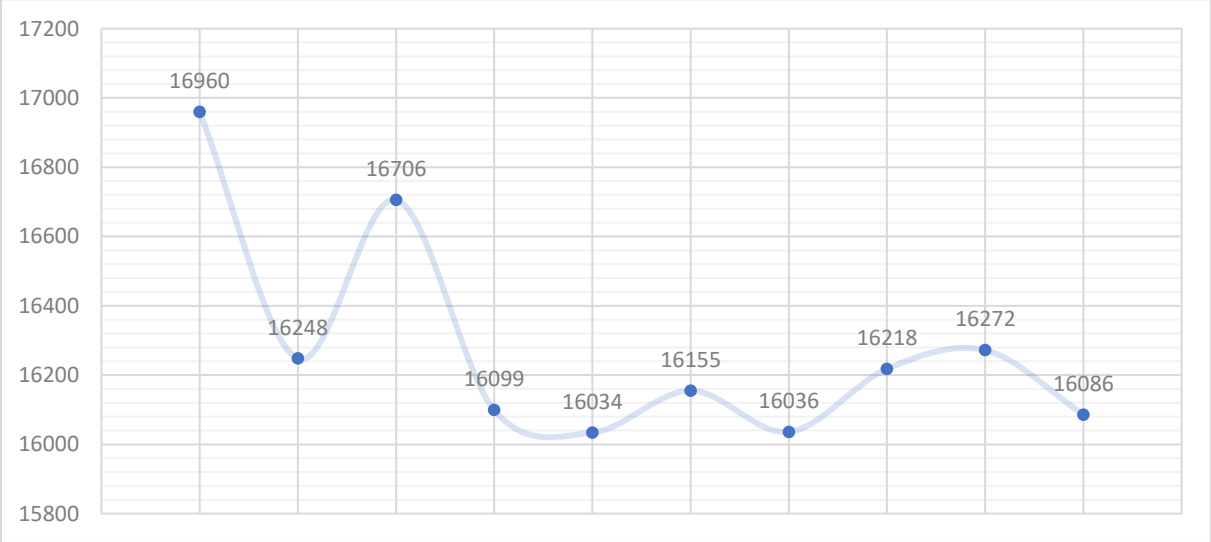
Tablo 4.12 REST JSON Küçük Veri Java HttpURLConnection

Tekrar	LOCAL	S4	IDES
1	8085	16960	180
2	8053	16248	170
3	8095	16706	181
4	8154	16099	179
5	8397	16034	180
6	8317	16155	178
7	8238	16036	144
8	8306	16218	172
9	8376	16272	186
10	8382	16086	190
Ortalama	**8240**	**16281**	**176**



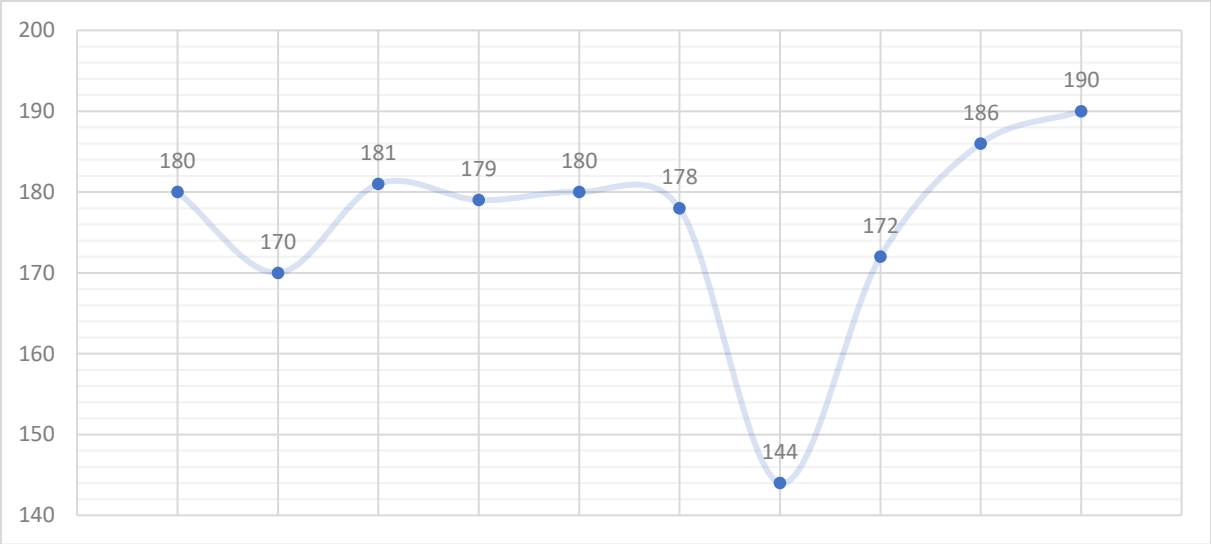
Şekil 4.34 LOCAL REST JSON Küçük Veri Java HttpURLConnection

Test ortalaması 8240 ms olup, açıklığın ortalamaya sapması %4,2 ve çeyrekler açıklığının ortalamaya sapması %3,1 olduğu kaydedilmiştir.



Şekil 4.35 S4 REST JSON Küçük Veri Java HttpURLConnection

Test ortalaması 16281 ms olup, açıklığın ortalamaya sapması %5,7 ve çeyrekler açıklığının ortalamaya sapması %1,1 olduğu kaydedilmiştir.



Şekil 4.36 IDES REST JSON Küçük Veri Java HttpURLConnection

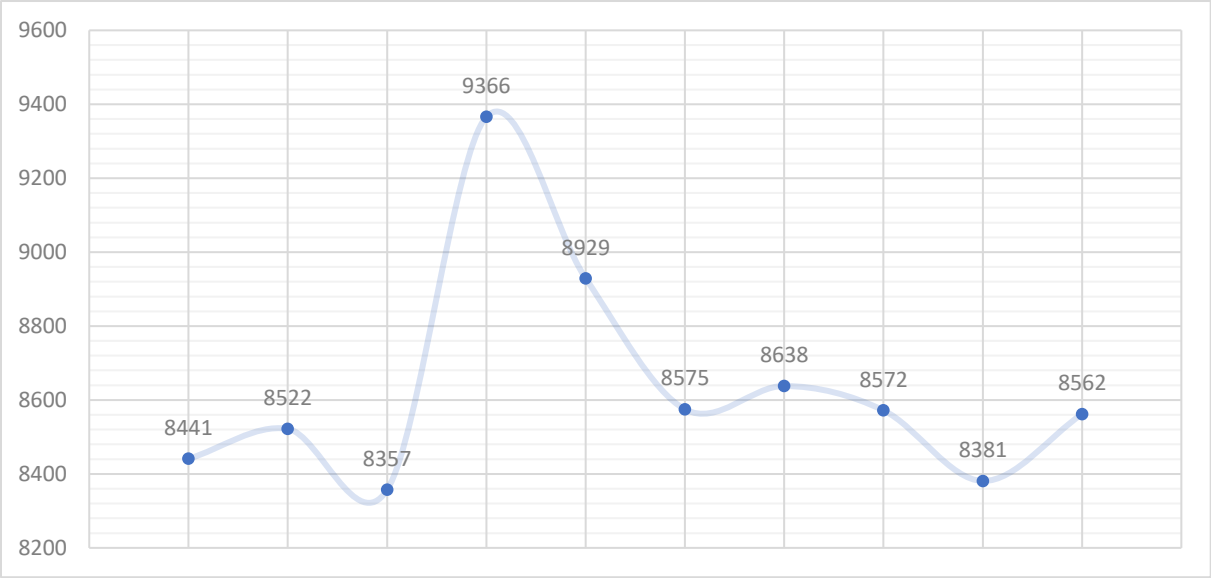
Test ortalaması 176 ms olup, açıklığın ortalamaya sapması %26,1 ve çeyrekler açıklığının ortalamaya sapması %4,1 olduğu kaydedilmiştir.

Java HttpClient Çalışması

Bu çalışma Java programlama dilinde JDK 19 sürümünde HttpClient sınıfı kullanılarak IntelliJ IDEA ortamında gerçeklenmiş ve Tablo 4.13'teki ve Şekil 4.37-Şekil 4.39 arasındaki sonuçlar elde edilmiştir. Program koduna EK 2.5'te yer verilmiştir.

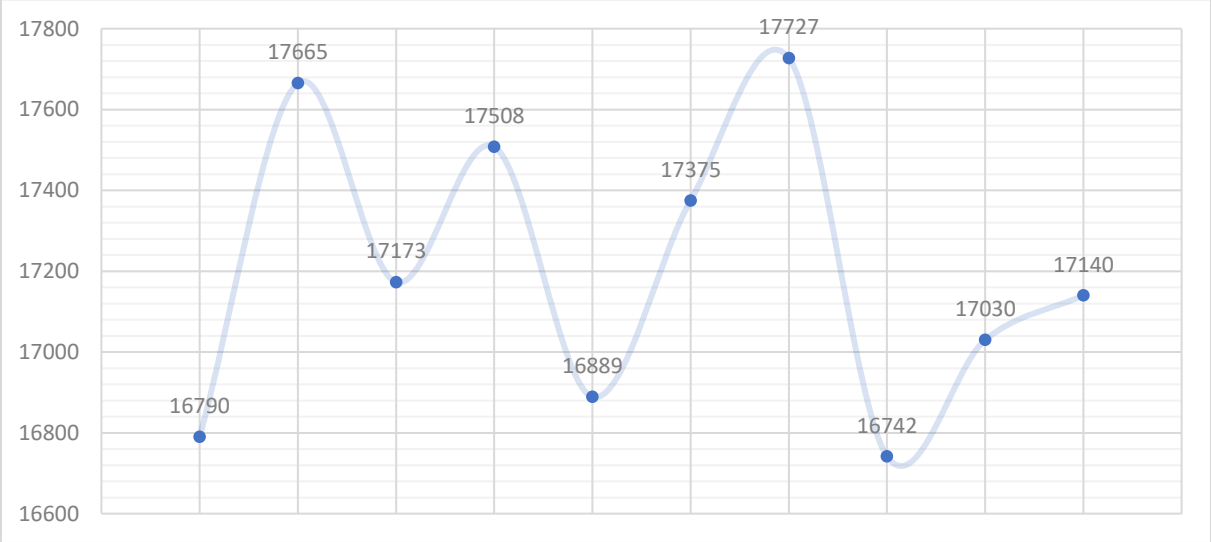
Tablo 4.13 REST JSON Küçük Veri Java HttpClient

Tekrar	LOCAL	S4	IDES
1	8441	16790	487
2	8522	17665	478
3	8357	17173	463
4	9366	17508	447
5	8929	16889	429
6	8575	17375	435
7	8638	17727	431
8	8572	16742	453
9	8381	17030	485
10	8562	17140	432
Ortalama	**8634**	**17204**	**454**



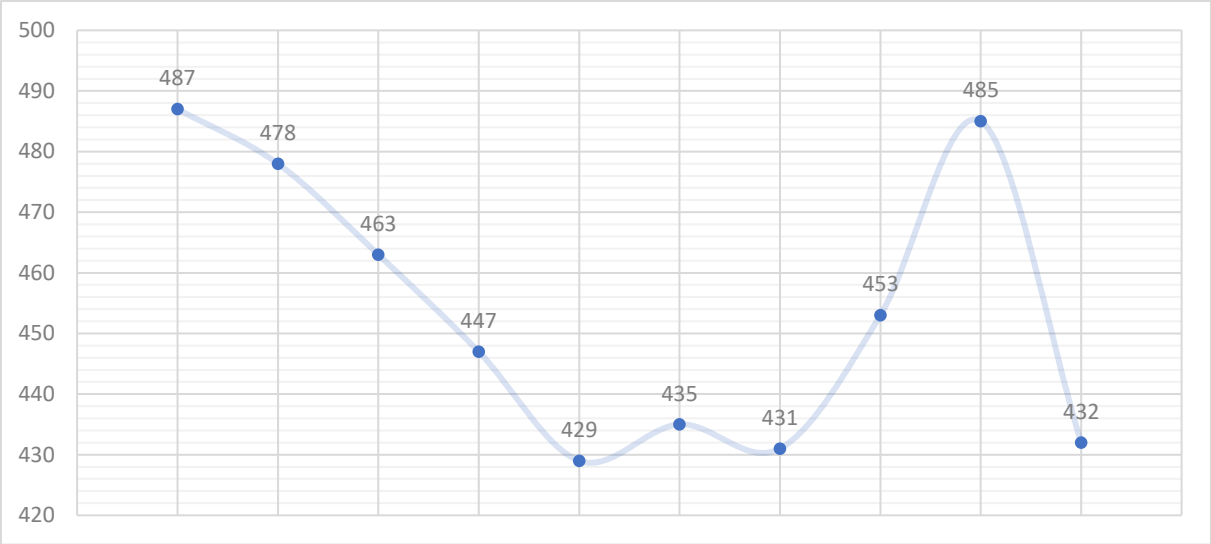
Şekil 4.37 LOCAL REST JSON Küçük Veri Java HttpClient

Test ortalaması 8634 ms olup, açıklığın ortalamaya sapması %11,7 ve çeyrekler açıklığının ortalamaya sapması %1,9 olduğu kaydedilmiştir.



Şekil 4.38 S4 REST JSON Küçük Veri Java HttpClient

Test ortalaması 17204 ms olup, açıklığın ortalamaya sapması %5,7 ve çeyrekler açıklığının ortalamaya sapması %3,2 olduğu kaydedilmiştir.



Şekil 4.39 IDES REST JSON Küçük Veri Java HttpClient

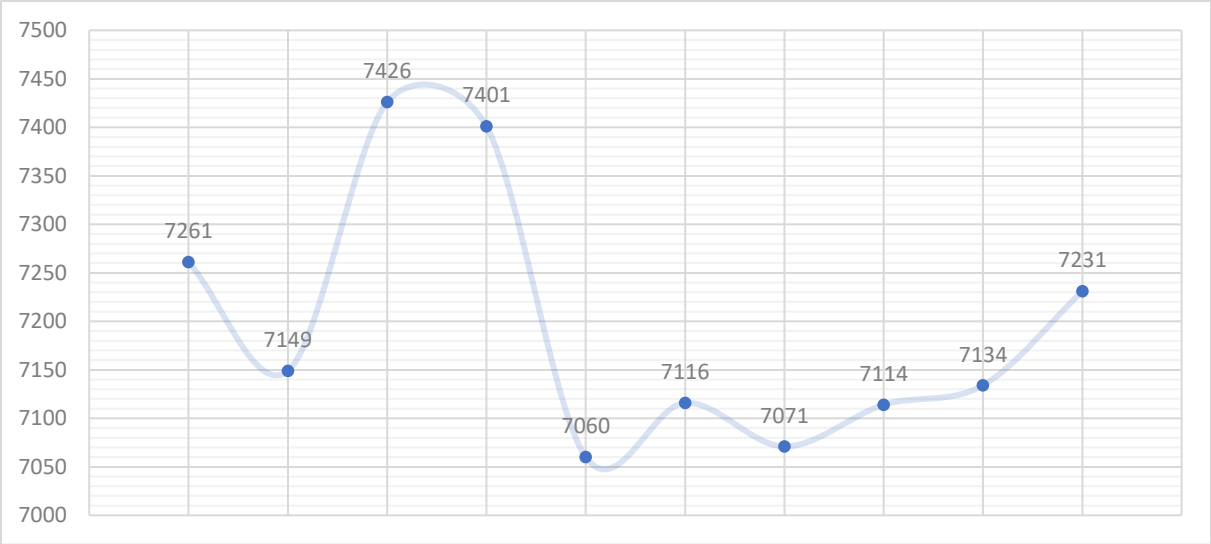
Test ortalaması 454 ms olup, açıklığın ortalamaya sapması %12,8 ve çeyrekler açıklığının ortalamaya sapması %9,1 olduğu kaydedilmiştir.

Node.js request Çalışması

Bu çalışma JavaScript programlama dilinde Node.js v18 sürümünde request kütüphanesi kullanılarak Visual Studio Code ortamında gerçeklenmiş ve Tablo 4.14'teki ve Şekil 4.40-Şekil 4.42 arasındaki sonuçlar elde edilmiştir. Program koduna EK 2.6'da yer verilmiştir.

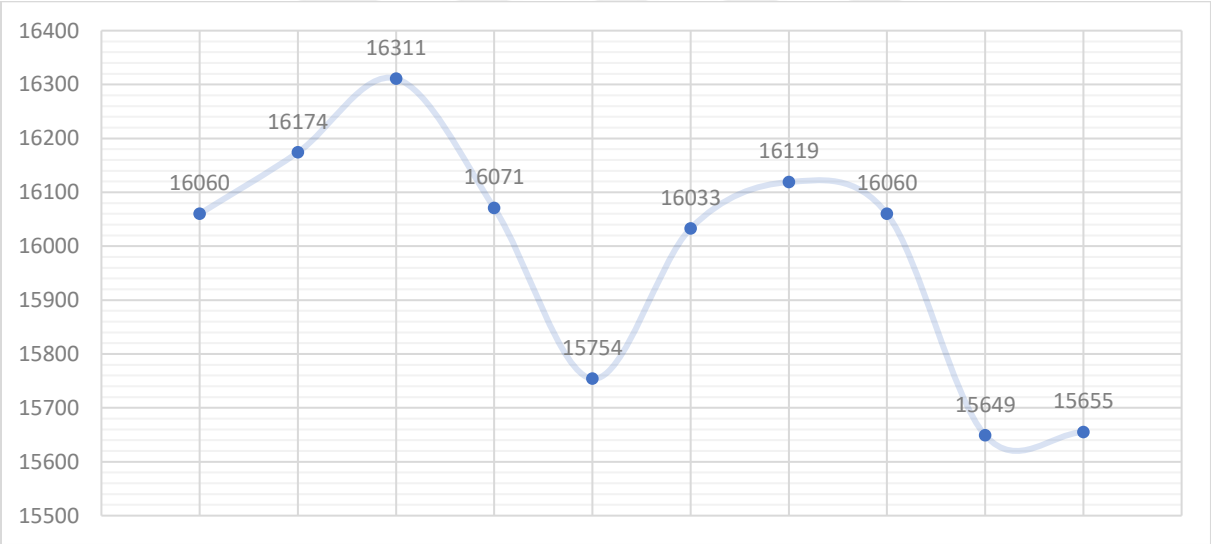
Tablo 4.14 REST JSON Küçük Veri Node.js request

Tekrar	LOCAL	S4	IDES
1	7261	16060	140
2	7149	16174	128
3	7426	16311	154
4	7401	16071	139
5	7060	15754	168
6	7116	16033	152
7	7071	16119	152
8	7114	16060	177
9	7134	15649	154
10	7231	15655	147
Ortalama	**7196**	**15989**	**151**



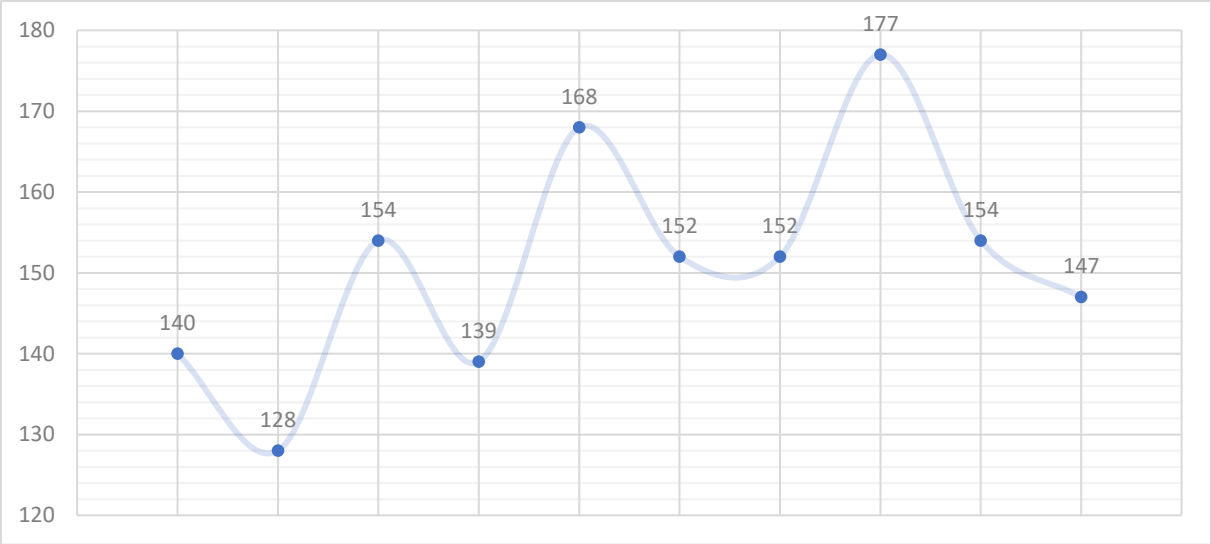
Şekil 4.40 LOCAL REST JSON Küçük Veri Node.js request

Test ortalaması 7196 ms olup, açıklığın ortalamaya sapması %5,1 ve çeyrekler açıklığının ortalamaya sapması %1,9 olduğu kaydedilmiştir.



Şekil 4.41 S4 REST JSON Küçük Veri Node.js request

Test ortalaması 15989 ms olup, açıklığın ortalamaya sapması %4,1 ve çeyrekler açıklığının ortalamaya sapması %1,8 olduğu kaydedilmiştir.



Şekil 4.42 IDES REST JSON Küçük Veri Node.js request

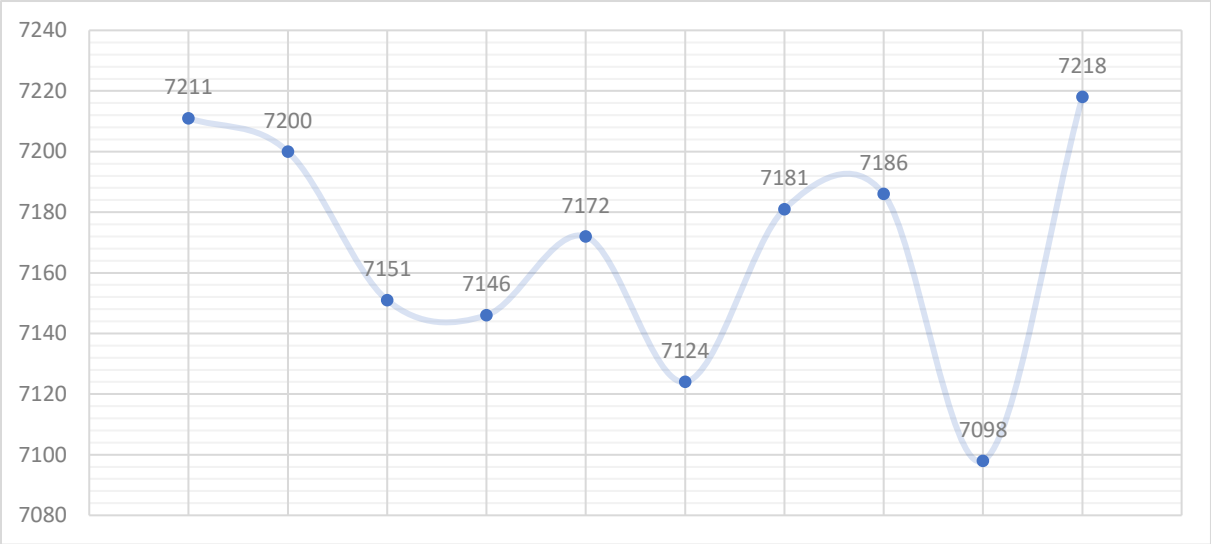
Test ortalaması 151 ms olup, açıklığın ortalamaya sapması %32,4 ve çeyrekler açıklığının ortalamaya sapması %8,1 olduğu kaydedilmiştir.

Node.js axios Çalışması

Bu çalışma JavaScript programlama dilinde Node.js v18 sürümünde axios kütüphanesi kullanılarak Visual Studio Code ortamında gerçeklenmiş ve Tablo 4.15'teki ve Şekil 4.43-Şekil 4.45 arasındaki sonuçlar elde edilmiştir. Program koduna EK 2.7'de yer verilmiştir.

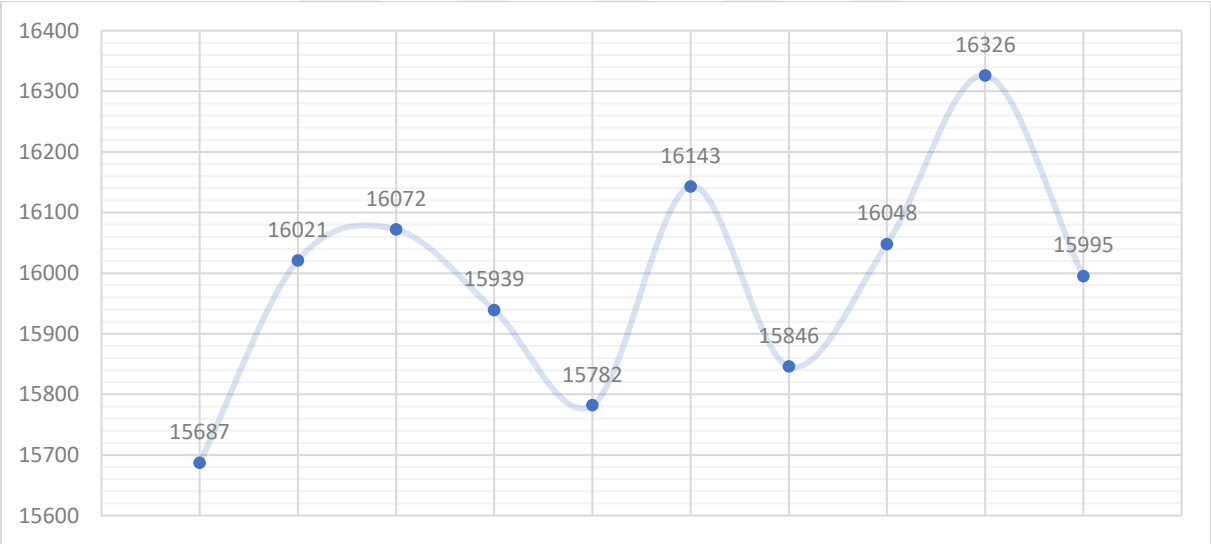
Tablo 4.15 REST JSON Küçük Veri Node.js axios

Tekrar	LOCAL	S4	IDES
1	7211	15687	168
2	7200	16021	154
3	7151	16072	166
4	7146	15939	165
5	7172	15782	143
6	7124	16143	150
7	7181	15846	143
8	7186	16048	147
9	7098	16326	175
10	7218	15995	206
Ortalama	**7169**	**15986**	**162**



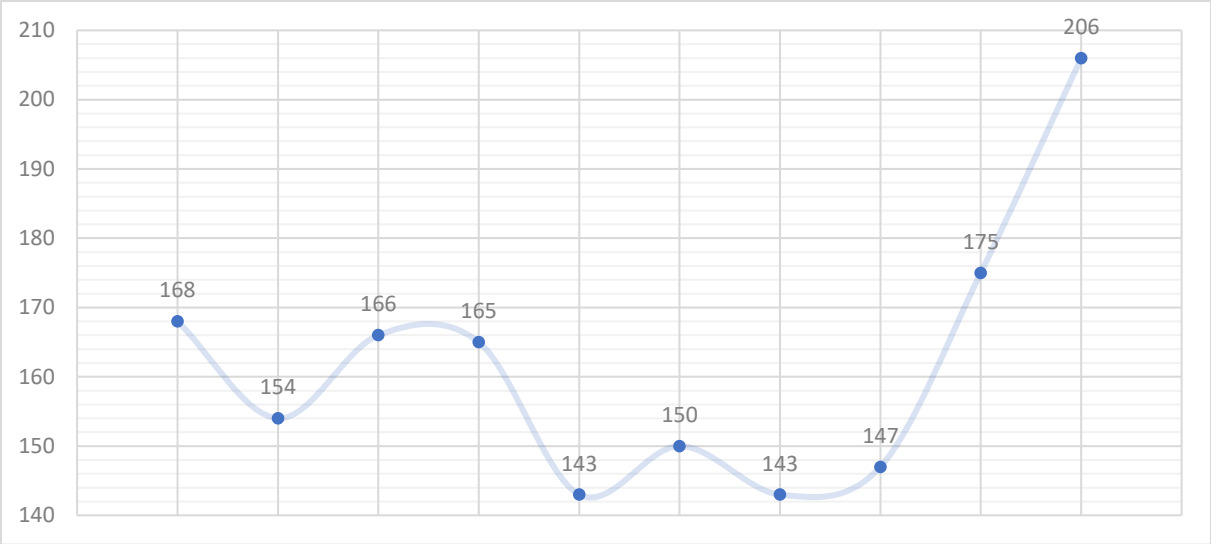
Şekil 4.43 LOCAL REST JSON Küçük Veri Node.js axios

Test ortalaması 7169 ms olup, açıklığın ortalamaya sapması %1,7 ve çeyrekler açıklığının ortalamaya sapması %0,7 olduğu kaydedilmiştir.



Şekil 4.44 S4 REST JSON Küçük Veri Node.js axios

Test ortalaması 15986 ms olup, açıklığın ortalamaya sapması %4 ve çeyrekler açıklığının ortalamaya sapması %1,2 olduğu kaydedilmiştir.



Şekil 4.45 IDES REST JSON Küçük Veri Node.js axios

Test ortalaması 162 ms olup, açıklığın ortalamaya sapması %39 ve çeyrekler açıklığının ortalamaya sapması %12,2 olduğu kaydedilmiştir.

4.1.3. Küçük Veriyle REST XML Uygulamaları

Bu tez çalışmasında üçüncü olarak XML formatındaki küçük verilerle REST API çalışılmıştır. Test çeşitliliği ve sonuçların doğru yorumlanabilmesi için üç farklı sunucuyla çalışılmış ve sunuculardaki veri büyüklükleri sırasıyla IDES ile LOCAL arasında yaklaşık 100 kat, LOCAL ile S4 arasında yaklaşık 5 kat büyüklük farkı olacak şeklindedir. Üç farklı programlama dili kullanılıp bunlar sırasıyla C#, Java ve JavaScript(Node.js) dilleri olmuştur. Bu programlama dillerinin de kendi içlerinde farklı kütüphaneler ve sınıflar kullanılarak test sonuçları her programlama dili içerisinde çeşitlendirilmiştir. Her bir program uygun olduğu geliştirme ortamında geliştirilip derlenmiş ve çalıştırılmıştır. Her bir çalıştırma döngüsü kendi içerisinde üç ana başlıktan oluşmaktadır; istek paketlerinin oluşturulması, gönderilmesi ve cevabın alınması, cevabın yorumlanması. Bu üç başlığın toplam çalışma süresi milisaniye(ms) biriminden kaydedilmiş ve tez çalışması kapsamında yorumlanmıştır. Bu tez çalışması kapsamında yorumlanan veriler; programların toplam çalışma süreleri değil, belirtilen bu üç ana başlıkta geçirilen sürelerdir.

SAP sunucusunun REST API ile XML formatında döndürdüğü cevaba EK 3.1'de yer verilmiştir.

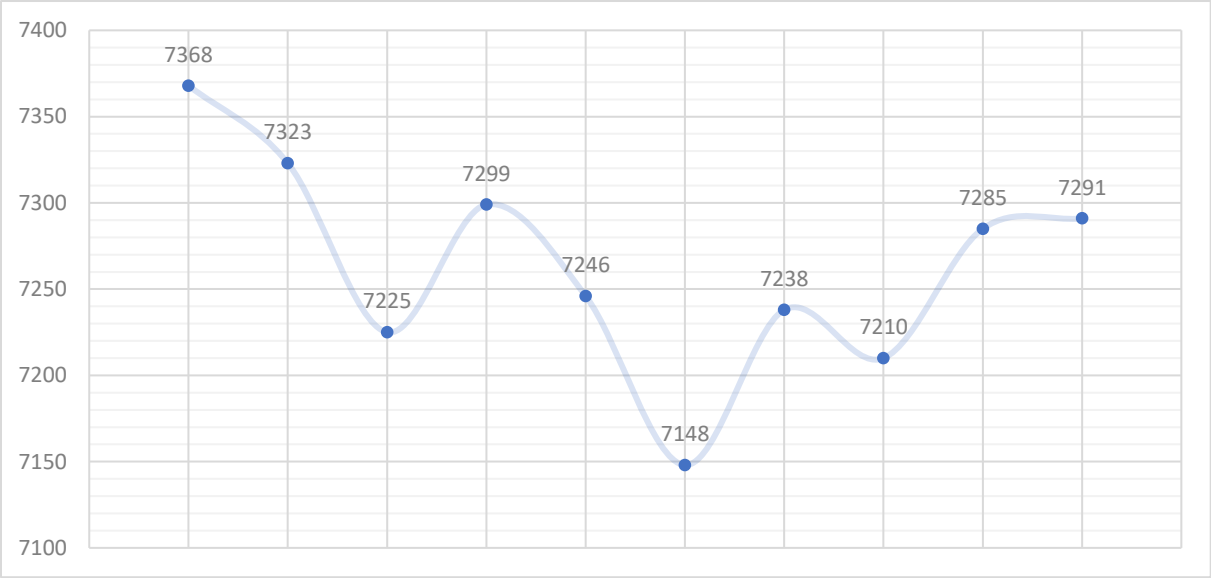
Kullanılan farklı diller ve bu dillerde kullanılan farklı kütüphaneler veya sınıflar REST isteklerini farklı yöntemlerle oluşturup, gönderip, EK 3.1'de yer verilen aynı cevabı almıştır. Bu cevap; SAP tarafında veritabanının farklı alanlarından SQL sorguları ve ABAP fonksiyonlarıyla çekilmiş, paketlenmiş ve gönderilmiştir. SAP ABAP program koduna EK 4.3'de yer verilmiştir.

C# HttpClient Çalışması

Bu çalışma C# programlama dilinde .NET 7.0 platformunda HttpClient sınıfı kullanılarak Visual Studio ortamında gerçeklenmiş ve Tablo 4.16'daki ve Şekil 4.46-Şekil 4.48 arasındaki sonuçlar elde edilmiştir. Program koduna EK 3.2'de yer verilmiştir.

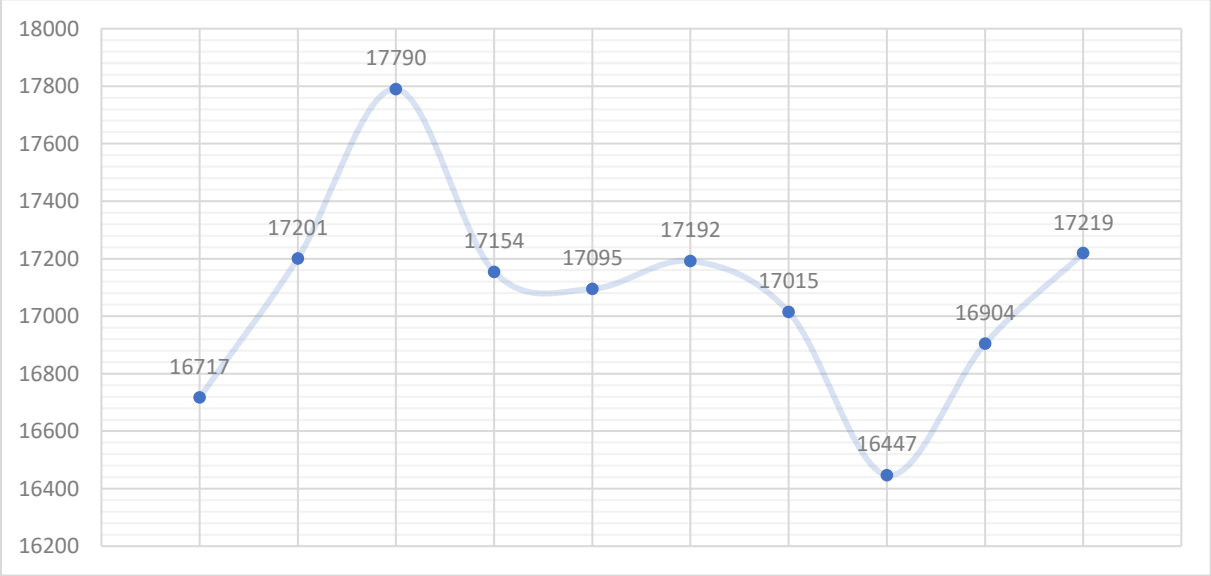
Tablo 4.16 REST XML Küçük Veri C# HttpClient

Tekrar	LOCAL	S4	IDES
1	7368	16717	187
2	7323	17201	192
3	7225	17790	214
4	7299	17154	195
5	7246	17095	208
6	7148	17192	217
7	7238	17015	205
8	7210	16447	191
9	7285	16904	213
10	7291	17219	247
Ortalama	**7263**	**17073**	**207**



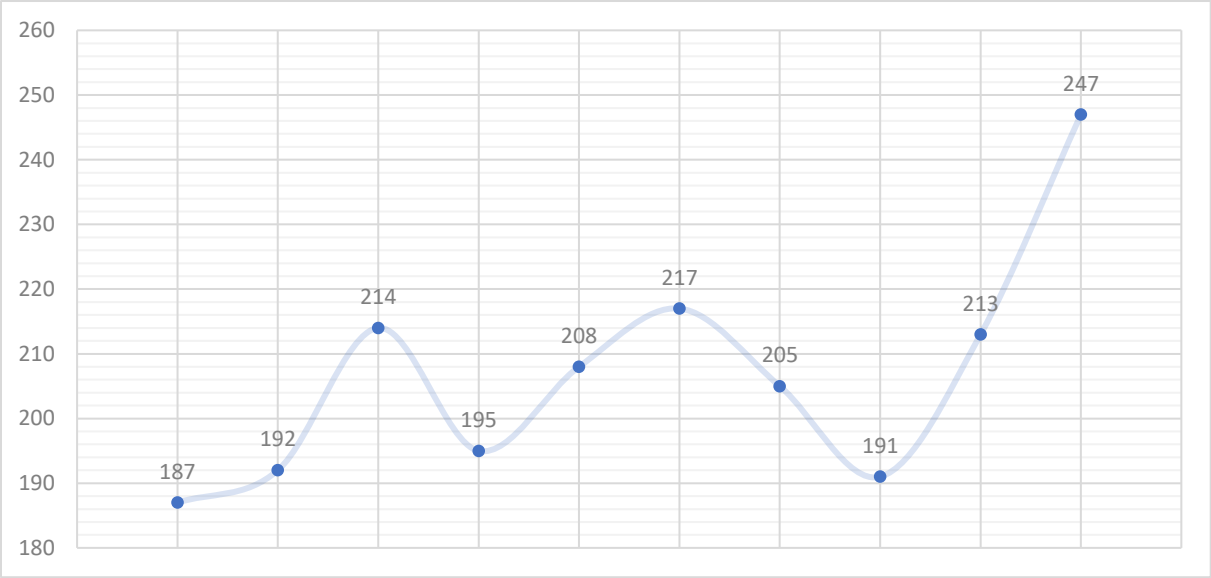
Şekil 4.46 LOCAL REST XML Küçük Veri C# HttpClient

Test ortalaması 7263 ms olup, açıklığın ortalamaya sapması %3 ve çeyrekler açıklığının ortalamaya sapması %0,9 olduğu kaydedilmiştir.



Şekil 4.47 S4 REST XML Küçük Veri C# HttpClient

Test ortalaması 17073 ms olup, açıklığın ortalamaya sapması %7,9 ve çeyrekler açıklığının ortalamaya sapması %1,6 olduğu kaydedilmiştir.



Şekil 4.48 IDES REST XML Küçük Veri C# HttpClient

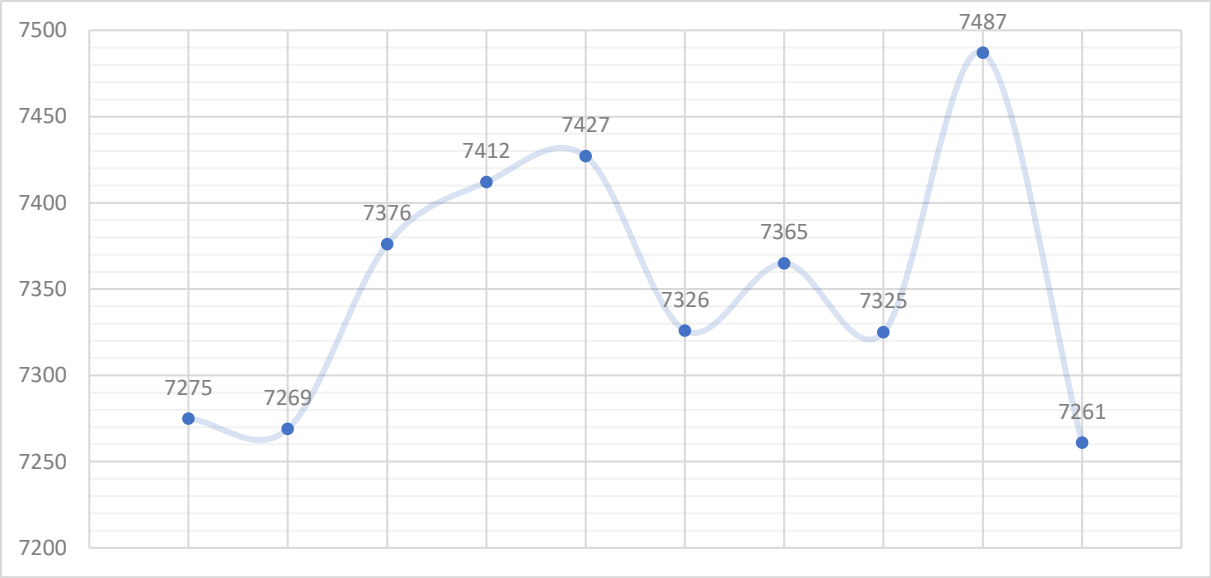
Test ortalaması 207 ms olup, açıklığın ortalamaya sapması %29 ve çeyrekler açıklığının ortalamaya sapması %10,1 olduğu kaydedilmiştir.

C# RestSharp Çalışması

Bu çalışma C# programlama dilinde .NET 7.0 platformunda RestSharp sınıfı kullanılarak Visual Studio ortamında gerçeklenmiş ve Tablo 4.17'deki ve Şekil 4.49-Şekil 4.51 arasındaki sonuçlar elde edilmiştir. Program koduna EK 3.3'te yer verilmiştir.

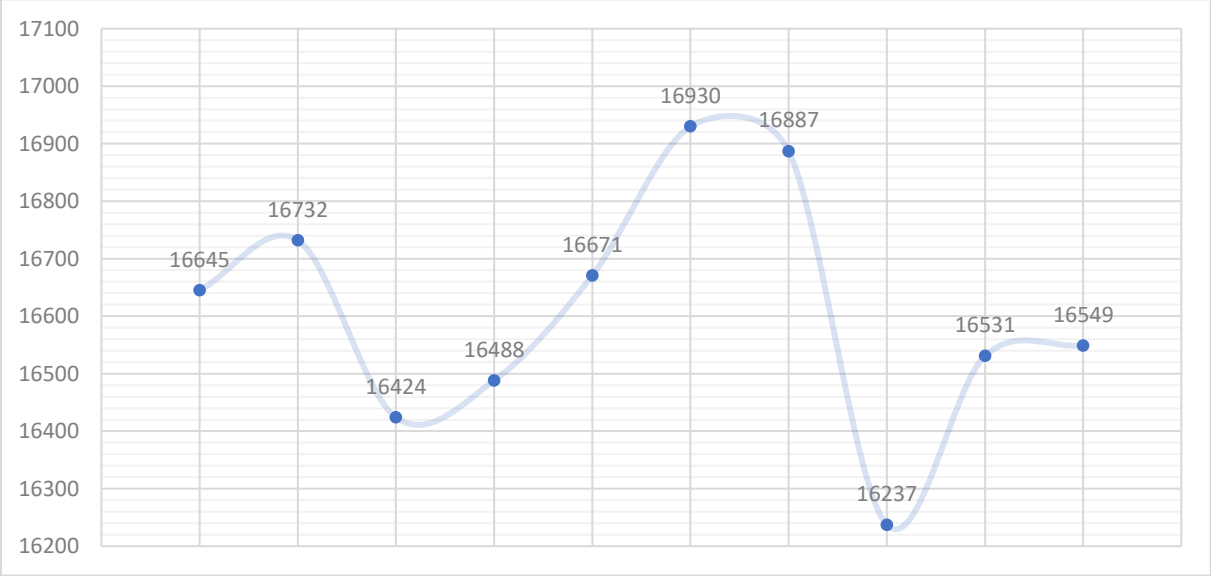
Tablo 4.17 REST XML Küçük Veri C# RestSharp

Tekrar	LOCAL	S4	IDES
1	7275	16645	247
2	7269	16732	264
3	7376	16424	260
4	7412	16488	282
5	7427	16671	245
6	7326	16930	258
7	7365	16887	279
8	7325	16237	248
9	7487	16531	246
10	7261	16549	318
Ortalama	**7352**	**16609**	**265**



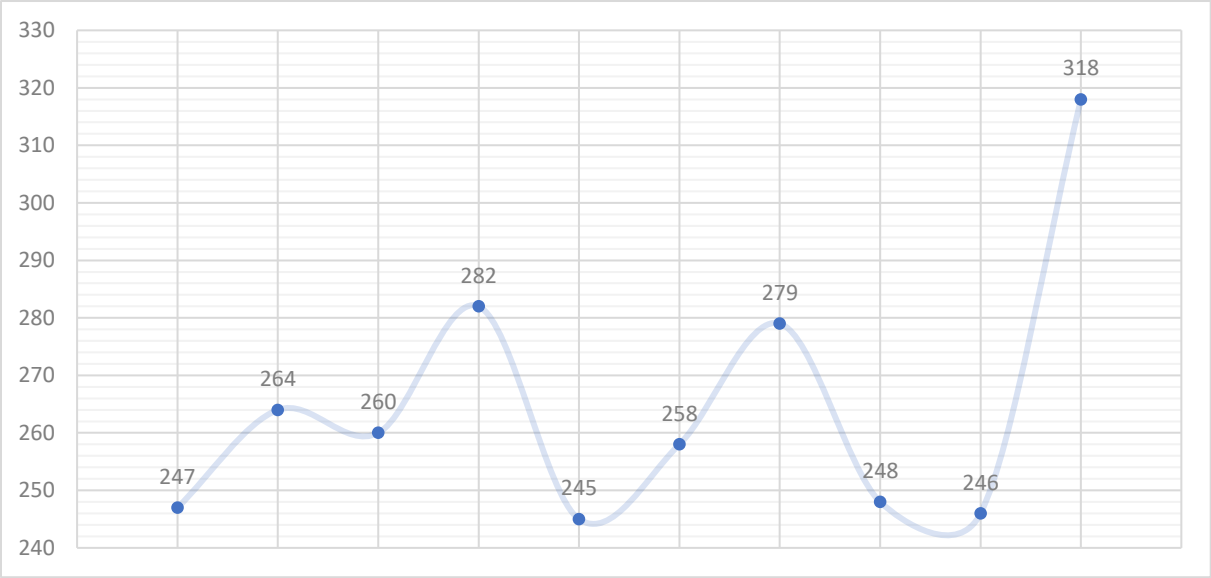
Şekil 4.49 LOCAL REST XML Küçük Veri C# RestSharp

Test ortalaması 7352 ms olup, açıklığın ortalamaya sapması %3,1 ve çeyrekler açıklığının ortalamaya sapması %1,6 olduğu kaydedilmiştir.



Şekil 4.60 S4 REST XML Küçük Veri C# RestSharp

Test ortalaması 16609 ms olup, açıklığın ortalamaya sapması %4,2 ve çeyrekler açıklığının ortalamaya sapması %1,3 olduğu kaydedilmiştir.



Şekil 4.71 IDES REST XML Küçük Veri C# RestSharp

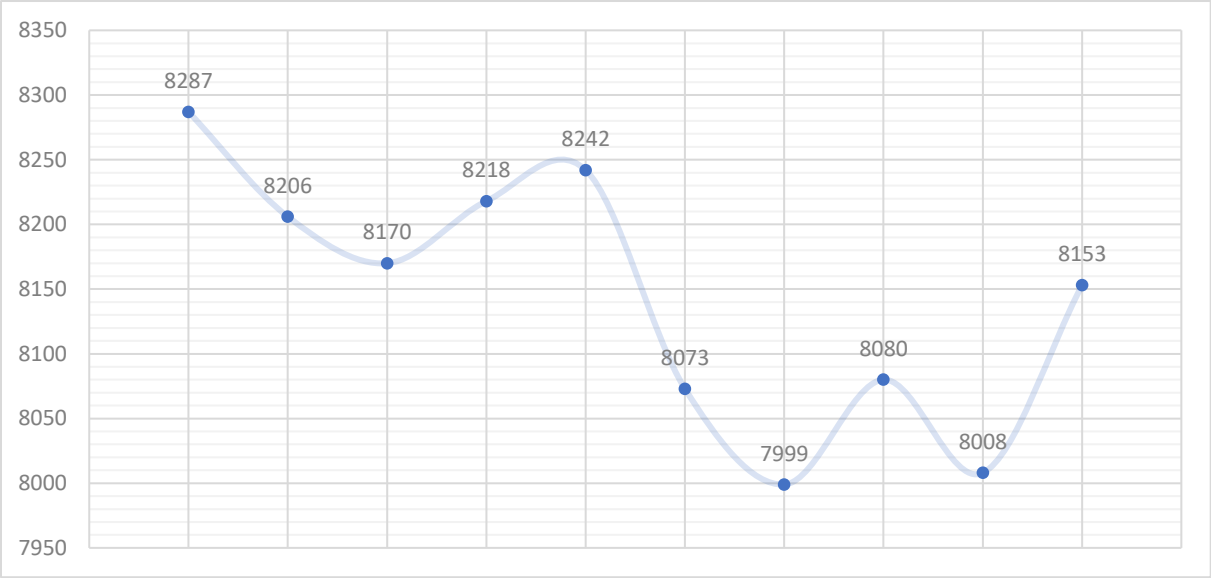
Test ortalaması 265 ms olup, açıklığın ortalamaya sapması %27,6 ve çeyrekler açıklığının ortalamaya sapması %10,6 olduğu kaydedilmiştir.

Java HttpURLConnection Çalışması

Bu çalışma Java programlama dilinde JDK 19 sürümünde HttpURLConnection sınıfı kullanılarak IntelliJ IDEA ortamında gerçeklenmiş ve Tablo 4.18'deki ve Şekil 4.52-Şekil 4.54 arasındaki sonuçlar elde edilmiştir. Program koduna EK 3.4'te yer verilmiştir.

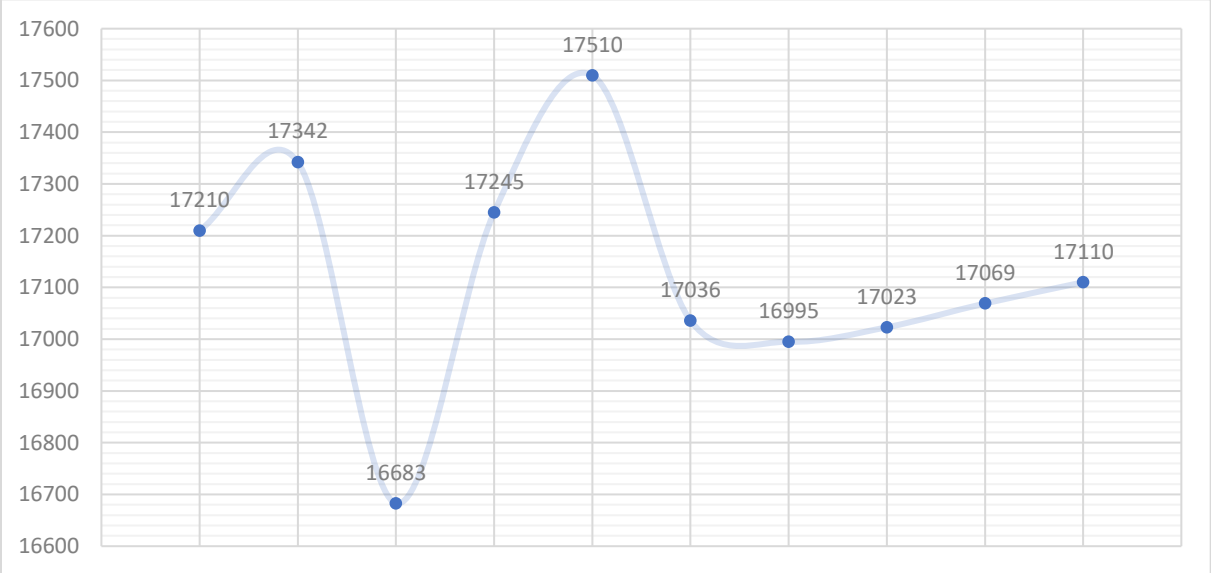
Tablo 4.18 REST XML Küçük Veri Java HttpURLConnection

Tekrar	LOCAL	S4	IDES
1	8287	17210	179
2	8206	17342	168
3	8170	16683	141
4	8218	17245	166
5	8242	17510	170
6	8073	17036	156
7	7999	16995	153
8	8080	17023	169
9	8008	17069	144
10	8153	17110	155
Ortalama	**8144**	**17122**	**160**



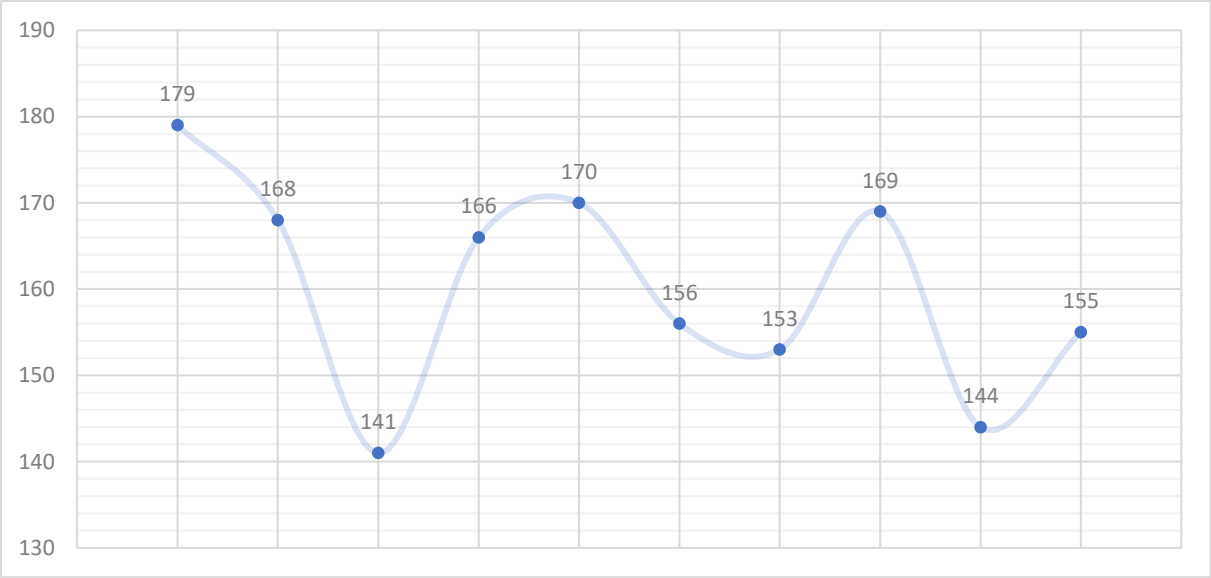
Şekil 4.82 LOCAL REST XML Küçük Veri Java HttpURLConnection

Test ortalaması 8144 ms olup, açıklığın ortalamaya sapması %3,5 ve çeyrekler açıklığının ortalamaya sapması %1,7 olduğu kaydedilmiştir.



Şekil 4.93 S4 REST XML Küçük Veri Java HttpURLConnection

Test ortalaması 17122 ms olup, açıklığın ortalamaya sapması %4,8 ve çeyrekler açıklığının ortalamaya sapması %1,2 olduğu kaydedilmiştir.



Şekil 4.104 IDES REST XML Küçük Veri Java HttpURLConnection

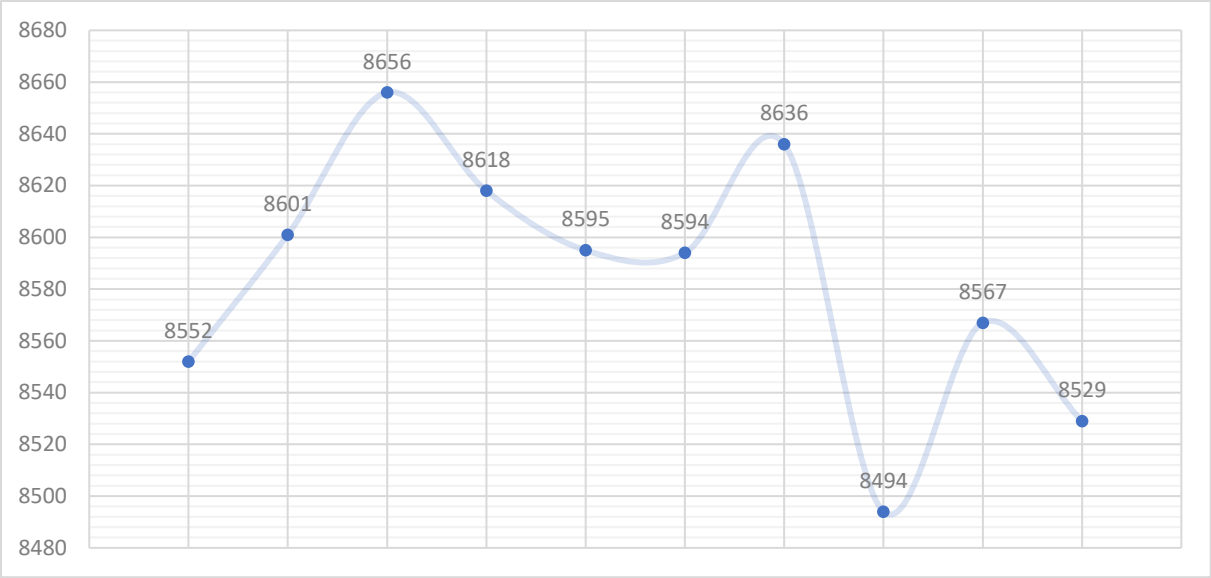
Test ortalaması 160 ms olup, açıklığın ortalamaya sapması %23,7 ve çeyrekler açıklığının ortalamaya sapması %9,5 olduğu kaydedilmiştir.

Java HttpClient Çalışması

Bu çalışma Java programlama dilinde JDK 19 sürümünde HttpClient sınıfı kullanılarak IntelliJ IDEA ortamında gerçeklenmiş ve Tablo 4.19'daki ve Şekil 4.55-Şekil 4.57 arasındaki sonuçlar elde edilmiştir. Program koduna EK 3.5'te yer verilmiştir.

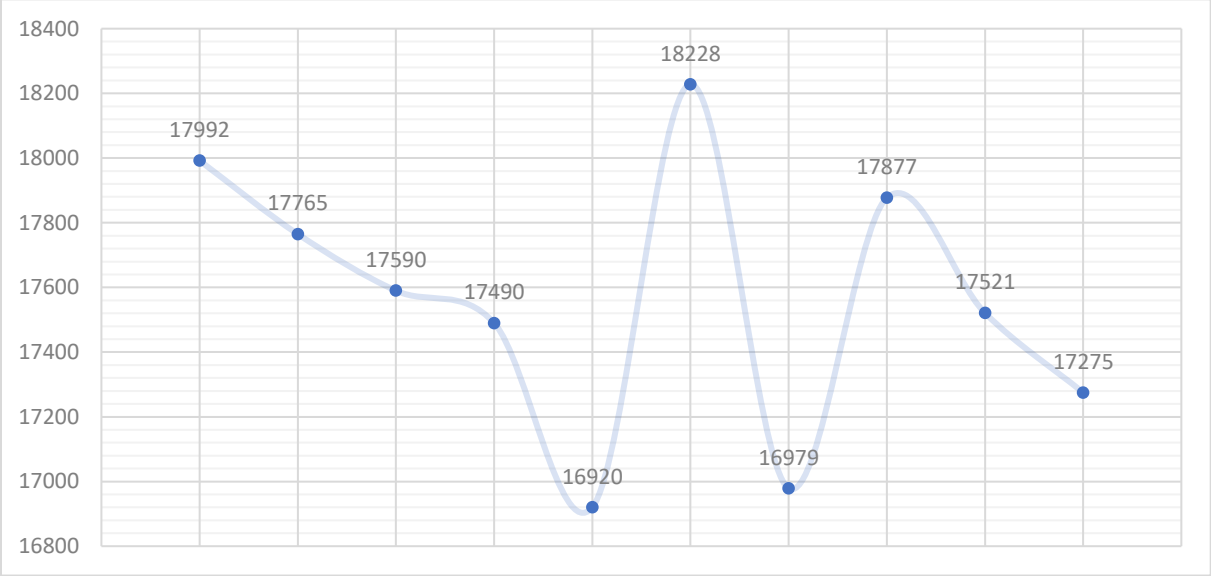
Tablo 4.19 REST XML Küçük Veri Java HttpClient

Tekrar	LOCAL	S4	IDES
1	8552	17992	508
2	8601	17765	472
3	8656	17590	426
4	8618	17490	431
5	8595	16920	437
6	8594	18228	434
7	8636	16979	456
8	8494	17877	459
9	8567	17521	462
10	8529	17275	437
Ortalama	**8584**	**17564**	**452**



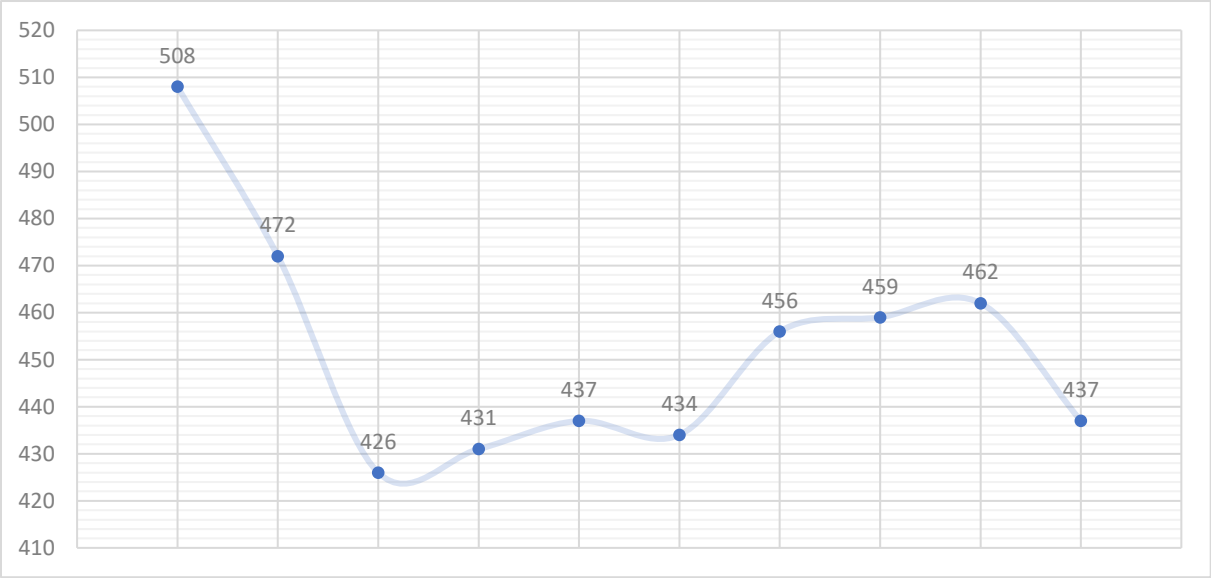
Şekil 4.55 LOCAL REST XML Küçük Veri Java HttpClient

Test ortalaması 8584 ms olup, açıklığın ortalamaya sapması %1,9 ve çeyrekler açıklığının ortalamaya sapması %0,7 olduğu kaydedilmiştir.



Şekil 4.116 S4 REST XML Küçük Veri Java HttpClient

Test ortalaması 17564 ms olup, açıklığın ortalamaya sapması %7,4 ve çeyrekler açıklığının ortalamaya sapması %3 olduğu kaydedilmiştir.



Şekil 4.127 IDES REST XML Küçük Veri Java HttpClient

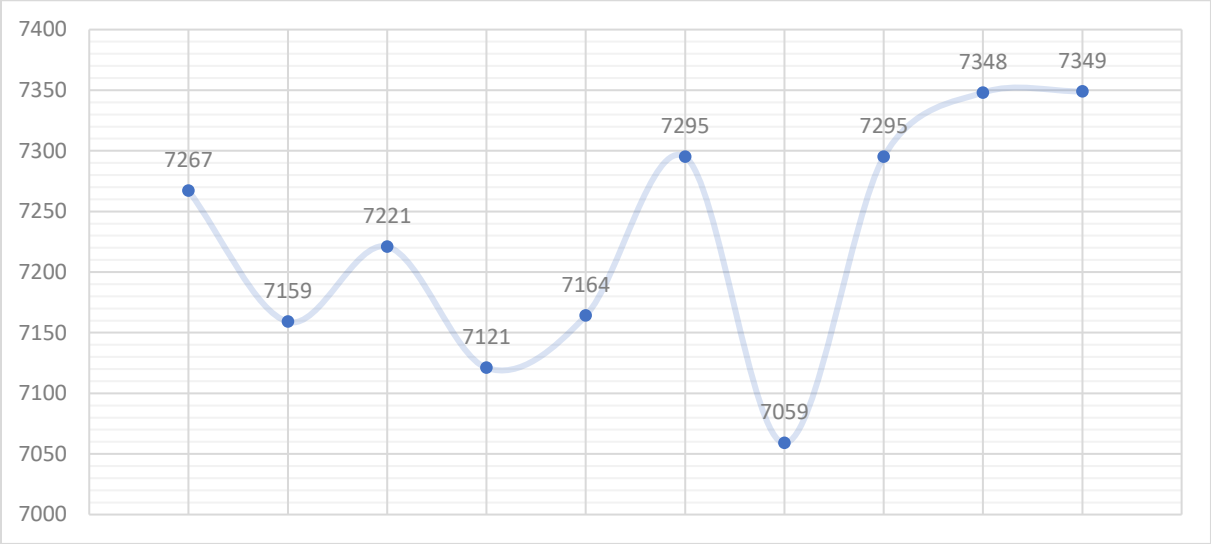
Test ortalaması 452 ms olup, açıklığın ortalamaya sapması %18,1 ve çeyrekler açıklığının ortalamaya sapması %5,9 olduğu kaydedilmiştir.

Node.js request Çalışması

Bu çalışma JavaScript programlama dilinde Node.js v18 sürümünde request kütüphanesi kullanılarak Visual Studio Code ortamında gerçeklenmiş ve Tablo 4.20'deki ve Şekil 4.58-Şekil 4.60 arasındaki sonuçlar elde edilmiştir. Program koduna EK 3.6'da yer verilmiştir.

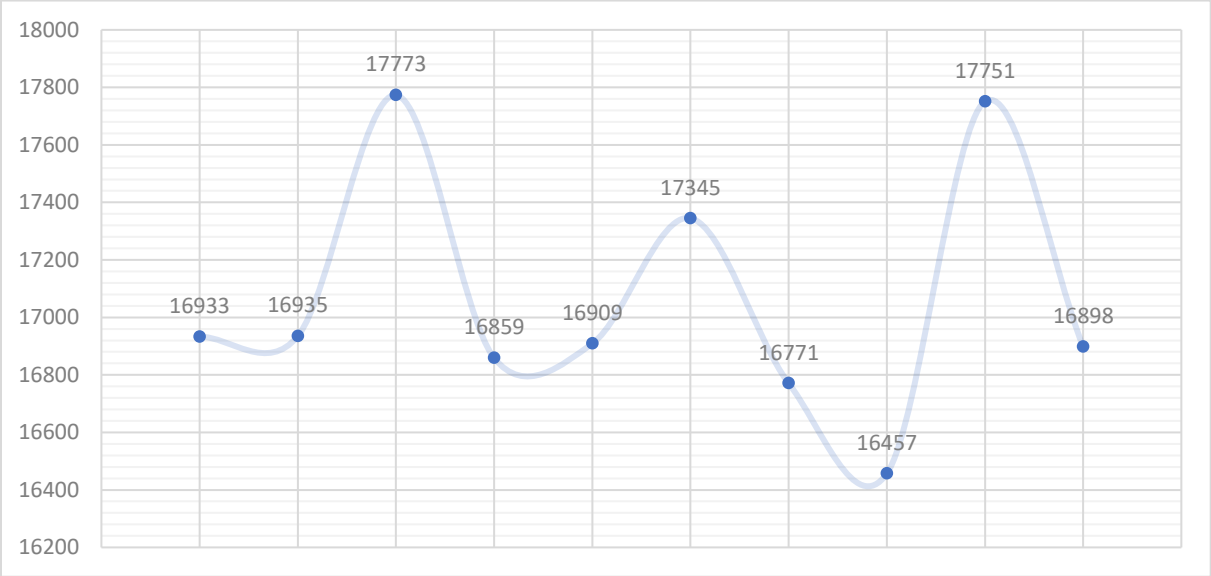
Tablo 4.20 REST XML Küçük Veri Node.js request

Tekrar	LOCAL	S4	IDES
1	7267	16933	159
2	7159	16935	128
3	7221	17773	142
4	7121	16859	216
5	7164	16909	205
6	7295	17345	167
7	7059	16771	182
8	7295	16457	204
9	7348	17751	156
10	7349	16898	179
Ortalama	**7228**	**17063**	**174**



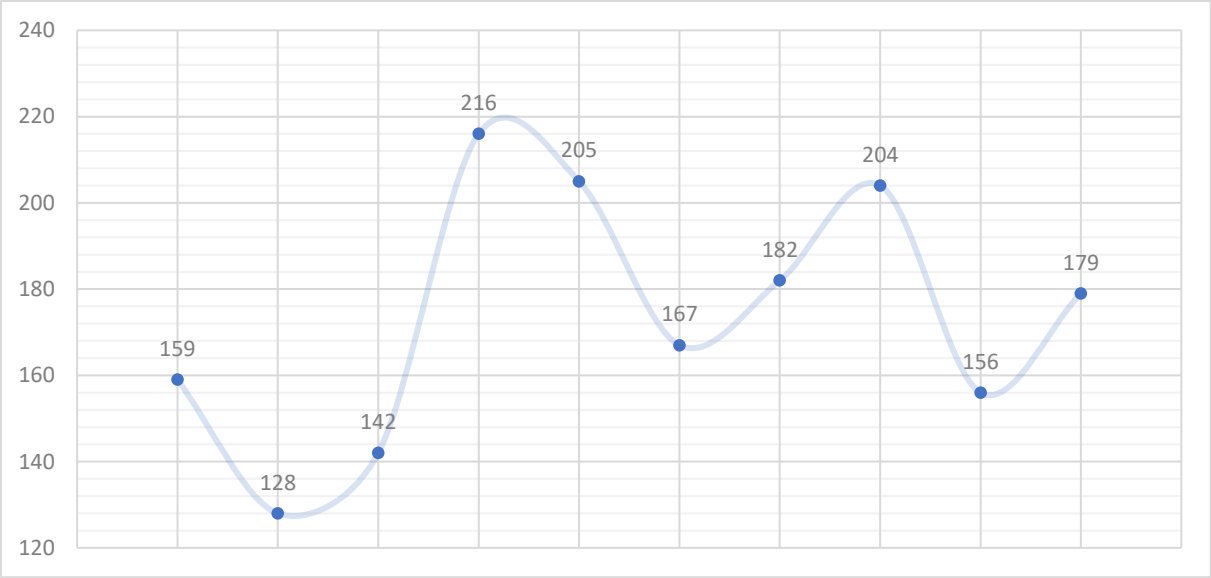
Şekil 4.58 LOCAL REST XML Küçük Veri Node.js request

Test ortalaması 7228 ms olup, açıklığın ortalamaya sapması %4 ve çeyrekler açıklığının ortalamaya sapması %1,9 olduğu kaydedilmiştir.



Şekil 4.139 S4 REST XML Küçük Veri Node.js request

Test ortalaması 17063 ms olup, açıklığın ortalamaya sapması %7,7 ve çeyrekler açıklığının ortalamaya sapması %2,2 olduğu kaydedilmiştir.



Şekil 4.60 IDES REST XML Küçük Veri Node.js request

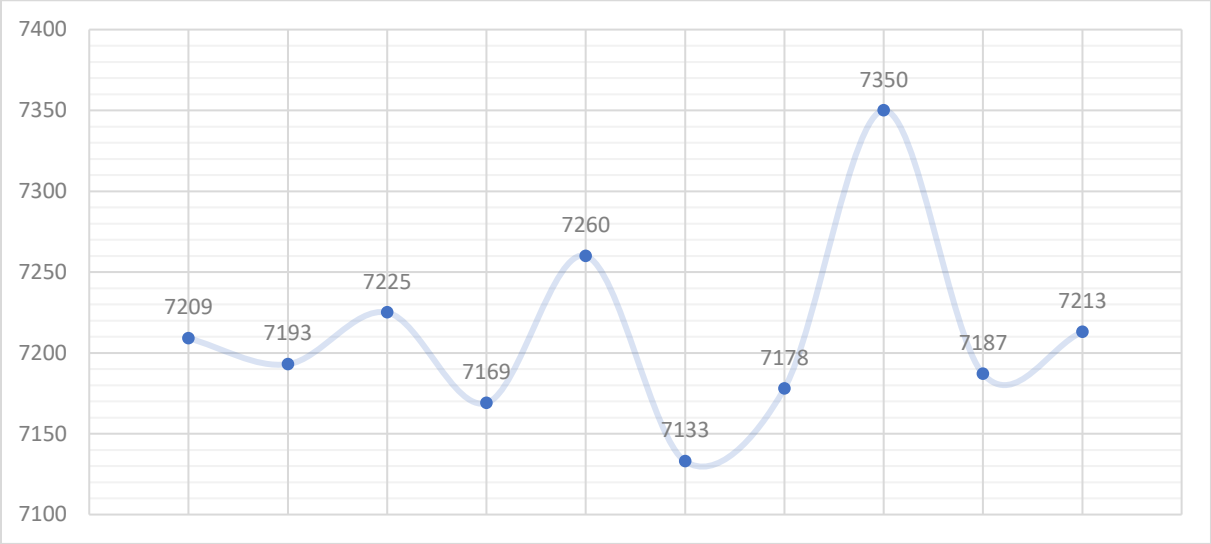
Test ortalaması 174 ms olup, açıklığın ortalamaya sapması %50,6 ve çeyrekler açıklığının ortalamaya sapması %24 olduğu kaydedilmiştir.

Node.js axios Çalışması

Bu çalışma JavaScript programlama dilinde Node.js v18 sürümünde axios kütüphanesi kullanılarak Visual Studio Code ortamında gerçeklenmiş ve Tablo 4.21'deki ve Şekil 4.61-Şekil 4.63 arasındaki sonuçlar elde edilmiştir. Program koduna EK 3.7'de yer verilmiştir.

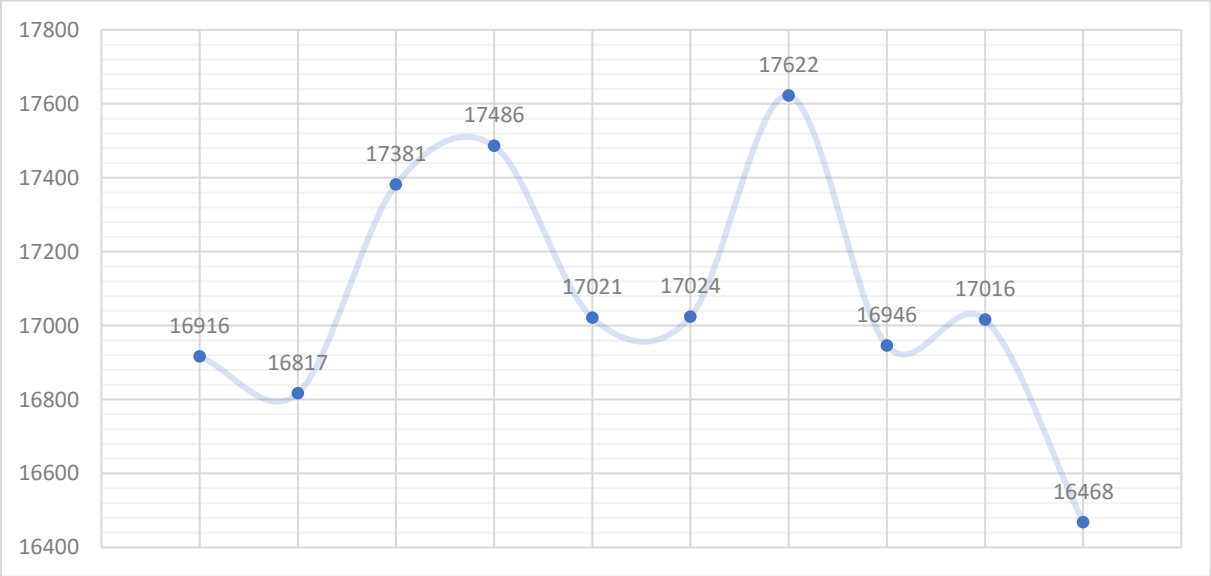
Tablo 4.21 REST XML Küçük Veri Node.js axios

Tekrar	LOCAL	S4	IDES
1	7209	16916	162
2	7193	16817	149
3	7225	17381	144
4	7169	17486	146
5	7260	17021	158
6	7133	17024	176
7	7178	17622	158
8	7350	16946	146
9	7187	17016	154
10	7213	16468	184
Ortalama	**7212**	**17070**	**158**



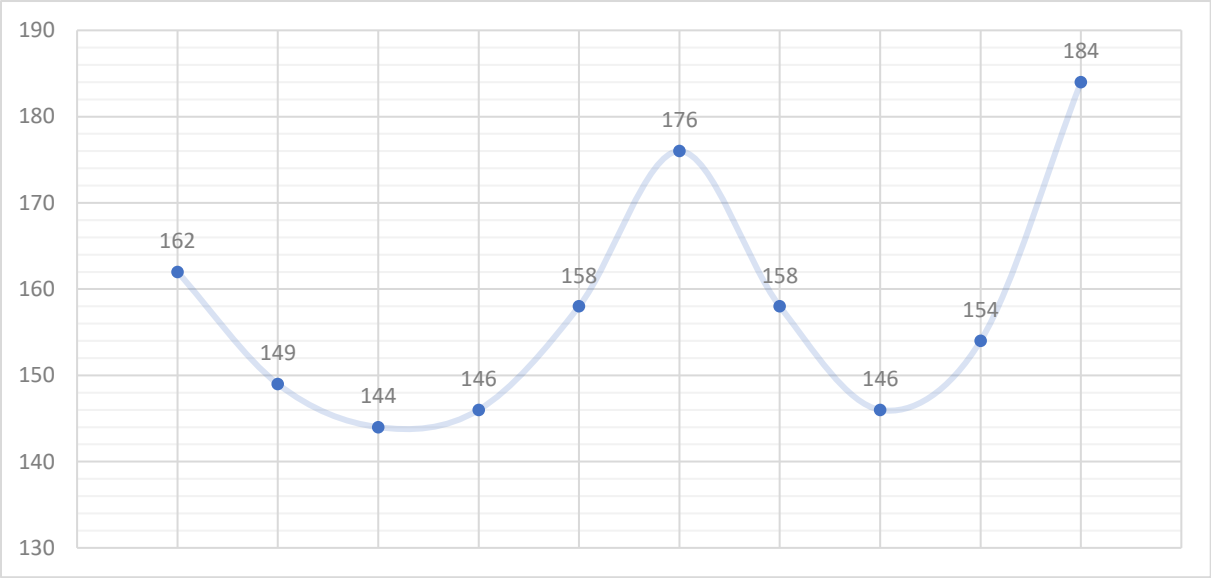
Şekil 4.61 LOCAL REST XML Küçük Veri Node.js axios

Test ortalaması 7212 ms olup, açıklığın ortalamaya sapması %3 ve çeyrekler açıklığının ortalamaya sapması %0,6 olduğu kaydedilmiştir.



Şekil 4.62 S4 REST XML Küçük Veri Node.js axios

Test ortalaması 17070 ms olup, açıklığın ortalamaya sapması %6,8 ve çeyrekler açıklığının ortalamaya sapması %2,2 olduğu kaydedilmiştir.



Şekil 4.63 IDES REST XML Küçük Veri Node.js axios

Test ortalaması 158 ms olup, açıklığın ortalamaya sapması %25,4 ve çeyrekler açıklığının ortalamaya sapması %9 olduğu kaydedilmiştir.

4.1.4. Büyük Veriyle SOAP Uygulamaları

Bu tez çalışmasının yeni bölümünde büyük verilerle SOAP API çalışılmıştır. Küçük verilerle çalışmalara göre 100-10.000 kat daha büyük ve her sunucuda aynı verilerle çalışılmış böylelikle veri boyutlarının SAP tarafında, API üzerinde, dosya formatı boyutlarında ve istemci tarafında yorumlanmasındaki performans farkları daha net ortaya koyulmuştur. Üç farklı programlama dili kullanılıp bunlar sırasıyla C#, Java ve JavaScript(Node.js) dilleri olmuştur. Bu programlama dillerinin de kendi içlerinde farklı kütüphaneler ve sınıflar kullanılarak test sonuçları her programlama dili içerisinde çeşitlendirilmiştir. Her bir program uygun olduğu geliştirme ortamında geliştirilip derlenmiş ve çalıştırılmıştır. Her bir çalıştırma döngüsü kendi içerisinde üç ana başlıktan oluşmaktadır; istek paketlerinin oluşturulması, gönderilmesi ve cevabın alınması, cevabın yorumlanması. Bu üç başlığın toplam çalışma süresi milisaniye(ms) biriminden kaydedilmiş ve tez çalışması kapsamında yorumlanmıştır. Bu tez çalışması kapsamında yorumlanan veriler; programların toplam çalışma süreleri değil, belirtilen bu üç ana başlıkta geçirilen sürelerdir.

İstek atılan SAP sunucusunun SOAP WSDL tanımlamasına EK 1.1'de yer verilmiştir.

SAP sunucumuza gönderilen SOAP istek mektubuna EK 1.2'de yer verilmiştir.

SAP sunucumuzun bu isteğe cevabına EK 1.3'te yer verilmiştir.

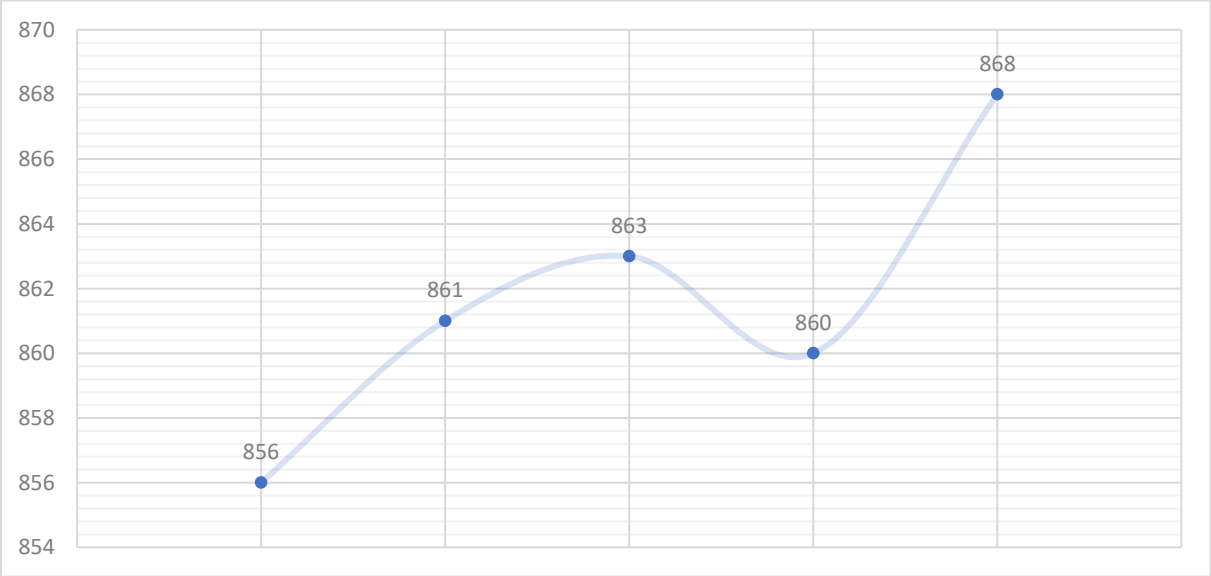
Kullanılan farklı diller ve bu dillerde kullanılan farklı kütüphaneler veya sınıflar SOAP istek mektubunu ve HTTP 1.1 isteğini farklı yöntemlerle oluşturup, gönderip, EK 1.3'te yer verilen aynı cevabı almıştır. Bu cevap; SAP tarafında veritabanının farklı alanlarından SQL sorguları ve ABAP fonksiyonlarıyla çekilmiş, paketlenmiş ve gönderilmiştir.

C# HttpClient Çalışması

Bu çalışma C# programlama dilinde .NET 7.0 platformunda HttpClient sınıfı kullanılarak Visual Studio ortamında gerçeklenmiş ve Tablo 4.22'deki ve Şekil 4.64-Şekil 4.66 arasındaki sonuçlar elde edilmiştir. Program koduna EK 1.4'te yer verilmiştir.

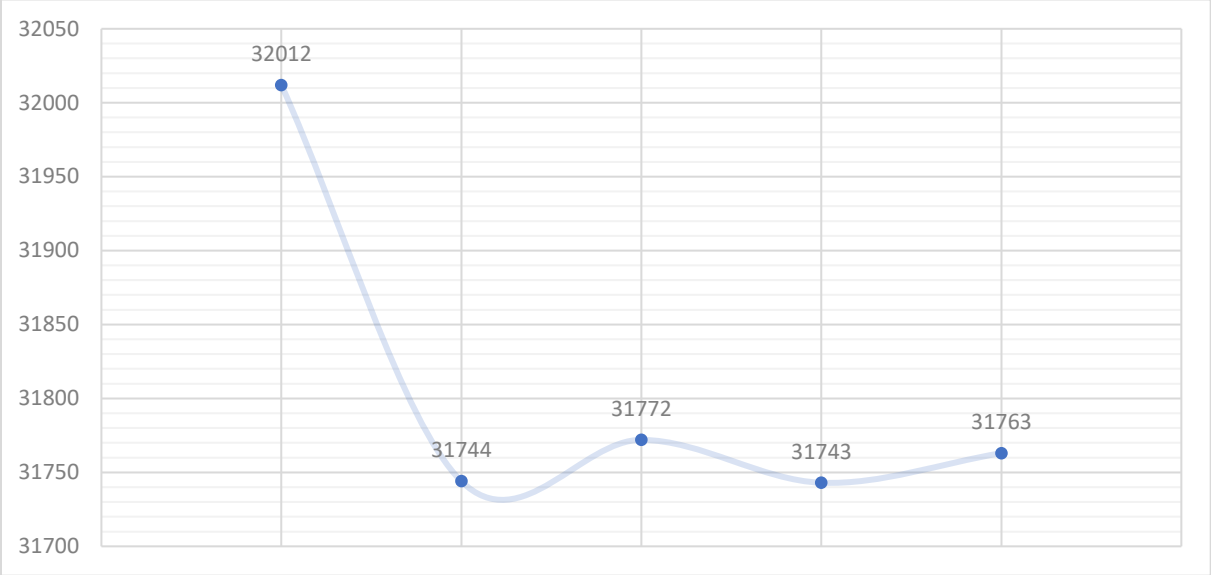
Tablo 4.22 SOAP Büyük Veri C# HttpClient

Tekrar	LOCAL	S4	IDES
1	856	32012	34269
2	861	31744	33310
3	863	31772	33530
4	860	31743	33467
5	868	31763	33260
Ortalama	**862**	**31807**	**33567**



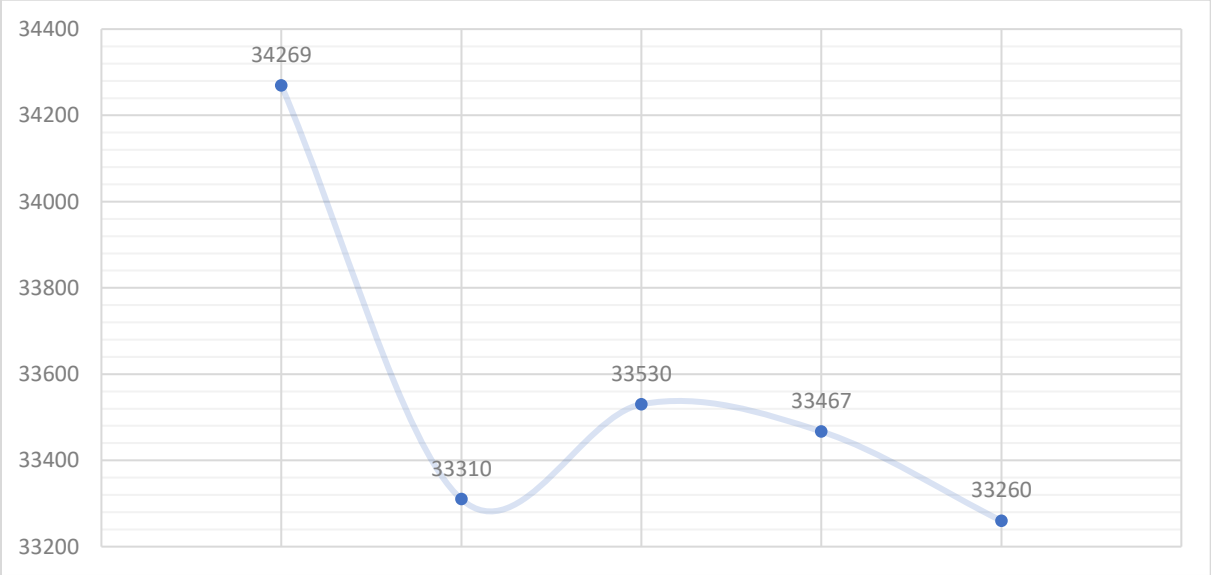
Şekil 4.64 LOCAL SOAP Büyük Veri C# HttpClient

Test ortalaması 862 ms olup, açıklığın ortalamaya sapması %1,4 ve çeyrekler açıklığının ortalamaya sapması %0,3 olduğu kaydedilmiştir.



Şekil 4.65 S4 SOAP Büyük Veri C# HttpClient

Test ortalaması 31807 ms olup, açıklığın ortalamaya sapması %0,8 ve çeyrekler açıklığının ortalamaya sapması %0,1 olduğu kaydedilmiştir.



Şekil 4.66 IDES SOAP Büyük Veri C# HttpClient

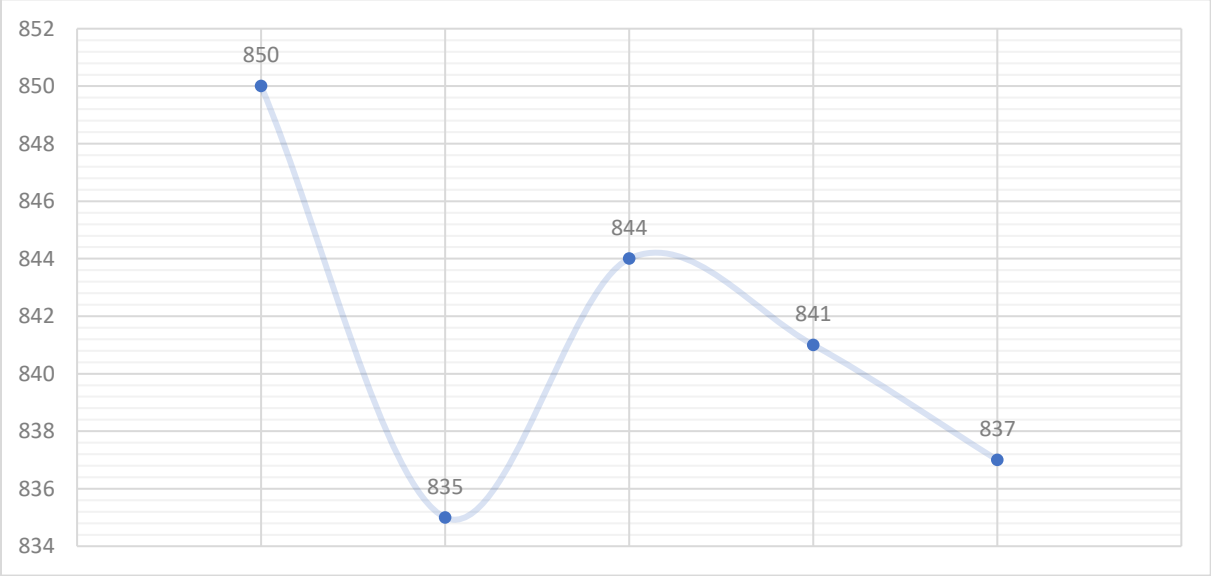
Test ortalaması 33567 ms olup, açıklığın ortalamaya sapması %3 ve çeyrekler açıklığının ortalamaya sapması %0,7 olduğu kaydedilmiştir.

C# WebClient Çalışması

Bu çalışma C# programlama dilinde .NET 7.0 platformunda WebClient sınıfı kullanılarak Visual Studio ortamında gerçeklenmiş ve Tablo 4.23'teki ve Şekil 4.67-Şekil 4.69 arasındaki sonuçlar elde edilmiştir. Program koduna EK 1.5'te yer verilmiştir.

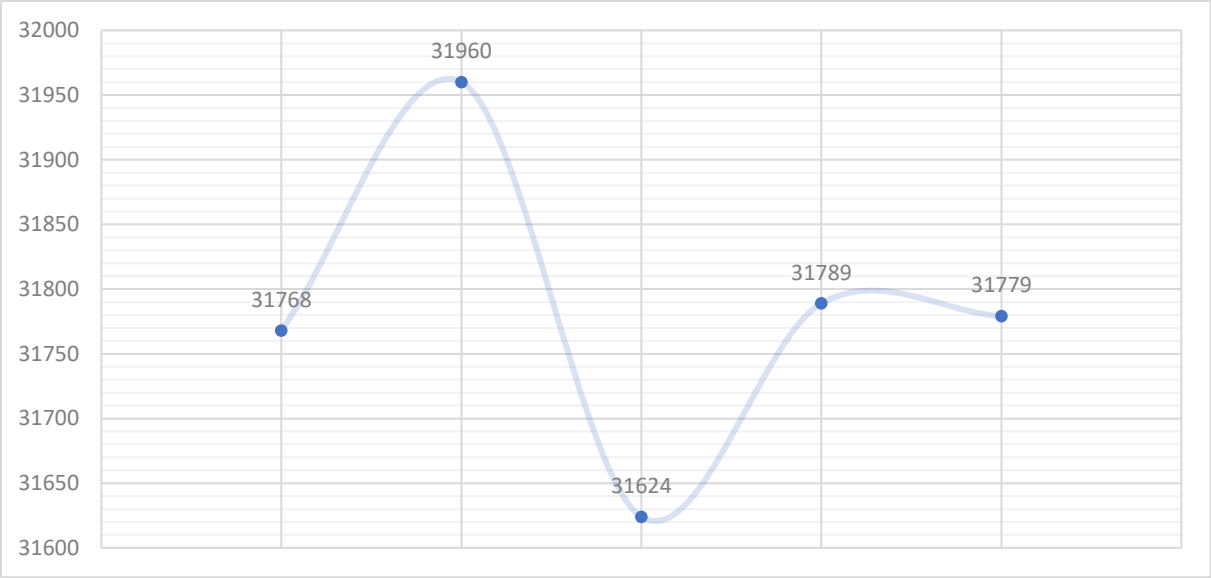
Tablo 4.223 SOAP Büyük Veri C# WebClient

Tekrar	LOCAL	S4	IDES
1	850	31768	33611
2	835	31960	33279
3	844	31624	33401
4	841	31789	33612
5	837	31779	33702
Ortalama	**841**	**31784**	**33521**



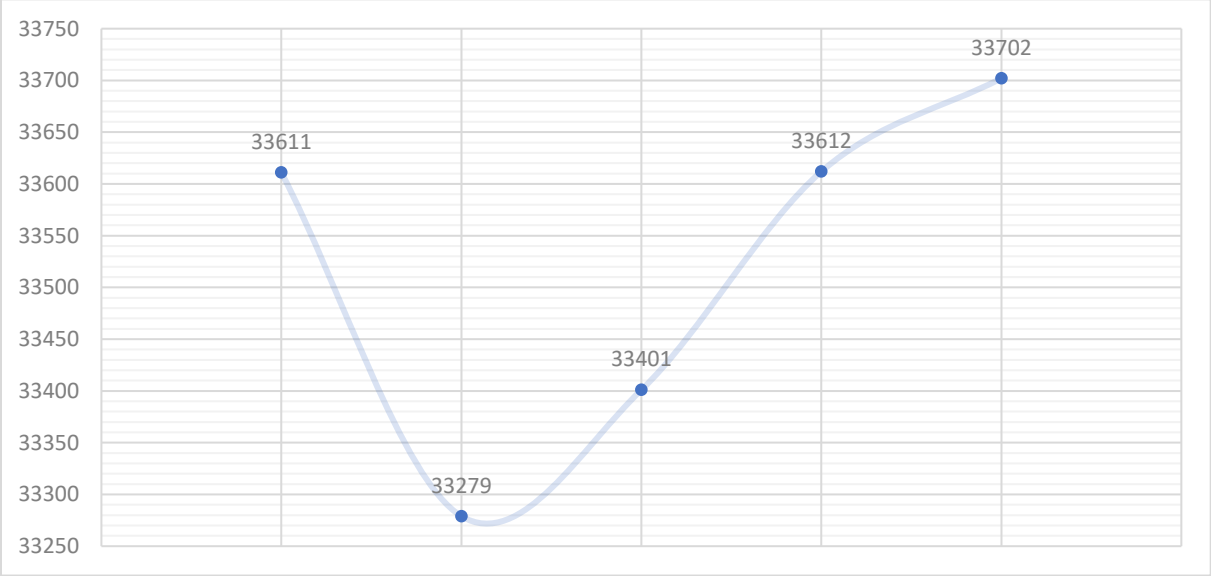
Şekil 4.67 LOCAL SOAP Büyük Veri C# WebClient

Test ortalaması 841 ms olup, açıklığın ortalamaya sapması %1,8 ve çeyrekler açıklığının ortalamaya sapması %0,8 olduğu kaydedilmiştir.



Şekil 4.68 S4 SOAP Büyük Veri C# WebClient

Test ortalaması 31784 ms olup, açıklığın ortalamaya sapması %1,1 ve çeyrekler açıklığının ortalamaya sapması %0,8 olduğu kaydedilmiştir.



Şekil 4.69 IDES SOAP Büyük Veri C# WebClient

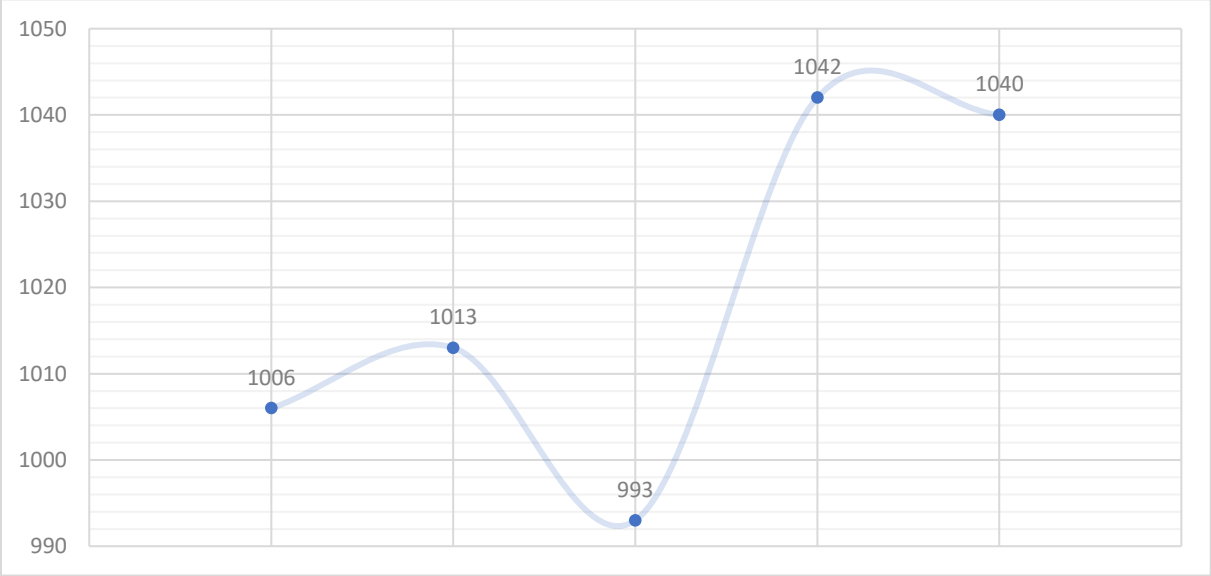
Test ortalaması 33521 ms olup, açıklığın ortalamaya sapması %1,3 ve çeyrekler açıklığının ortalamaya sapması %0,6 olduğu kaydedilmiştir.

Java HttpURLConnection Çalışması

Bu çalışma Java programlama dilinde JDK 19 sürümünde HttpURLConnection sınıfı kullanılarak IntelliJ IDEA ortamında gerçeklenmiş ve Tablo 4.24'teki ve Şekil 4.70-Şekil 4.72 arasındaki sonuçlar elde edilmiştir. Program koduna EK 1.6'da yer verilmiştir.

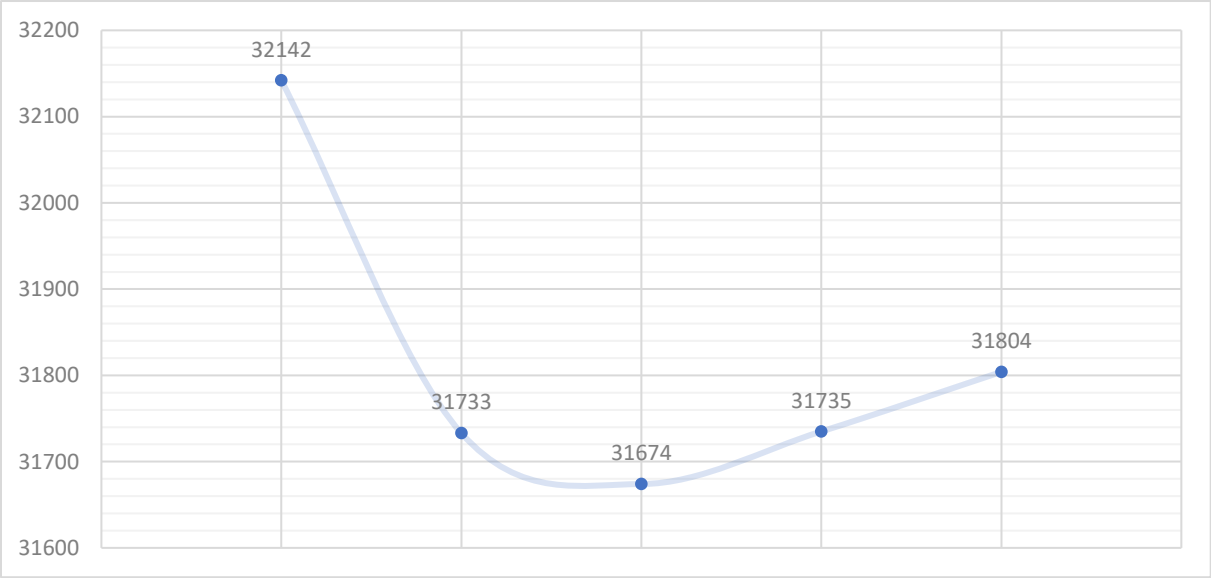
Tablo 4.24 SOAP Büyük Veri Java HttpURLConnection

Tekrar	LOCAL	S4	IDES
1	1006	32142	33471
2	1013	31733	33277
3	993	31674	33373
4	1042	31735	33408
5	1040	31804	33276
Ortalama	**1019**	**31818**	**33361**



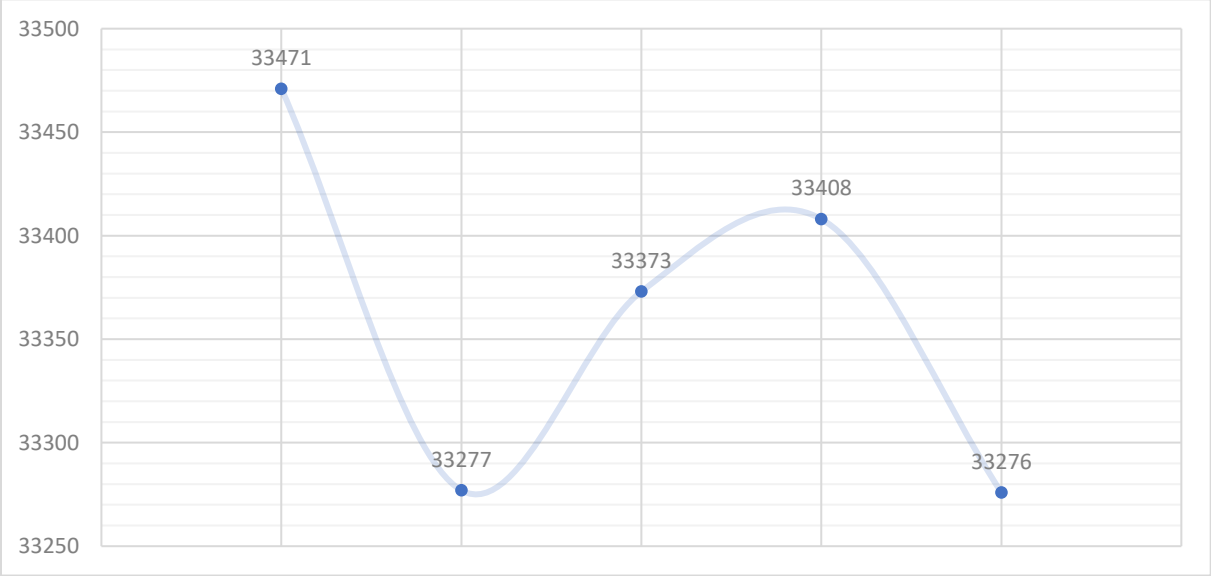
Şekil 4.70 LOCAL SOAP Büyük Veri Java HttpURLConnection

Test ortalaması 1019 ms olup, açıklığın ortalamaya sapması %4,8 ve çeyrekler açıklığının ortalamaya sapması %3,3 olduğu kaydedilmiştir.



Şekil 4.71 S4 SOAP Büyük Veri Java HttpURLConnection

Test ortalaması 31818 ms olup, açıklığın ortalamaya sapması %1,5 ve çeyrekler açıklığının ortalamaya sapması %0,2 olduğu kaydedilmiştir.



Şekil 4.72 IDES SOAP Büyük Veri Java HttpURLConnection

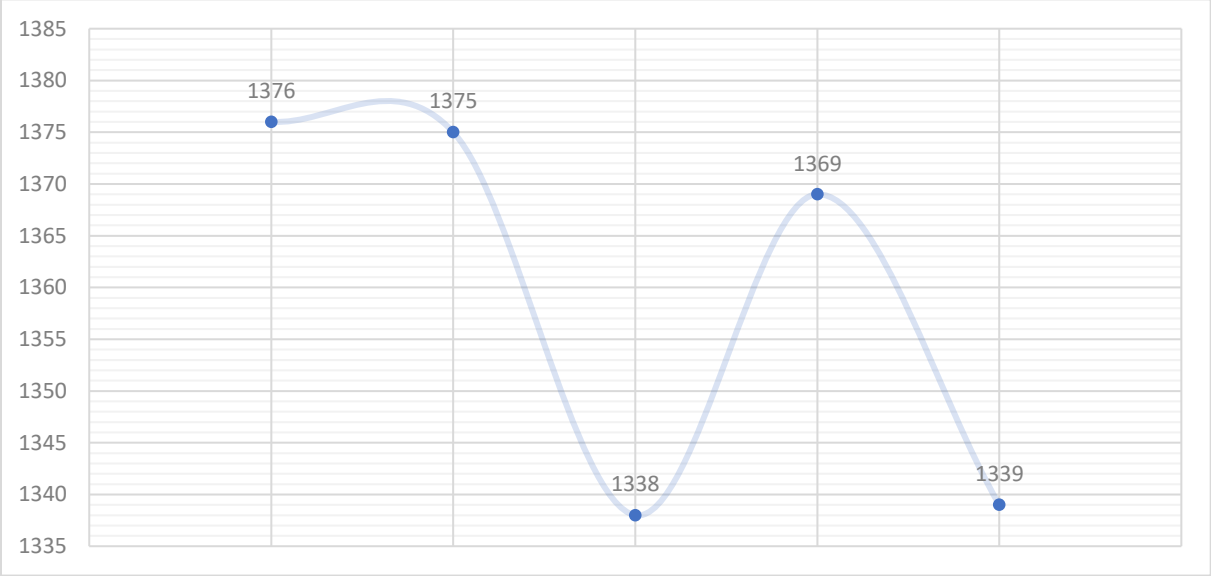
Test ortalaması 33361 ms olup, açıklığın ortalamaya sapması %0,6 ve çeyrekler açıklığının ortalamaya sapması %0,4 olduğu kaydedilmiştir.

Java HttpClient Çalışması

Bu çalışma Java programlama dilinde JDK 19 sürümünde HttpClient sınıfı kullanılarak IntelliJ IDEA ortamında gerçeklenmiş ve Tablo 4.25'teki ve Şekil 4.73-Şekil 4.75 arasındaki sonuçlar elde edilmiştir. Program koduna EK 1.7'de yer verilmiştir.

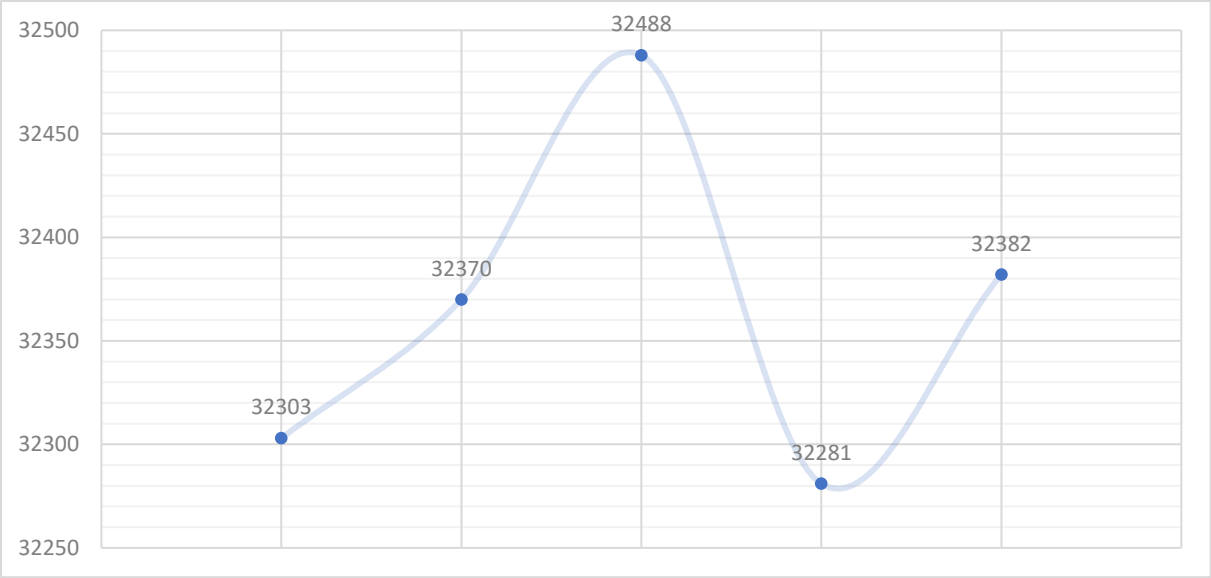
Tablo 4.25 SOAP Büyük Veri Java HttpClient

Tekrar	LOCAL	S4	IDES
1	1376	32303	34328
2	1375	32370	34059
3	1338	32488	34039
4	1369	32281	34233
5	1339	32382	33976
Ortalama	**1359**	**32365**	**34127**



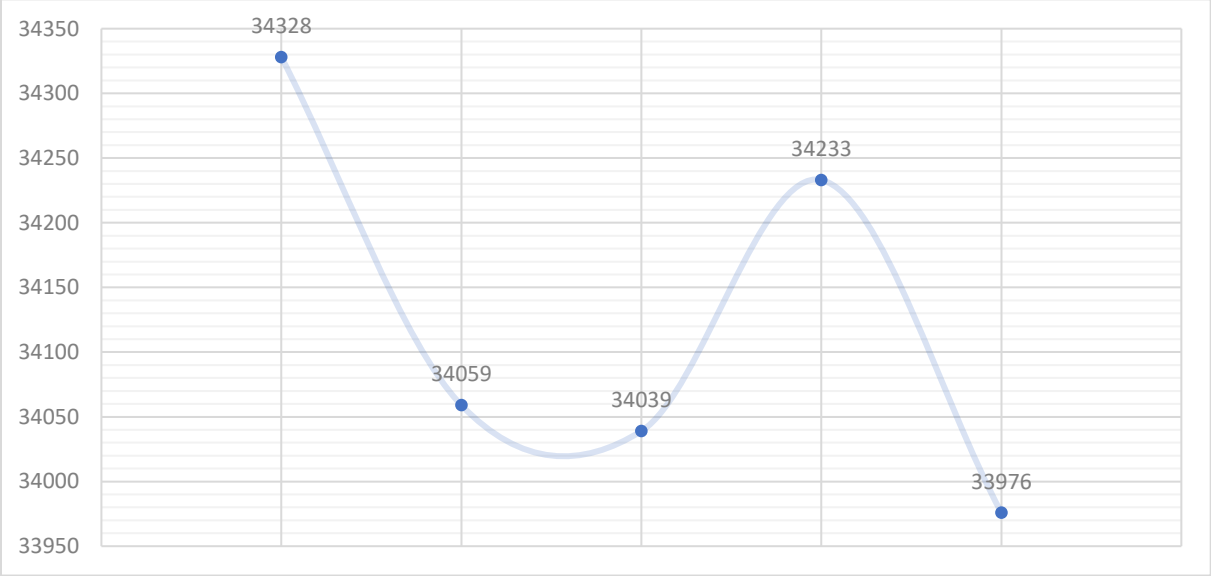
Şekil 4.143 LOCAL SOAP Büyük Veri Java HttpClient

Test ortalaması 1359 ms olup, açıklığın ortalamaya sapması %2,8 ve çeyrekler açıklığının ortalamaya sapması %2,6 olduğu kaydedilmiştir.



Şekil 4.74 S4 SOAP Büyük Veri Java HttpClient

Test ortalaması 32365 ms olup, açıklığın ortalamaya sapması %0,6 ve çeyrekler açıklığının ortalamaya sapması %0,2 olduğu kaydedilmiştir.



Şekil 4.75 IDES SOAP Büyük Veri Java HttpClient

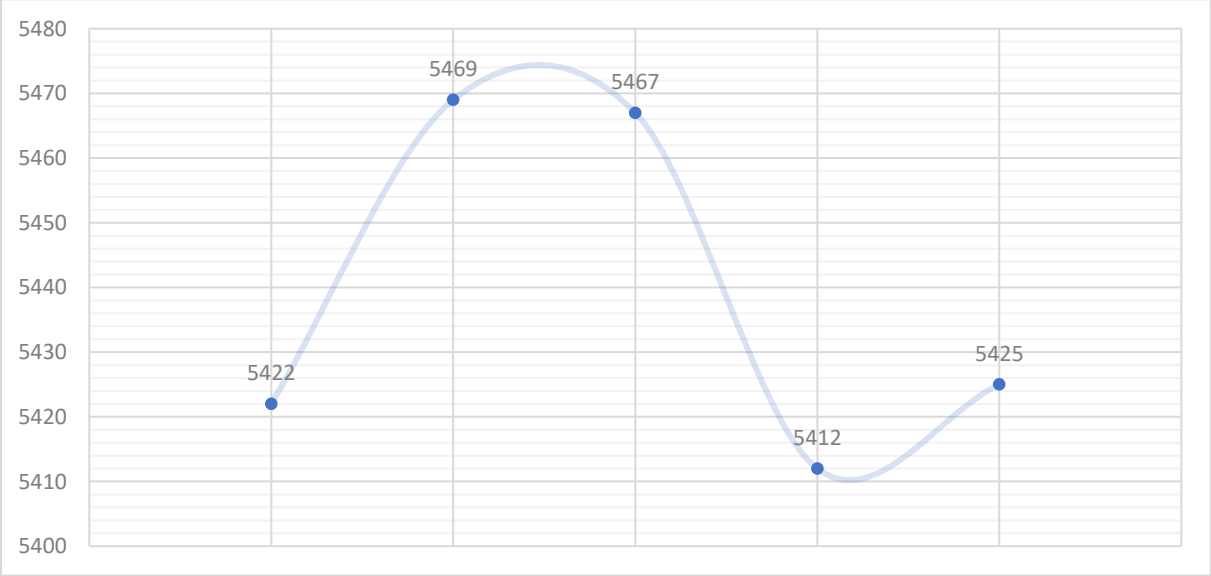
Test ortalaması 34127 ms olup, açıklığın ortalamaya sapması %1 ve çeyrekler açıklığının ortalamaya sapması %0,6 olduğu kaydedilmiştir.

Node.js soap Çalışması

Bu çalışma JavaScript programlama dilinde Node.js v18 sürümünde soap kütüphanesi kullanılarak Visual Studio Code ortamında gerçeklenmiş ve Tablo 4.26'daki ve Şekil 4.76-Şekil 4.78 arasındaki sonuçlar elde edilmiştir. Program koduna EK 1.8'de yer verilmiştir.

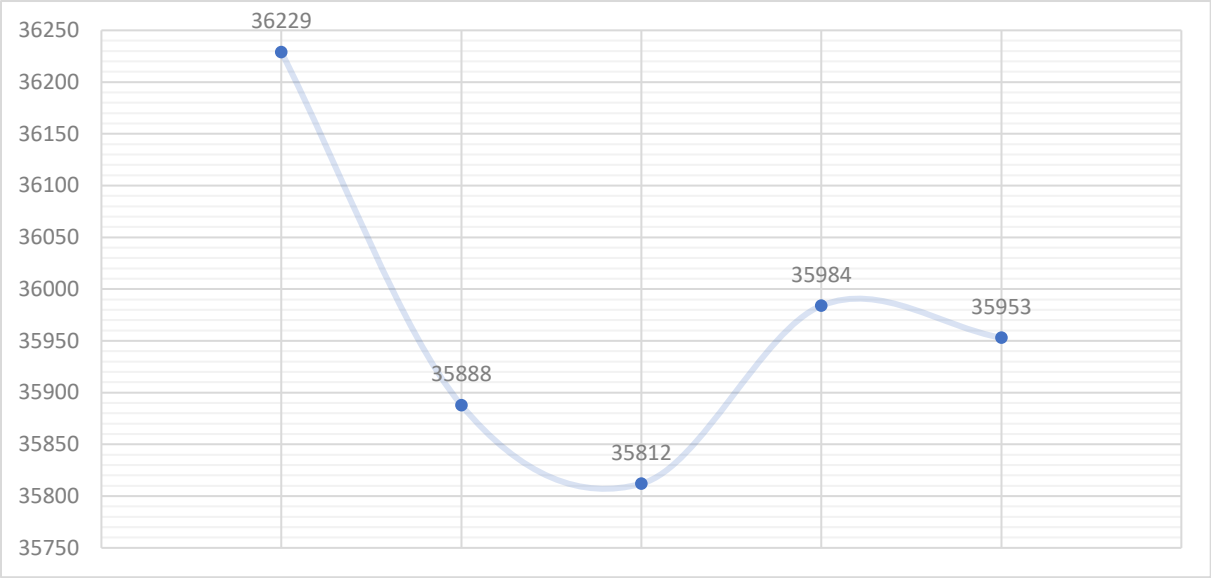
Tablo 4.26 SOAP Büyük Veri Node.js soap

Tekrar	LOCAL	S4	IDES
1	5422	36229	38540
2	5469	35888	38149
3	5467	35812	38341
4	5412	35984	39036
5	5425	35953	37843
Ortalama	**5439**	**35973**	**38382**



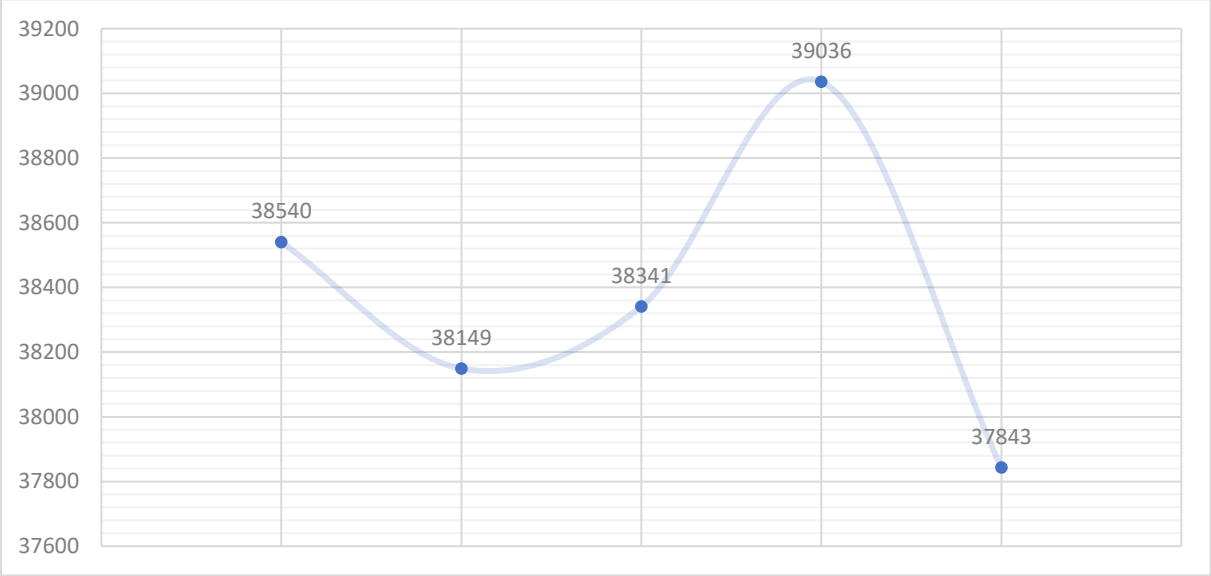
Şekil 4.76 LOCAL SOAP Büyük Veri Node.js soap

Test ortalaması 5439 ms olup, açıklığın ortalamaya sapması %1 ve çeyrekler açıklığının ortalamaya sapması %0,8 olduğu kaydedilmiştir.



Şekil 4.77 S4 SOAP Büyük Veri Node.js soap

Test ortalaması 35973 ms olup, açıklığın ortalamaya sapması %1,2 ve çeyrekler açıklığının ortalamaya sapması %0,3 olduğu kaydedilmiştir.



Şekil 4.78 IDES SOAP Büyük Veri Node.js soap

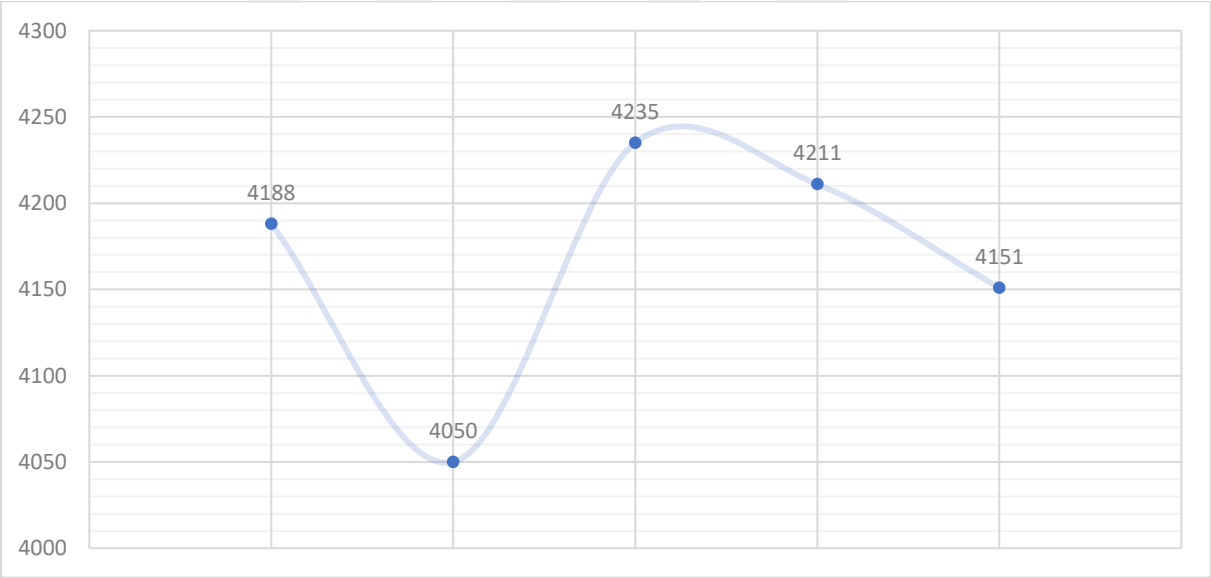
Test ortalaması 38382 ms olup, açıklığın ortalamaya sapması %3,1 ve çeyrekler açıklığının ortalamaya sapması %1 olduğu kaydedilmiştir.

Node.js strong-soap Çalışması

Bu çalışma JavaScript programlama dilinde Node.js v18 sürümünde strong-soap kütüphanesi kullanılarak Visual Studio Code ortamında gerçeklenmiş ve Tablo 4.27'deki ve Şekil 4.79-Şekil 4.81 arasındaki sonuçlar elde edilmiştir. Program koduna EK 1.9'da yer verilmiştir.

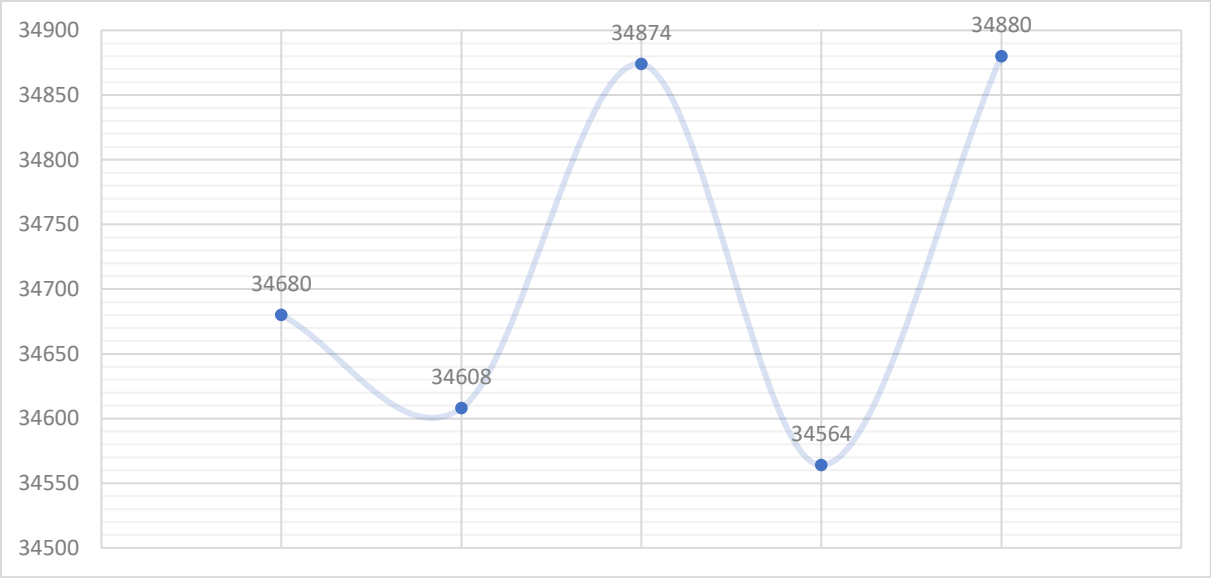
Tablo 4.27 SOAP Büyük Veri Node.js strong-soap

Tekrar	LOCAL	S4	IDES
1	4188	34680	36581
2	4050	34608	36713
3	4235	34874	36515
4	4211	34564	36637
5	4151	34880	37293
Ortalama	**4167**	**34721**	**36748**



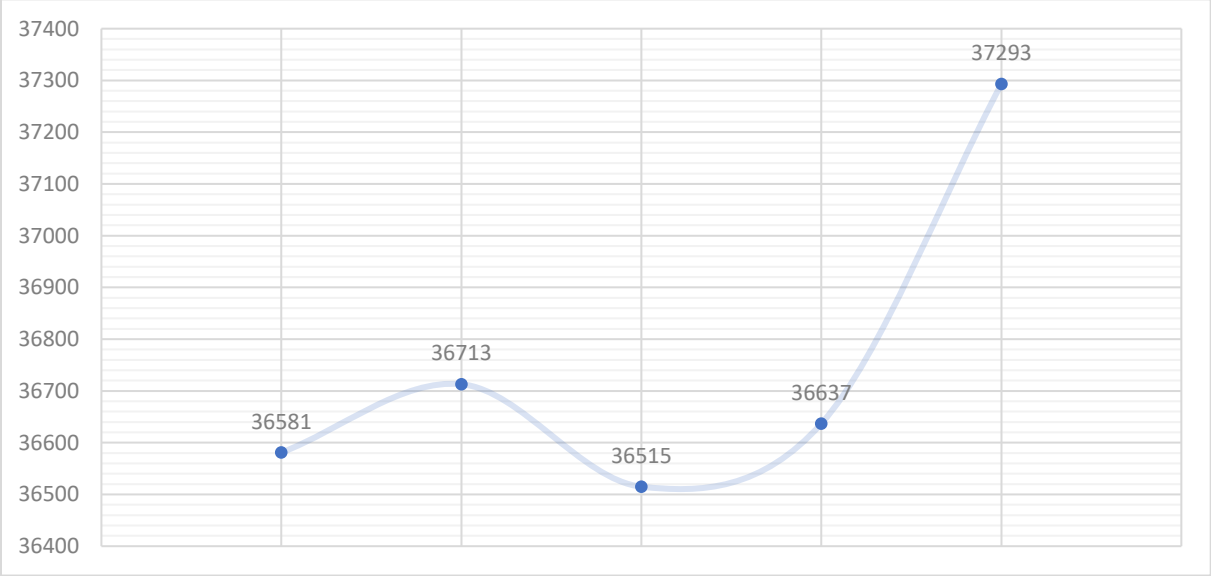
Şekil 4.715 LOCAL SOAP Büyük Veri Node.js strong-soap

Test ortalaması 4167 ms olup, açıklığın ortalamaya sapması %4,4 ve çeyrekler açıklığının ortalamaya sapması %1,4 olduğu kaydedilmiştir.



Şekil 4.80 S4 SOAP Büyük Veri Node.js strong-soap

Test ortalaması 34721 ms olup, açıklığın ortalamaya sapması %0,9 ve çeyrekler açıklığının ortalamaya sapması %0,8 olduğu kaydedilmiştir.



Şekil 4.81 IDES SOAP Büyük Veri Node.js strong-soap

Test ortalaması 36748 ms olup, açıklığın ortalamaya sapması %2,1 ve çeyrekler açıklığının ortalamaya sapması %0,4 olduğu kaydedilmiştir.

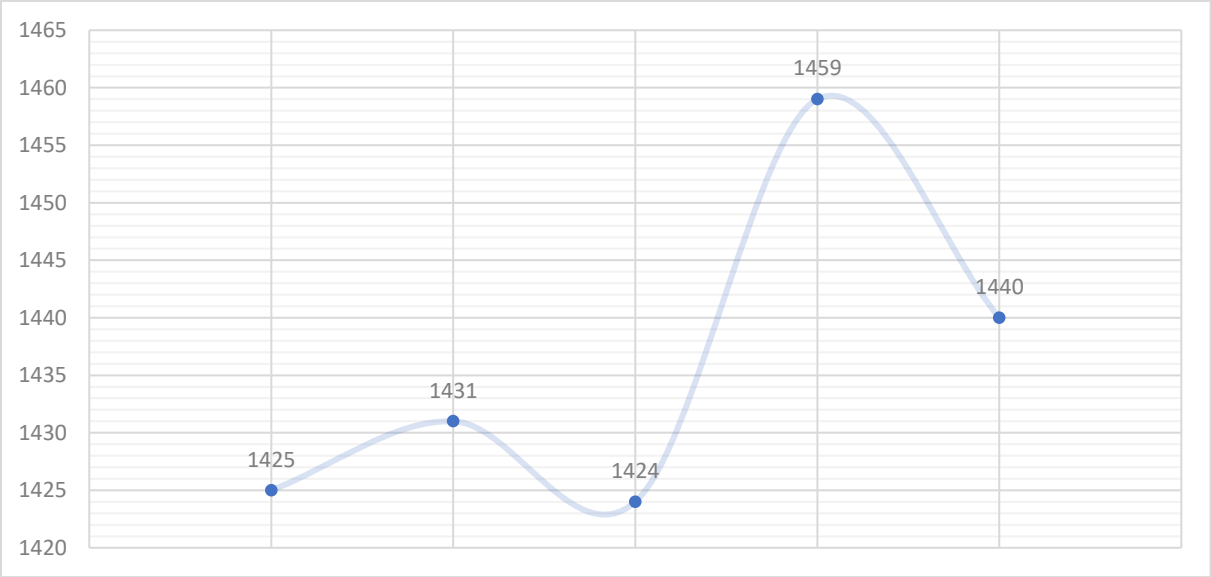
Node.js easy-soap-request Çalışması

Bu çalışma JavaScript programlama dilinde Node.js v18 sürümünde easy-soap-request kütüphanesi kullanılarak Visual Studio Code ortamında gerçeklenmiş ve Tablo 4.28'deki ve

Şekil 4.82-Şekil 4.84 arasındaki sonuçlar elde edilmiştir. Program koduna EK 1.10'da yer verilmiştir.

Tablo 4.28 SOAP Büyük Veri Node.js easy-soap-request

Tekrar	LOCAL	S4	IDES
1	1425	1494	9490
2	1431	1439	9489
3	1424	1423	9472
4	1459	1408	9506
5	1440	1379	9423
Ortalama	**1436**	**1429**	**9476**

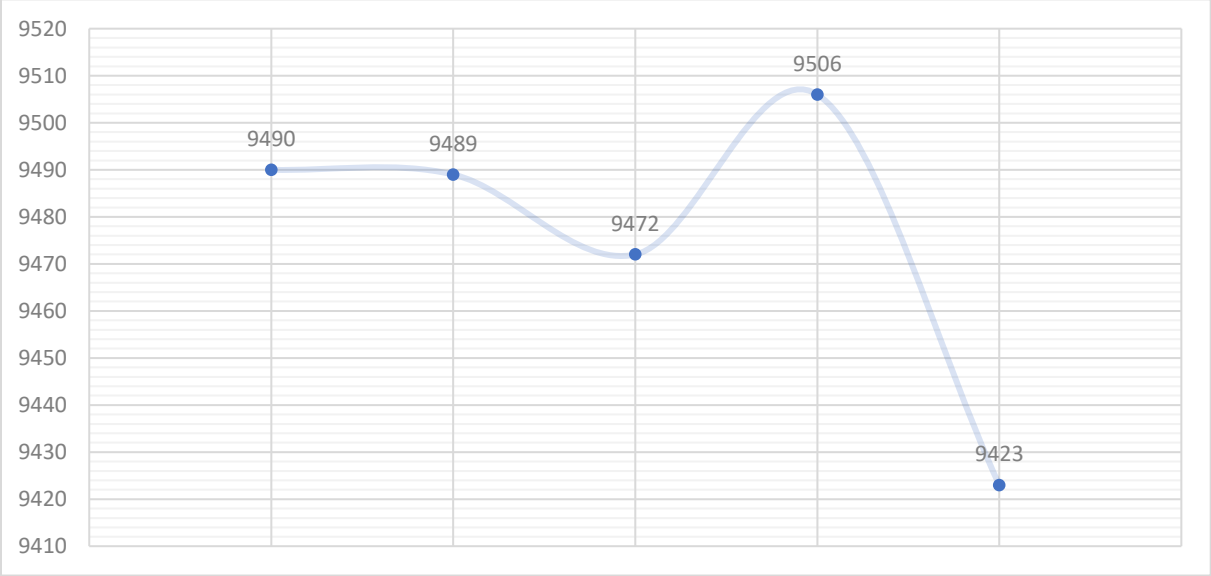


Şekil 4.82 LOCAL SOAP Büyük Veri Node.js easy-soap-request

Test ortalaması 1436 ms olup, açıklığın ortalamaya sapması %2,4 ve çeyrekler açıklığının ortalamaya sapması %1 olduğu kaydedilmiştir.

Şekil 4.83 S4 SOAP Büyük Veri Node.js easy-soap-request

Test ortalaması 1429 ms olup, açıklığın ortalamaya sapması %8 ve çeyrekler açıklığının ortalamaya sapması %2,2 olduğu kaydedilmiştir.



Şekil 4.84 IDES SOAP Büyük Veri Node.js easy-soap-request

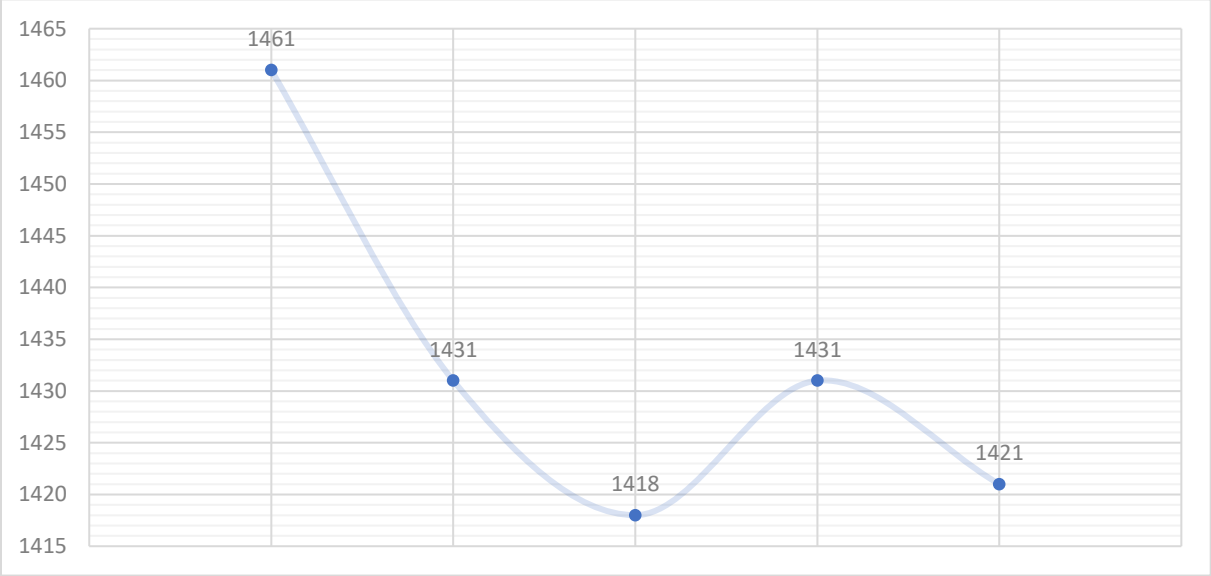
Test ortalaması 9476 ms olup, açıklığın ortalamaya sapması %0,9 ve çeyrekler açıklığının ortalamaya sapması %0,2 olduğu kaydedilmiştir.

Node.js axios Çalışması

Bu çalışma JavaScript programlama dilinde Node.js v18 sürümünde axios kütüphanesi kullanılarak Visual Studio Code ortamında gerçeklenmiş ve Tablo 4.29'daki ve Şekil 4.85-Şekil 4.87 arasındaki sonuçlar elde edilmiştir. Program koduna EK 1.11'de yer verilmiştir.

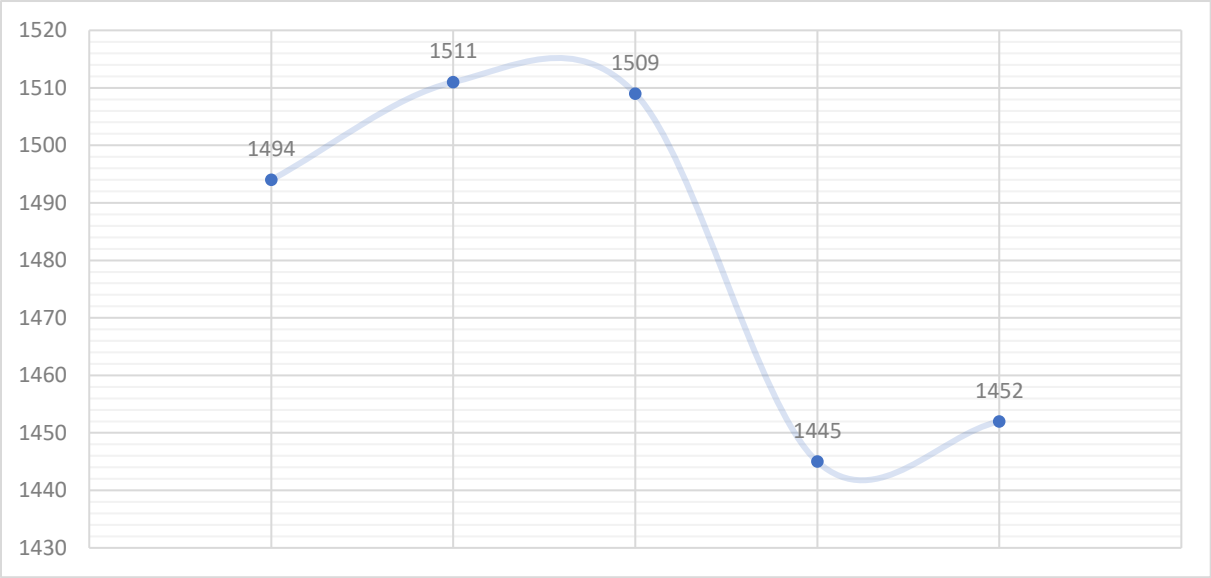
Tablo 4.29 SOAP Büyük Veri Node.js axios

Tekrar	LOCAL	S4	IDES
1	1461	1494	9466
2	1431	1511	9458
3	1418	1509	9359
4	1431	1445	9393
5	1421	1452	9343
Ortalama	**1432**	**1482**	**9404**



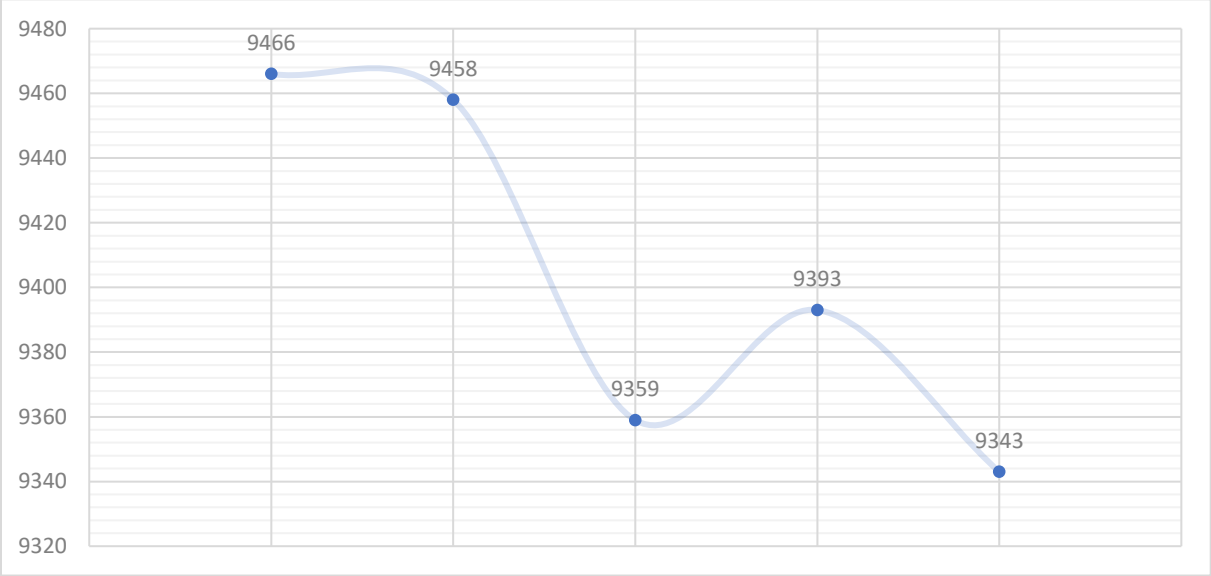
Şekil 4.85 LOCAL SOAP Büyük Veri Node.js axios

Test ortalaması 1432 ms olup, açıklığın ortalamaya sapması %3 ve çeyrekler açıklığının ortalamaya sapması %0,7 olduğu kaydedilmiştir.



Şekil 4.86 S4 SOAP Büyük Veri Node.js axios

Test ortalaması 1482 ms olup, açıklığın ortalamaya sapması %4,5 ve çeyrekler açıklığının ortalamaya sapması %3,8 olduğu kaydedilmiştir.



Şekil 4.87 IDES SOAP Büyük Veri Node.js axios

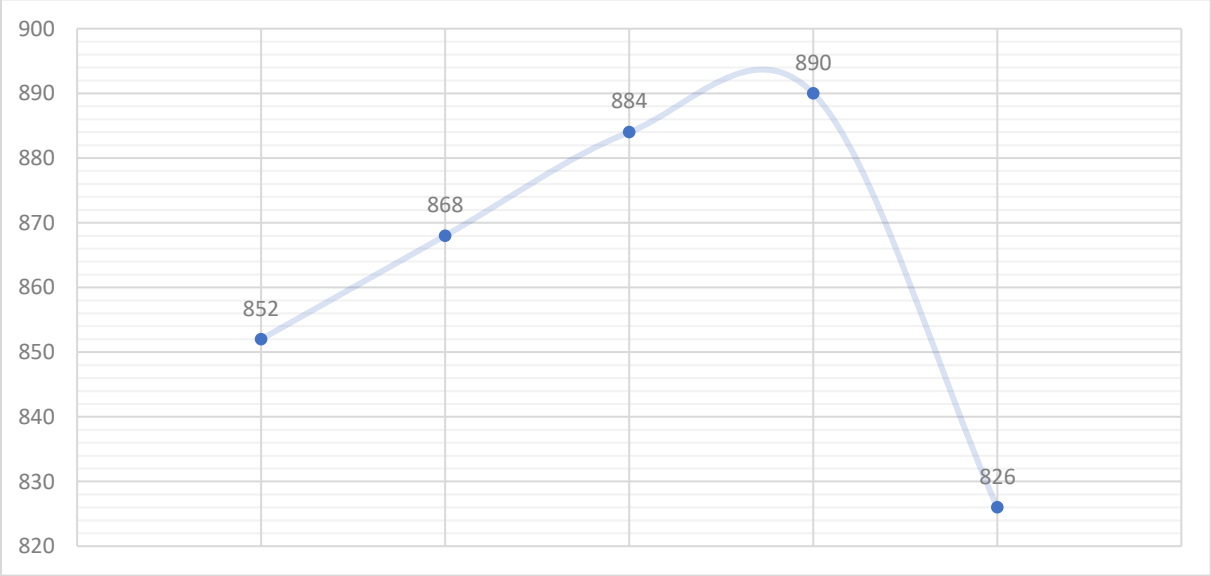
Test ortalaması 9404 ms olup, açıklığın ortalamaya sapması %1,3 ve çeyrekler açıklığının ortalamaya sapması %1,1 olduğu kaydedilmiştir.

Node.js request Çalışması

Bu çalışma JavaScript programlama dilinde Node.js v18 sürümünde request kütüphanesi kullanılarak Visual Studio Code ortamında gerçeklenmiş ve Tablo 4.30'daki ve Şekil 4.88-Şekil 4.90 arasındaki sonuçlar elde edilmiştir. Program koduna EK 1.12'de yer verilmiştir.

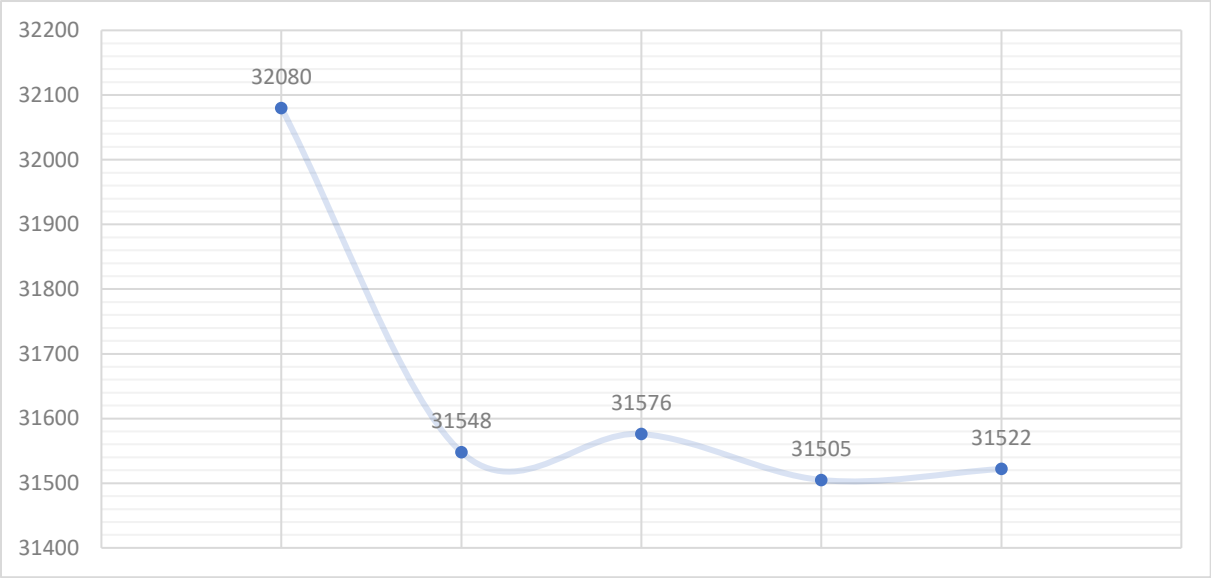
Tablo 4.30 SOAP Büyük Veri Node.js request

Tekrar	LOCAL	S4	IDES
1	852	32080	33529
2	868	31548	33368
3	884	31576	33347
4	890	31505	33282
5	826	31522	33318
Ortalama	**864**	**31646**	**33369**



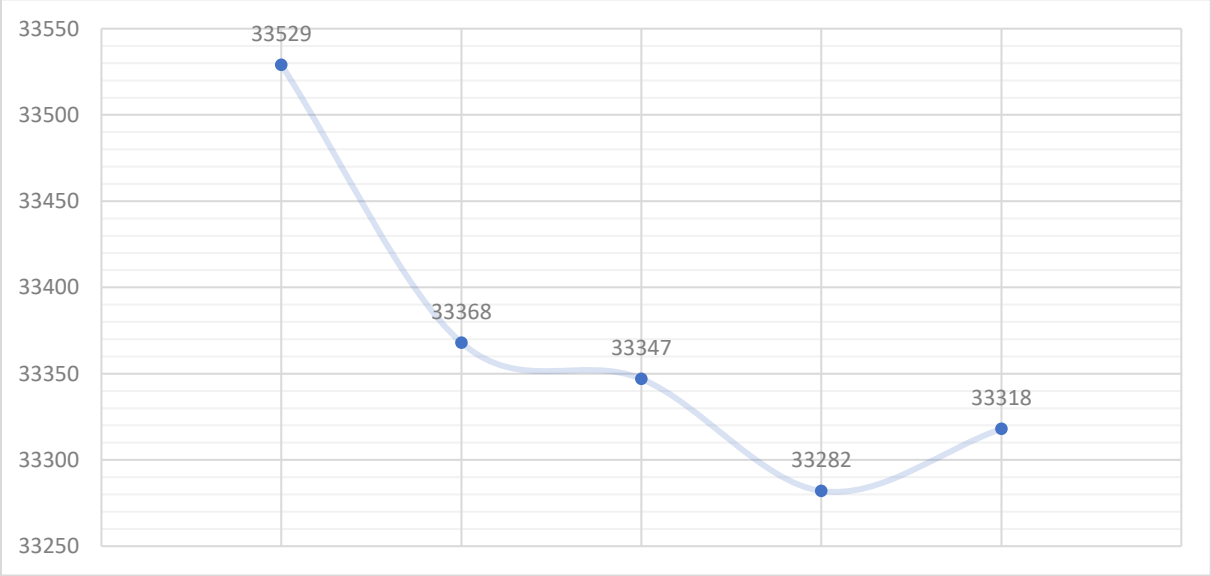
Şekil 4.88 LOCAL SOAP Büyük Veri Node.js request

Test ortalaması 864 ms olup, açıklığın ortalamaya sapması %7,4 ve çeyrekler açıklığının ortalamaya sapması %3,7 olduğu kaydedilmiştir.



Şekil 4.89 S4 LOCAL SOAP Büyük Veri Node.js request

Test ortalaması 31646 ms olup, açıklığın ortalamaya sapması %1,8 ve çeyrekler açıklığının ortalamaya sapması %0,2 olduğu kaydedilmiştir.



Şekil 4.90 IDES LOCAL SOAP Büyük Veri Node.js request

Test ortalaması 33369 ms olup, açıklığın ortalamaya sapması %0,7 ve çeyrekler açıklığının ortalamaya sapması %0,1 olduğu kaydedilmiştir.

4.1.5. Büyük Veriyle REST JSON Uygulamaları

Bu tez çalışmasının yeni bölümünde ikinci olarak JSON formatındaki büyük verilerle REST API çalışılmıştır. Küçük verilerle çalışmalara göre 100-10.000 kat daha büyük ve her sunucuda aynı verilerle çalışılmış böylelikle veri boyutlarının SAP tarafında, API üzerinde, dosya formatı boyutlarında ve istemci tarafında yorumlanmasındaki performans farkları daha net ortaya koyulmuştur. Üç farklı programlama dili kullanılıp bunlar sırasıyla C#, Java ve JavaScript(Node.js) dilleri olmuştur. Bu programlama dillerinin de kendi içlerinde farklı kütüphaneler ve sınıflar kullanılarak test sonuçları her programlama dili içerisinde çeşitlendirilmiştir. Her bir program uygun olduğu geliştirme ortamında geliştirilip derlenmiş ve çalıştırılmıştır. Her bir çalıştırma döngüsü kendi içerisinde üç ana başlıktan oluşmaktadır; istek paketlerinin oluşturulması, gönderilmesi ve cevabın alınması, cevabın yorumlanması. Bu üç başlığın toplam çalışma süresi milisaniye(ms) biriminden kaydedilmiş ve tez çalışması kapsamında yorumlanmıştır. Bu tez çalışması kapsamında yorumlanan veriler; programların toplam çalışma süreleri değil, belirtilen bu üç ana başlıkta geçirilen sürelerdir. SAP sunucusunun REST API ile JSON formatında döndürdüğü cevaba EK 2.1'de yer verilmiştir.

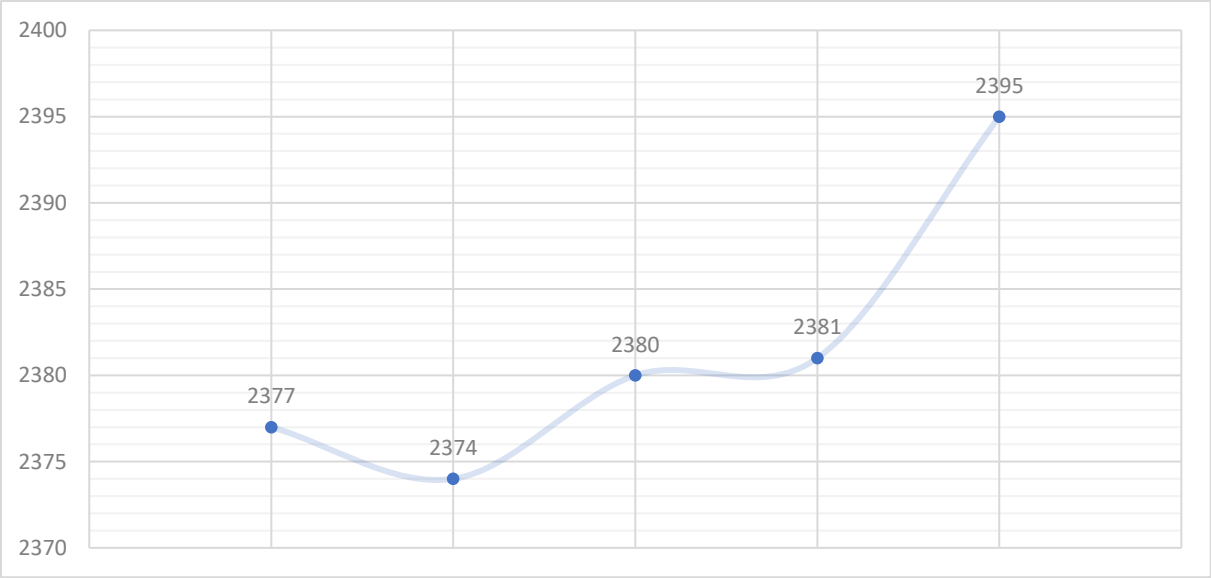
Kullanılan farklı diller ve bu dillerde kullanılan farklı kütüphaneler veya sınıflar REST isteklerini farklı yöntemlerle oluşturup, gönderip, EK 2.1'de yer verilen aynı cevabı almıştır. Bu cevap; SAP tarafında veritabanının farklı alanlarından SQL sorguları ve ABAP fonksiyonlarıyla çekilmiş, paketlenmiş ve gönderilmiştir.

C# HttpClient Çalışması

Bu çalışma C# programlama dilinde .NET 7.0 platformunda HttpClient sınıfı kullanılarak Visual Studio ortamında gerçeklenmiş ve Tablo 4.31'deki ve Şekil 4.91-Şekil 4.93 arasındaki sonuçlar elde edilmiştir. Program koduna EK 2.2'de yer verilmiştir.

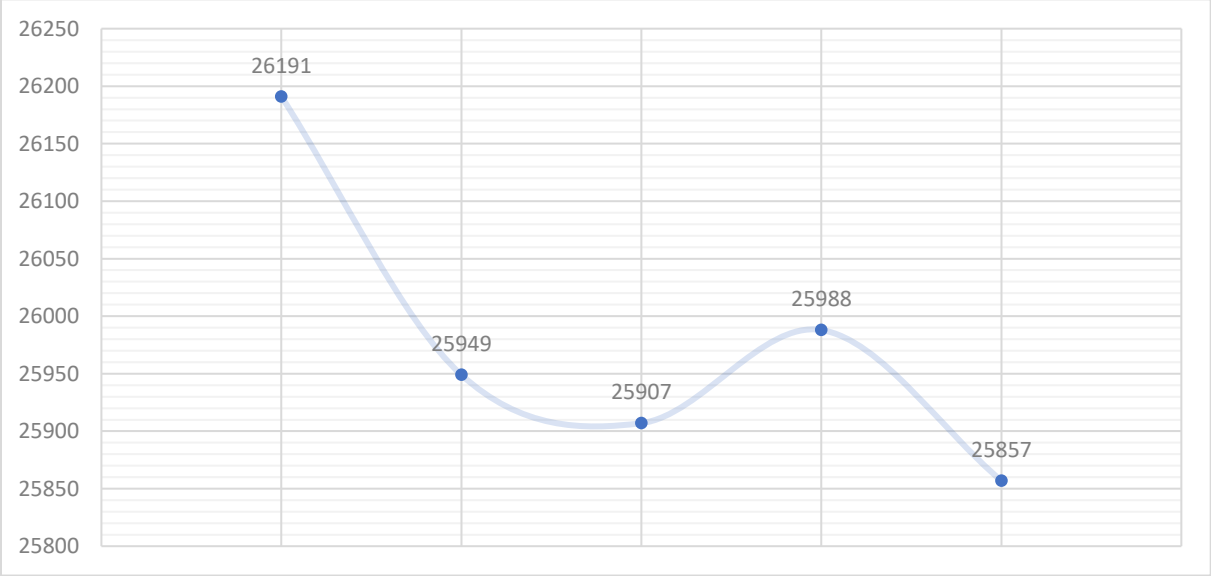
Tablo 4.323 REST JSON Büyük Veri C# HttpClient

Tekrar	LOCAL	S4	IDES
1	2377	26191	28182
2	2374	25949	28315
3	2380	25907	28380
4	2381	25988	28448
5	2395	25857	28359
Ortalama	**2381**	**25978**	**28337**



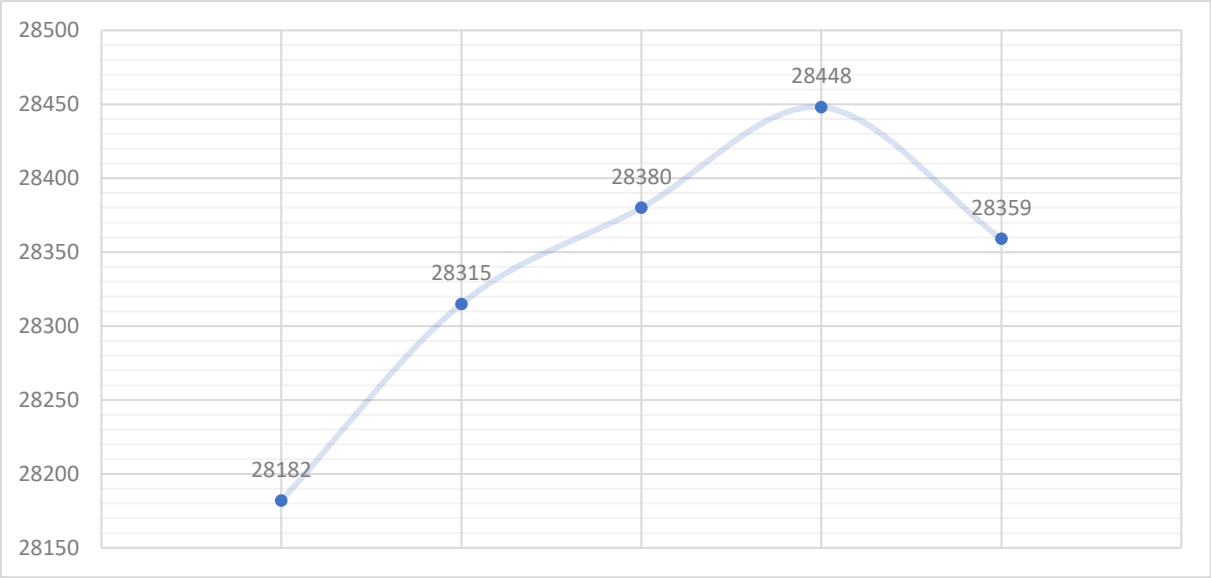
Şekil 4.916 LOCAL REST JSON Büyük Veri C# HttpClient

Test ortalaması 2381 ms olup, açıklığın ortalamaya sapması %0,9 ve çeyrekler açıklığının ortalamaya sapması %0,2 olduğu kaydedilmiştir.



Şekil 4.92 S4 REST JSON Büyük Veri C# HttpClient

Test ortalaması 25978 ms olup, açıklığın ortalamaya sapması %1,3 ve çeyrekler açıklığının ortalamaya sapması %0,3 olduğu kaydedilmiştir.



Şekil 4.93 IDES REST JSON Büyük Veri C# HttpClient

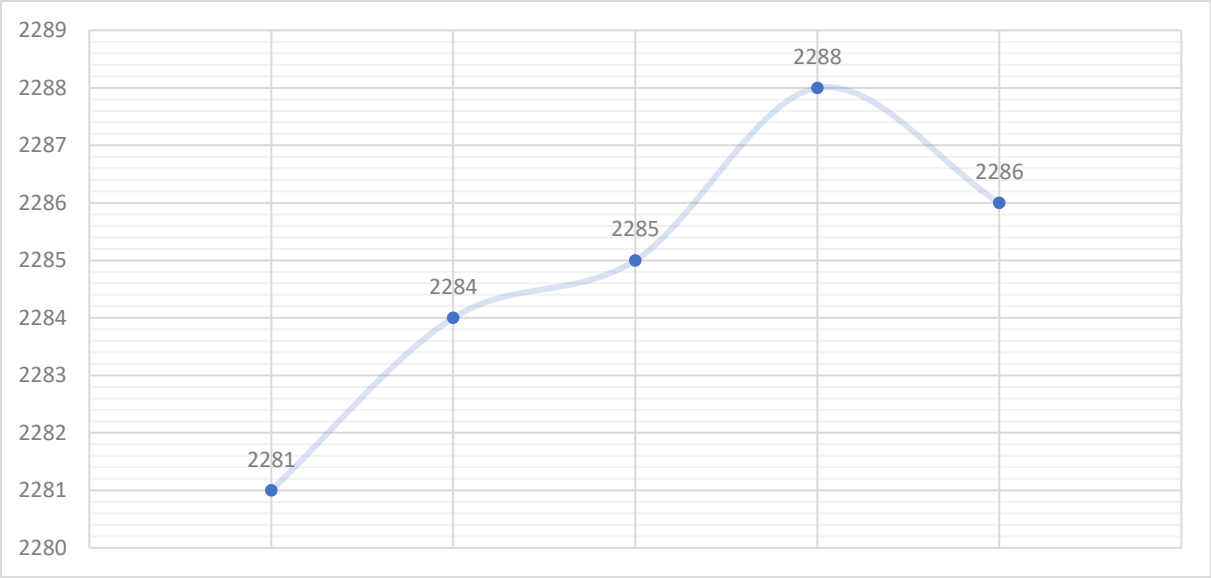
Test ortalaması 28337 ms olup, açıklığın ortalamaya sapması %0,9 ve çeyrekler açıklığının ortalamaya sapması %0,2 olduğu kaydedilmiştir.

C# RestSharp Çalışması

Bu çalışma C# programlama dilinde .NET 7.0 platformunda RestSharp sınıfı kullanılarak Visual Studio ortamında gerçeklenmiş ve Tablo 4.32'deki ve Şekil 4.94-Şekil 4.96 arasındaki sonuçlar elde edilmiştir. Program koduna EK 2.3'te yer verilmiştir.

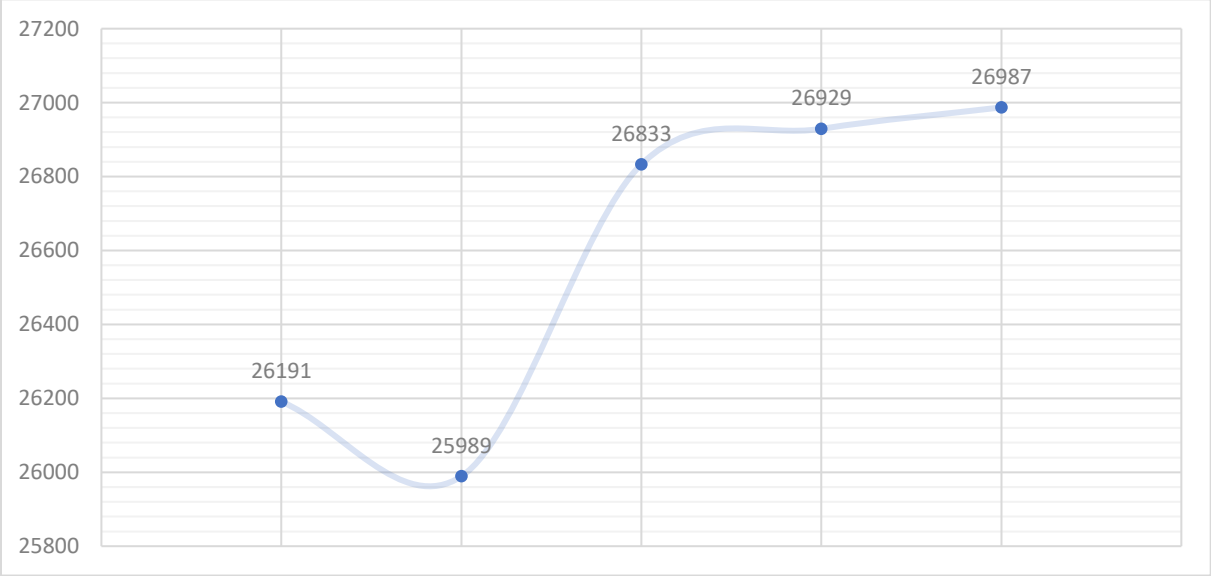
Tablo 4.32 REST JSON Büyük Veri C# RestSharp

Tekrar	LOCAL	S4	IDES
1	2281	26191	28555
2	2284	25989	28556
3	2285	26833	28440
4	2288	26929	28412
5	2286	26987	28581
Ortalama	**2285**	**26586**	**28509**



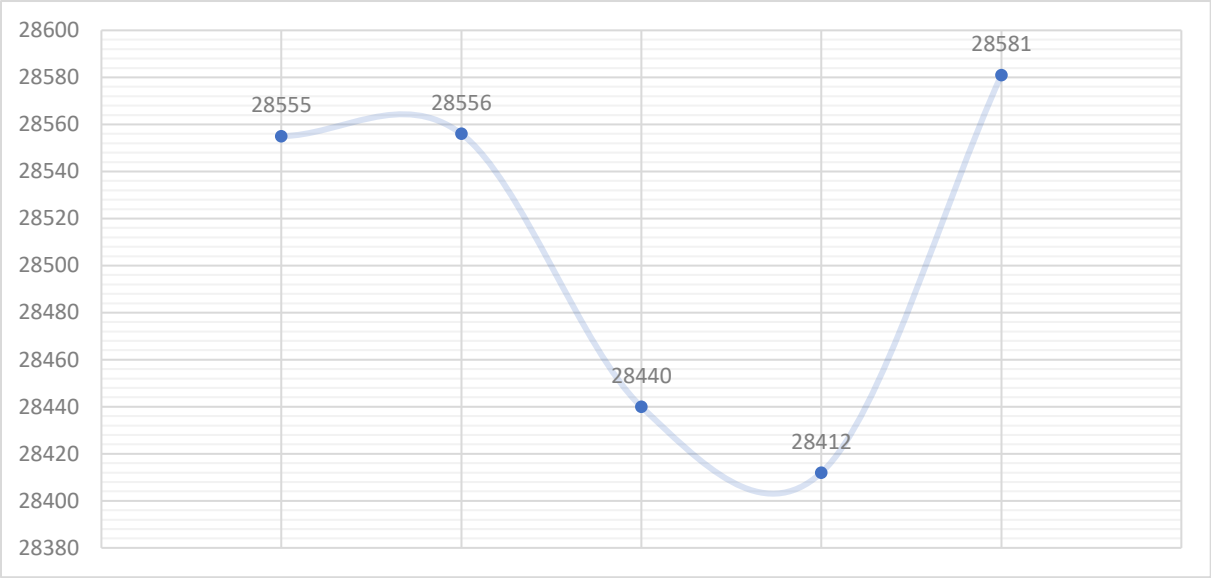
Şekil 4.94 LOCAL REST JSON Büyük Veri C# RestSharp

Test ortalaması 2285 ms olup, açıklığın ortalamaya sapması %0,3 ve çeyrekler açıklığının ortalamaya sapması %0,1 olduğu kaydedilmiştir.



Şekil 4.917 S4 REST JSON Büyük Veri C# RestSharp

Test ortalaması 26586 ms olup, açıklığın ortalamaya sapması %3,8 ve çeyrekler açıklığının ortalamaya sapması %2,8 olduğu kaydedilmiştir.



Şekil 4.96 IDES REST JSON Büyük Veri C# RestSharp

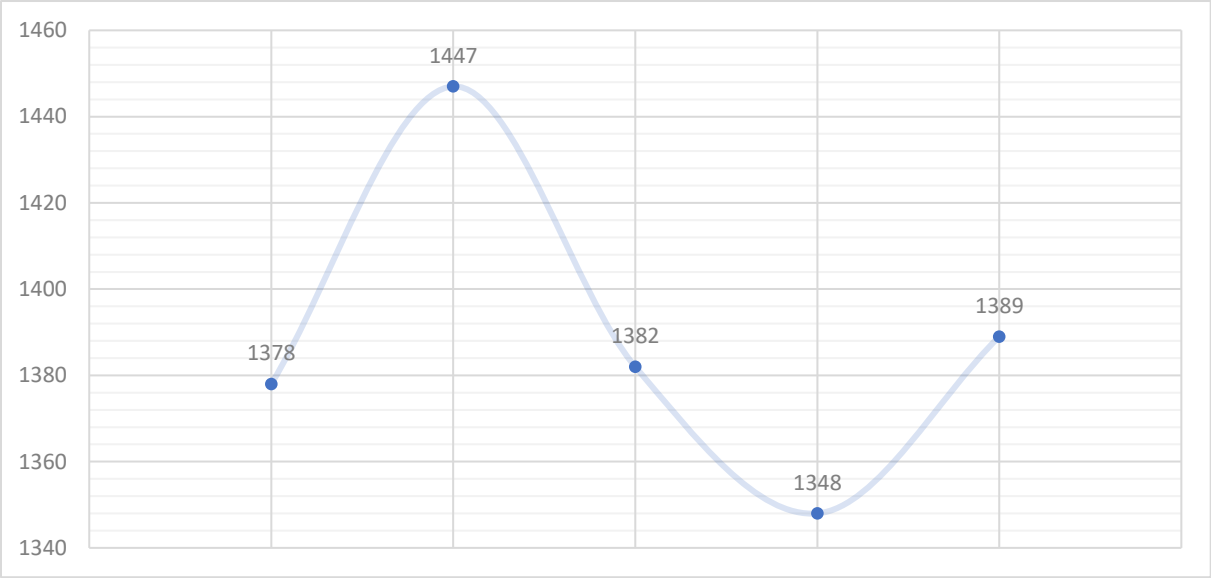
Test ortalaması 28509 ms olup, açıklığın ortalamaya sapması %0,6 ve çeyrekler açıklığının ortalamaya sapması %0,4 olduğu kaydedilmiştir.

Java HttpURLConnection Çalışması

Bu çalışma Java programlama dilinde JDK 19 sürümünde HttpURLConnection sınıfı kullanılarak IntelliJ IDEA ortamında gerçeklenmiş ve Tablo 4.33'teki ve Şekil 4.97-Şekil 4.99 arasındaki sonuçlar elde edilmiştir. Program koduna EK 2.4'te yer verilmiştir.

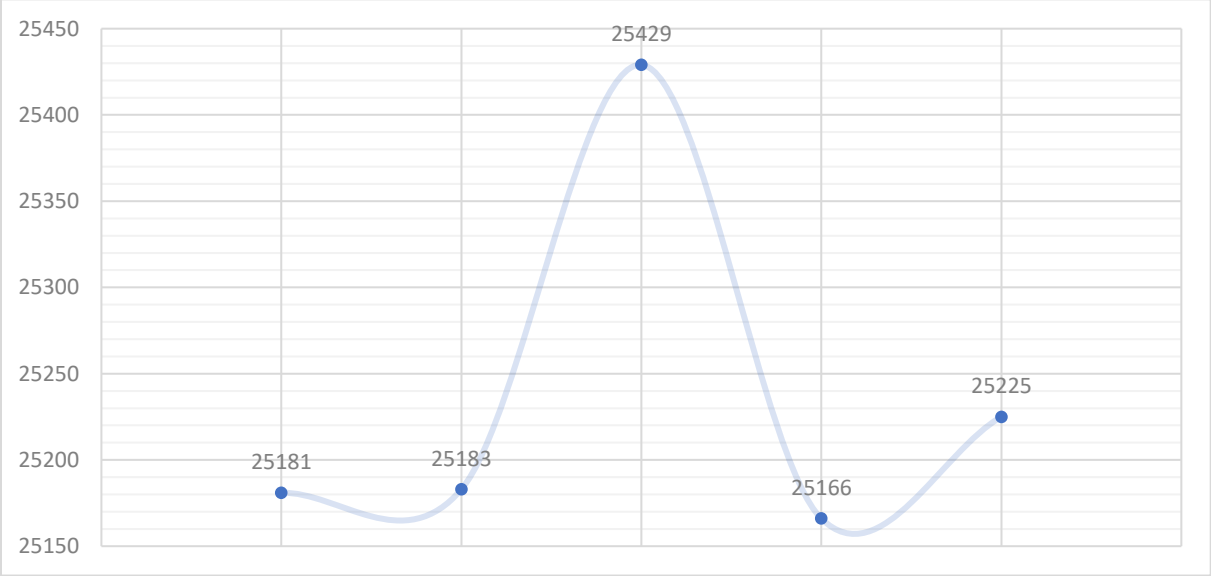
Tablo 4.33 REST JSON Büyük Veri Java HttpURLConnection

Tekrar	LOCAL	S4	IDES
1	1378	25181	27121
2	1447	25183	27392
3	1382	25429	26894
4	1348	25166	27174
5	1389	25225	27035
Ortalama	**1389**	**25237**	**27123**



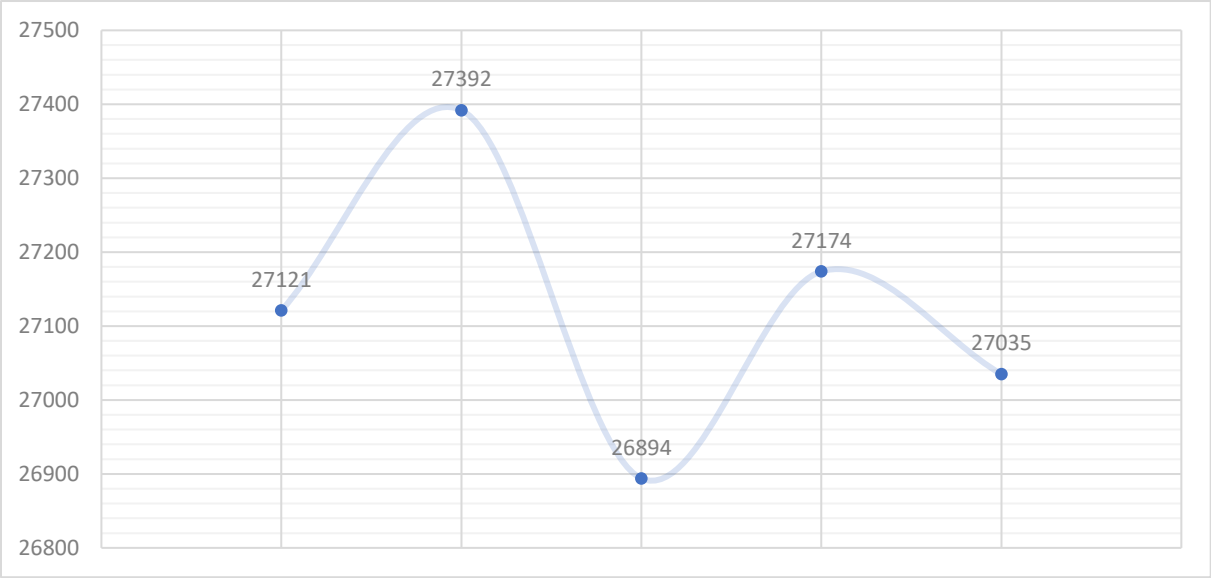
Şekil 4.97 LOCAL REST JSON Büyük Veri Java HttpURLConnection

Test ortalaması 1389 ms olup, açıklığın ortalamaya sapması %7,1 ve çeyrekler açıklığının ortalamaya sapması %0,8 olduğu kaydedilmiştir.



Şekil 4.98 S4 REST JSON Büyük Veri Java HttpURLConnection

Test ortalaması 25237 ms olup, açıklığın ortalamaya sapması %1 ve çeyrekler açıklığının ortalamaya sapması %0,2 olduğu kaydedilmiştir.



Şekil 4.99 IDES REST JSON Büyük Veri Java HttpURLConnection

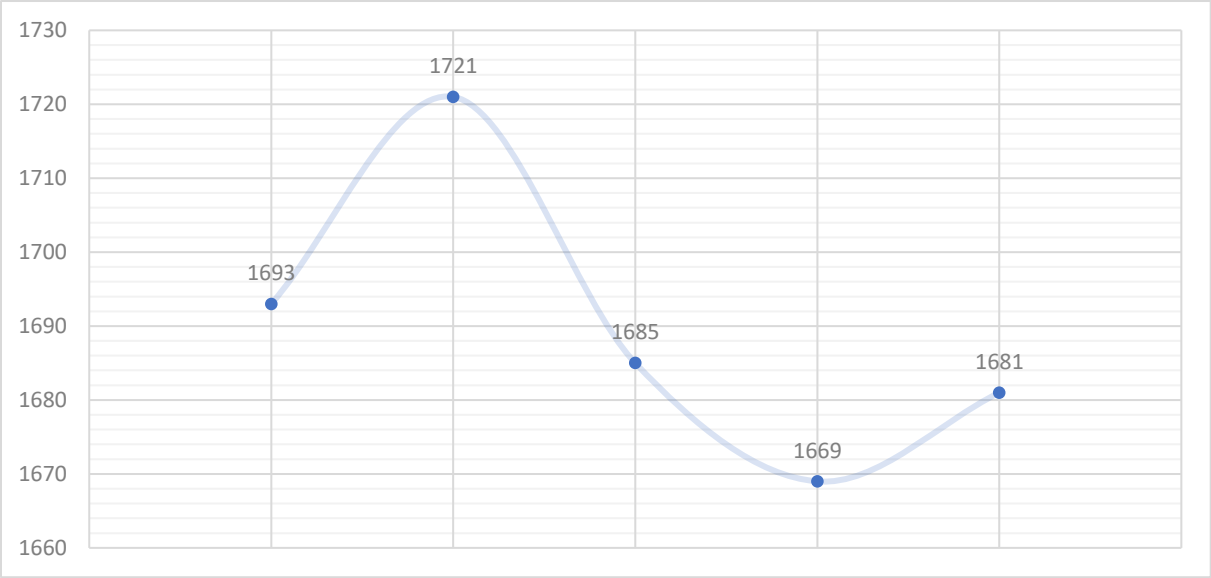
Test ortalaması 27123 ms olup, açıklığın ortalamaya sapması %1,8 ve çeyrekler açıklığının ortalamaya sapması %0,5 olduğu kaydedilmiştir.

Java HttpClient Çalışması

Bu çalışma Java programlama dilinde JDK 19 sürümünde HttpClient sınıfı kullanılarak IntelliJ IDEA ortamında gerçeklenmiş ve Tablo 4.34'teki ve Şekil 4.100-Şekil 4.102 arasındaki sonuçlar elde edilmiştir. Program koduna EK 2.5'te yer verilmiştir.

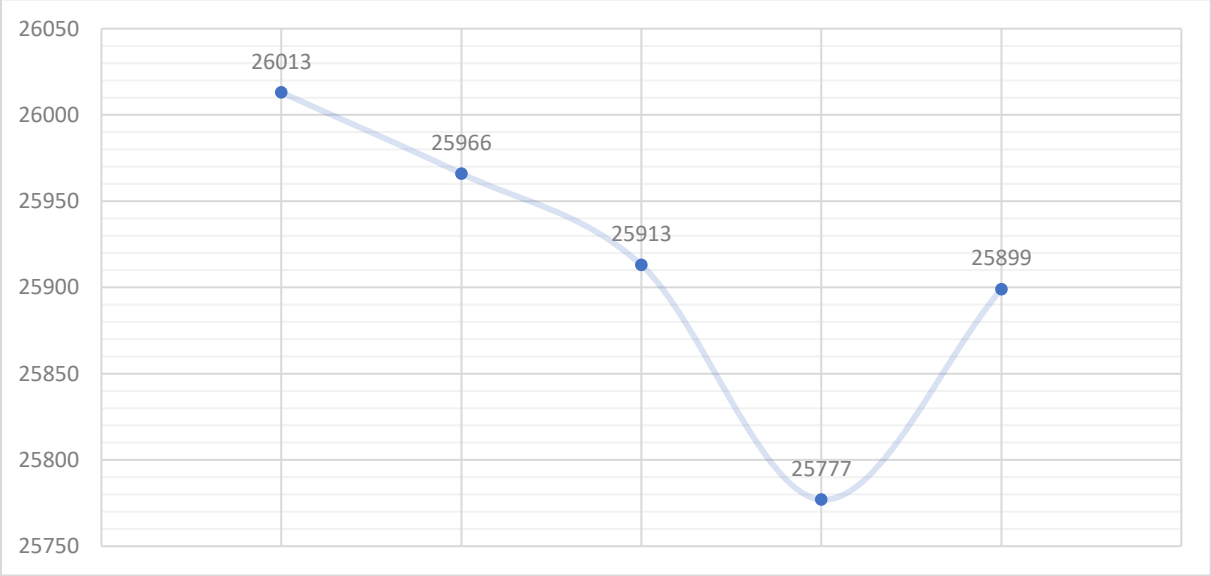
Tablo 4.34 REST JSON Büyük Veri Java HttpClient

Tekrar	LOCAL	S4	IDES
1	1693	26013	28060
2	1721	25966	27690
3	1685	25913	27704
4	1669	25777	27721
5	1681	25899	27774
Ortalama	**1690**	**25914**	**27790**



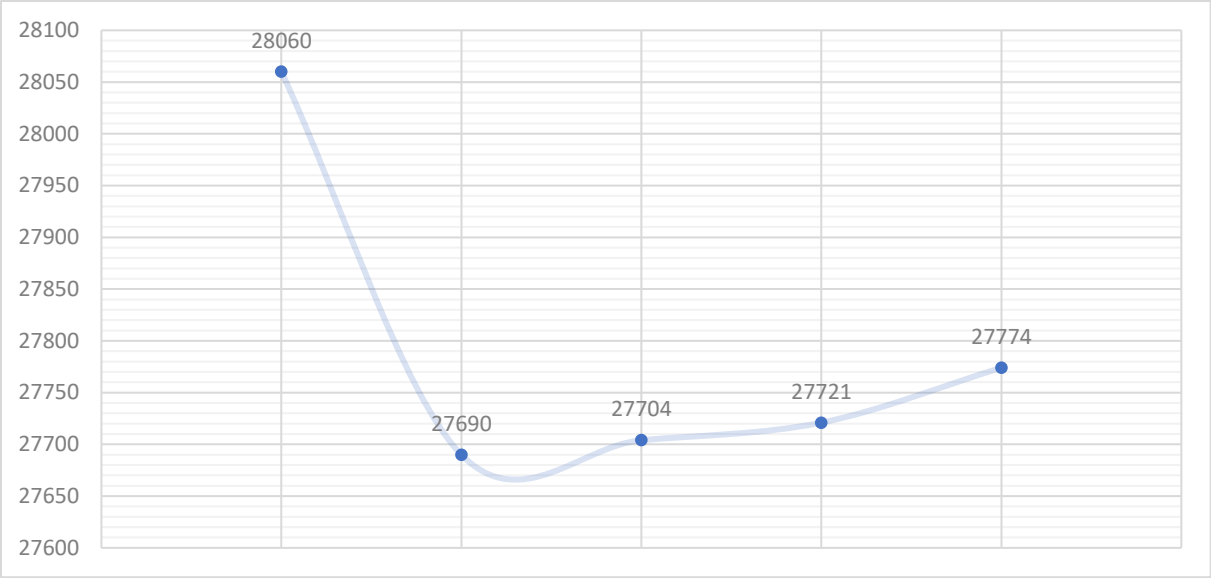
Şekil 4.18 LOCAL REST JSON Büyük Veri Java HttpClient

Test ortalaması 1690 ms olup, açıklığın ortalamaya sapması %3,1 ve çeyrekler açıklığının ortalamaya sapması %0,7 olduğu kaydedilmiştir.



Şekil 4.101 S4 REST JSON Büyük Veri Java HttpClient

Test ortalaması 25914 ms olup, açıklığın ortalamaya sapması %0,9 ve çeyrekler açıklığının ortalamaya sapması %0,3 olduğu kaydedilmiştir.



Şekil 4.102 IDES REST JSON Büyük Veri Java HttpClient

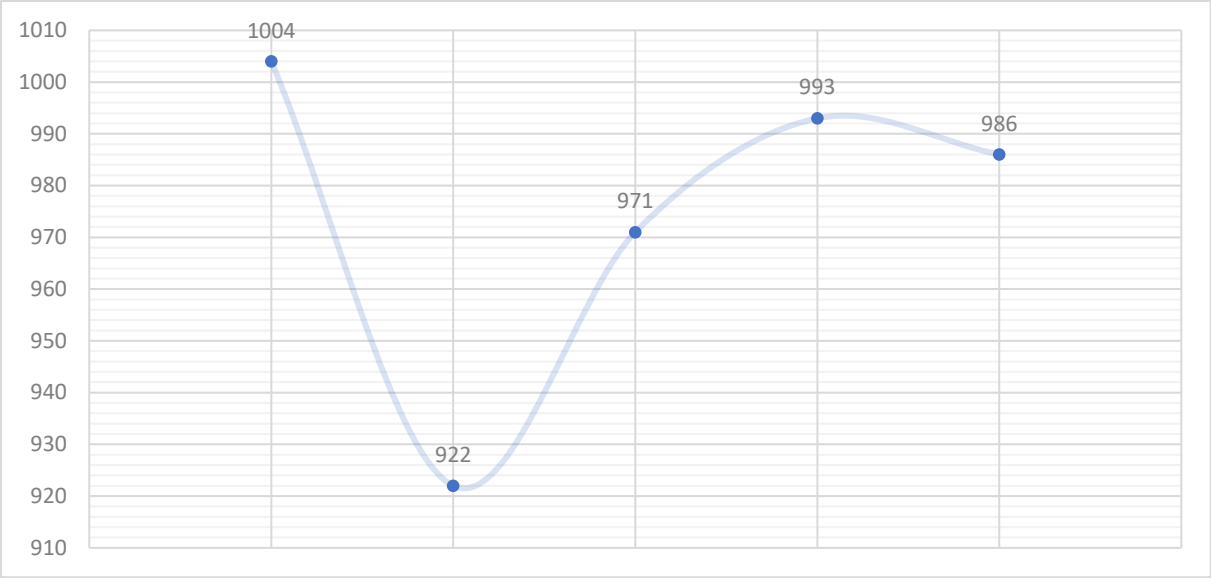
Test ortalaması 27790 ms olup, açıklığın ortalamaya sapması %1,3 ve çeyrekler açıklığının ortalamaya sapması %0,3 olduğu kaydedilmiştir.

Node.js request Çalışması

Bu çalışma JavaScript programlama dilinde Node.js v18 sürümünde request kütüphanesi kullanılarak Visual Studio Code ortamında gerçeklenmiş ve Tablo 4.35'teki ve Şekil 4.103-Şekil 4.105 arasındaki sonuçlar elde edilmiştir. Program koduna EK 2.6'da yer verilmiştir.

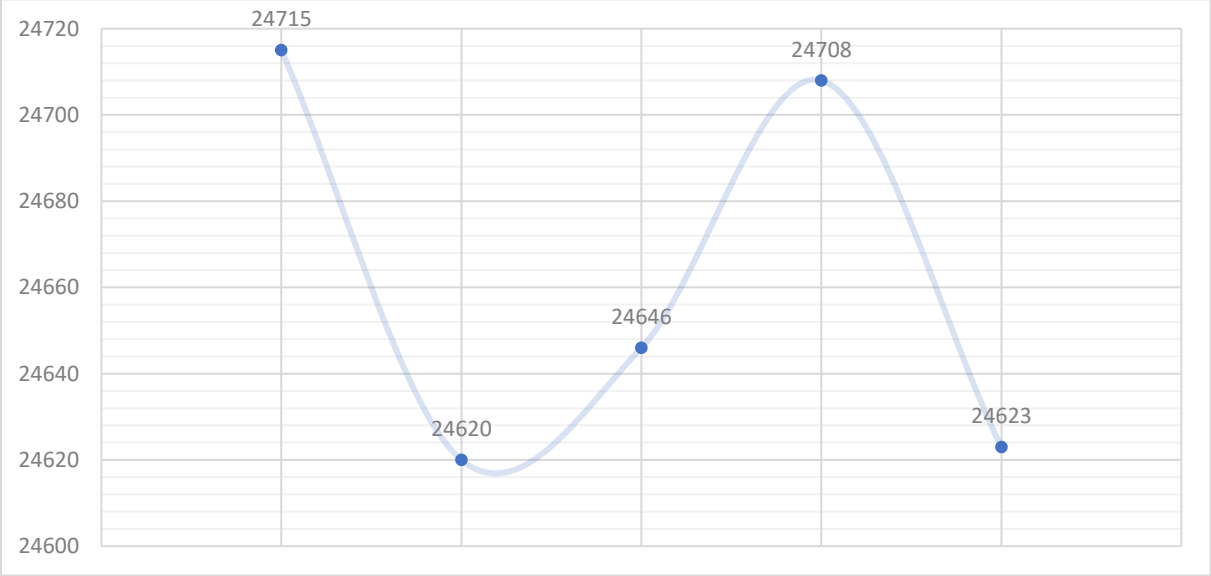
Tablo 4.35 REST JSON Büyük Veri Node.js request

Tekrar	LOCAL	S4	IDES
1	1004	24715	26892
2	922	24620	26705
3	971	24646	26865
4	993	24708	27408
5	986	24623	27437
Ortalama	**975**	**24662**	**27061**



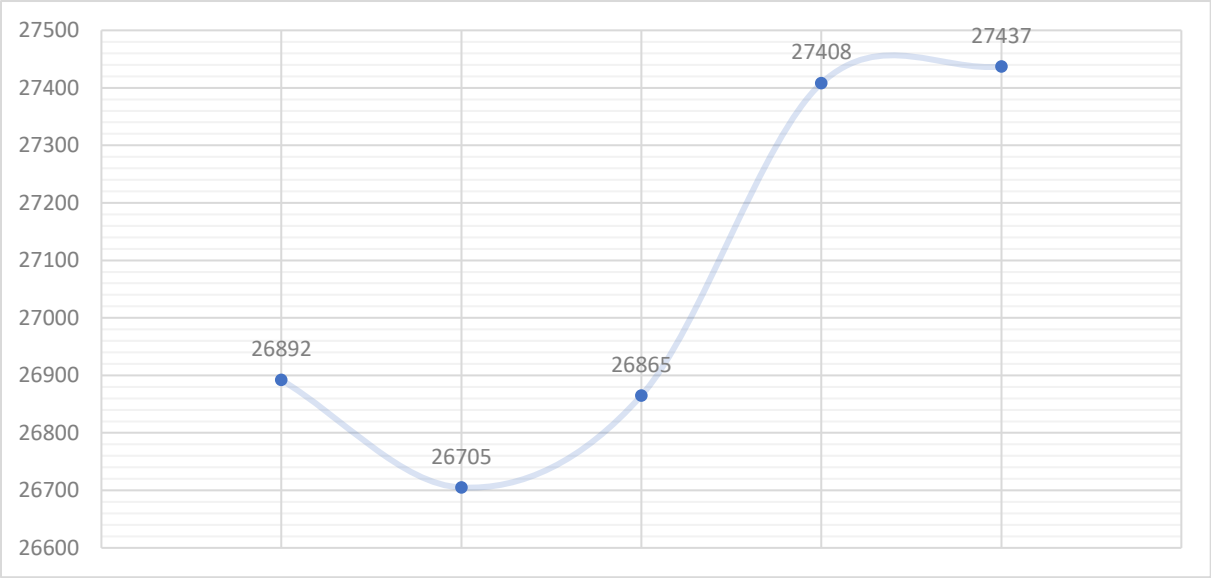
Şekil 4.19 LOCAL REST JSON Büyük Veri Node.js request

Test ortalaması 975 ms olup, açıklığın ortalamaya sapması %8,4 ve çeyrekler açıklığının ortalamaya sapması %2,3 olduğu kaydedilmiştir.



Şekil 4.104 S4 REST JSON Büyük Veri Node.js request

Test ortalaması 24662 ms olup, açıklığın ortalamaya sapması %0,4 ve çeyrekler açıklığının ortalamaya sapması %0,3 olduğu kaydedilmiştir.



Şekil 4.1020 IDES REST JSON Büyük Veri Node.js request

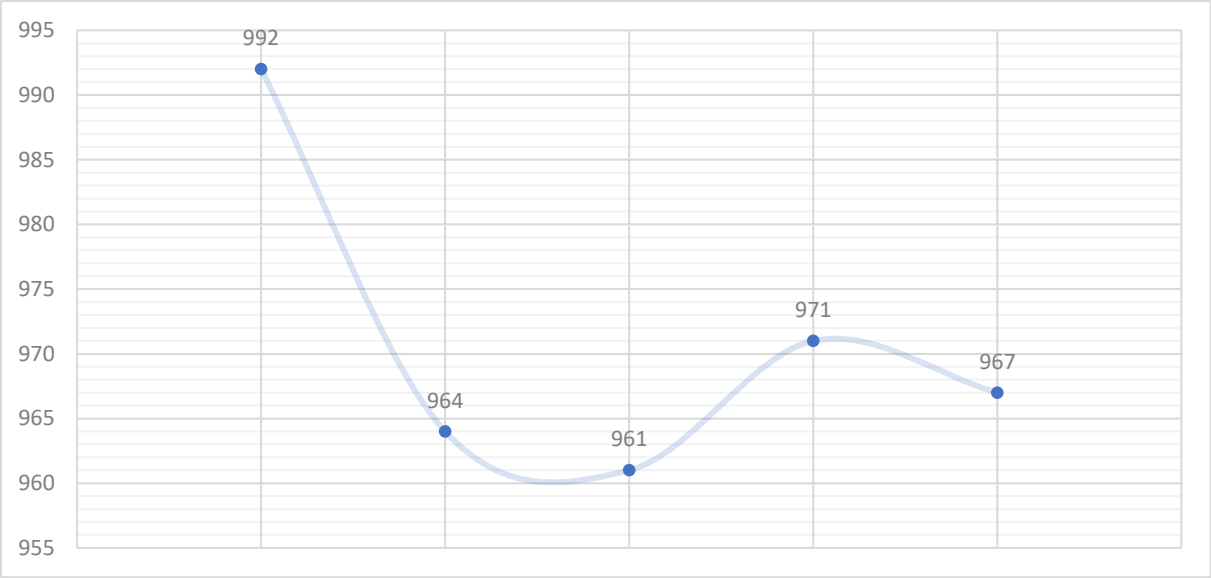
Test ortalaması 27061 ms olup, açıklığın ortalamaya sapması %2,7 ve çeyrekler açıklığının ortalamaya sapması %2 olduğu kaydedilmiştir.

Node.js axios Çalışması

Bu çalışma JavaScript programlama dilinde Node.js v18 sürümünde axios kütüphanesi kullanılarak Visual Studio Code ortamında gerçeklenmiş ve Tablo 4.36'daki ve Şekil 4.106-Şekil 4.108 arasındaki sonuçlar elde edilmiştir. Program koduna EK 2.7'de yer verilmiştir.

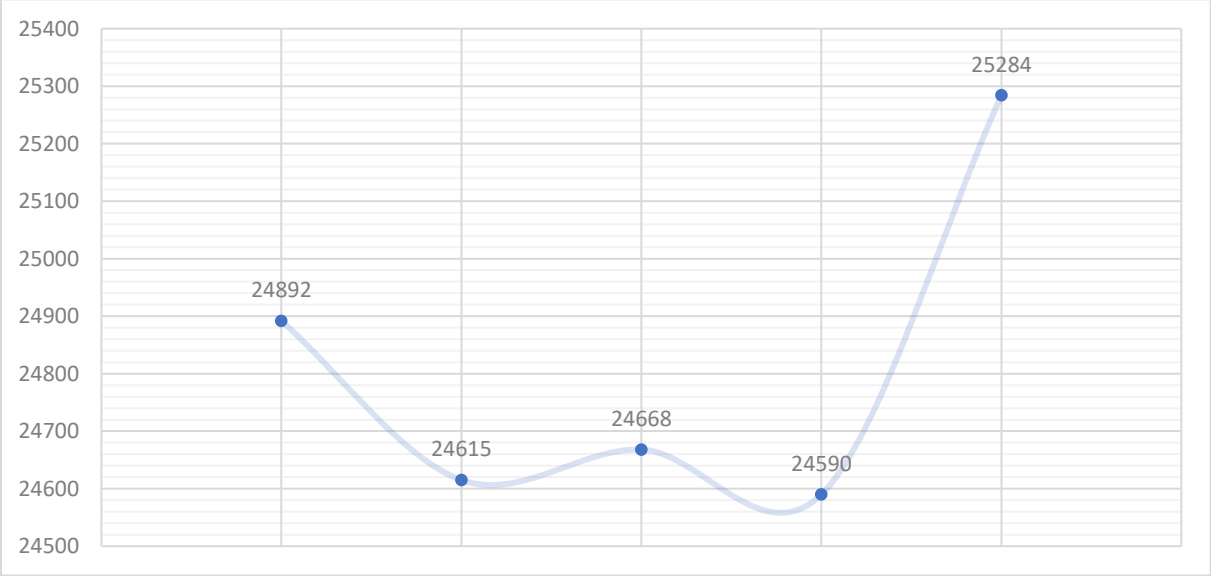
Tablo 4.36 REST JSON Büyük Veri Node.js axios

Tekrar	LOCAL	S4	IDES
1	992	24892	26632
2	964	24615	26536
3	961	24668	26688
4	971	24590	26620
5	967	25284	26350
Ortalama	**971**	**24810**	**26565**



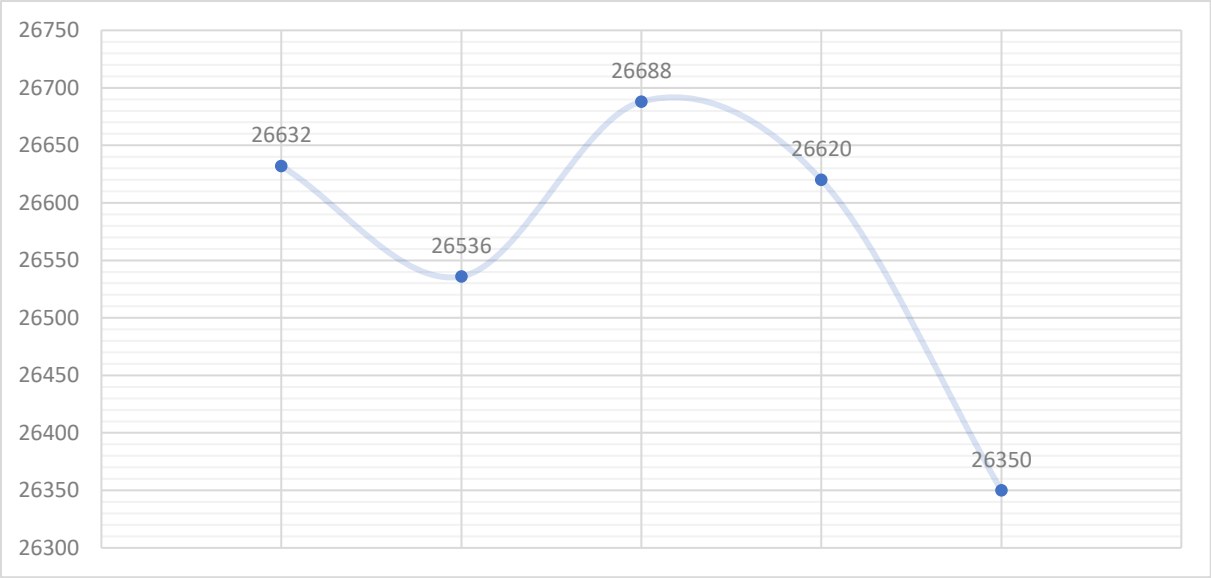
Şekil 4.21 LOCAL REST JSON Büyük Veri Node.js axios

Test ortalaması 971 ms olup, açıklığın ortalamaya sapması %3,2 ve çeyrekler açıklığının ortalamaya sapması %0,7 olduğu kaydedilmiştir.



Şekil 4.107 S4 REST JSON Büyük Veri Node.js axios

Test ortalaması 24810 ms olup, açıklığın ortalamaya sapması %2,8 ve çeyrekler açıklığının ortalamaya sapması %1,1 olduğu kaydedilmiştir.



Şekil 4.108 IDES REST JSON Büyük Veri Node.js axios

Test ortalaması 26565 ms olup, açıklığın ortalamaya sapması %1,3 ve çeyrekler açıklığının ortalamaya sapması %0,4 olduğu kaydedilmiştir.

4.1.6. Büyük Veriyle REST XML Uygulamaları

Bu tez çalışmasında olarak XML formatındaki büyük verilerle REST API çalışılmıştır. Küçük verilerle çalışmalara göre 100-10.000 kat daha büyük ve her sunucuda aynı verilerle çalışılmış böylelikle veri boyutlarının SAP tarafında, API üzerinde, dosya formatı boyutlarında ve istemci tarafında yorumlanmasındaki performans farkları daha net ortaya koyulmuştur. Üç farklı programlama dili kullanılıp bunlar sırasıyla C#, Java ve JavaScript(Node.js) dilleri olmuştur. Bu programlama dillerinin de kendi içlerinde farklı kütüphaneler ve sınıflar kullanılarak test sonuçları her programlama dili içerisinde çeşitlendirilmiştir. Her bir program uygun olduğu geliştirme ortamında geliştirilip derlenmiş ve çalıştırılmıştır. Her bir çalıştırma döngüsü kendi içerisinde üç ana başlıktan oluşmaktadır; istek paketlerinin oluşturulması, gönderilmesi ve cevabın alınması, cevabın yorumlanması. Bu üç başlığın toplam çalışma süresi milisaniye(ms) biriminden kaydedilmiş ve tez çalışması kapsamında yorumlanmıştır. Bu tez çalışması kapsamında yorumlanan veriler; programların toplam çalışma süreleri değil, belirtilen bu üç ana başlıkta geçirilen sürelerdir.

SAP sunucusunun REST API ile XML formatında döndürdüğü cevaba EK 3.1'de yer verilmiştir.

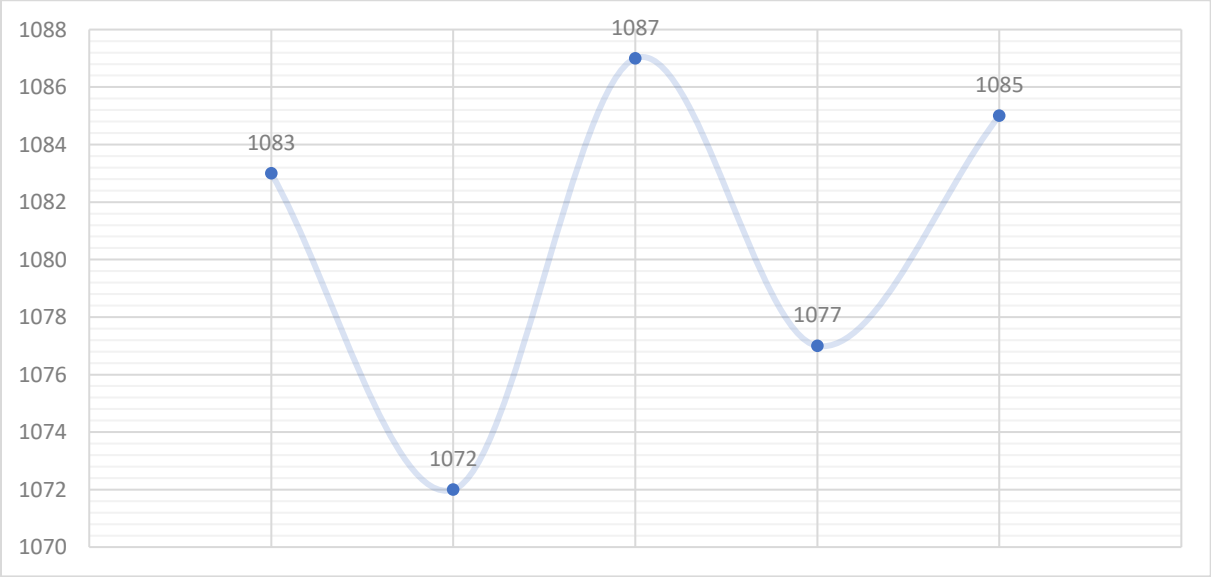
Kullanılan farklı diller ve bu dillerde kullanılan farklı kütüphaneler veya sınıflar REST isteklerini farklı yöntemlerle oluşturup, gönderip, EK 3.1'de yer verilen aynı cevabı almıştır. Bu cevap; SAP tarafında veritabanının farklı alanlarından SQL sorguları ve ABAP fonksiyonlarıyla çekilmiş, paketlenmiş ve gönderilmiştir.

C# HttpClient Çalışması

Bu çalışma C# programlama dilinde .NET 7.0 platformunda HttpClient sınıfı kullanılarak Visual Studio ortamında gerçeklenmiş ve Tablo 4.37'deki ve Şekil 4.109-Şekil 4.111 arasındaki sonuçlar elde edilmiştir. Program koduna EK 3.2'de yer verilmiştir.

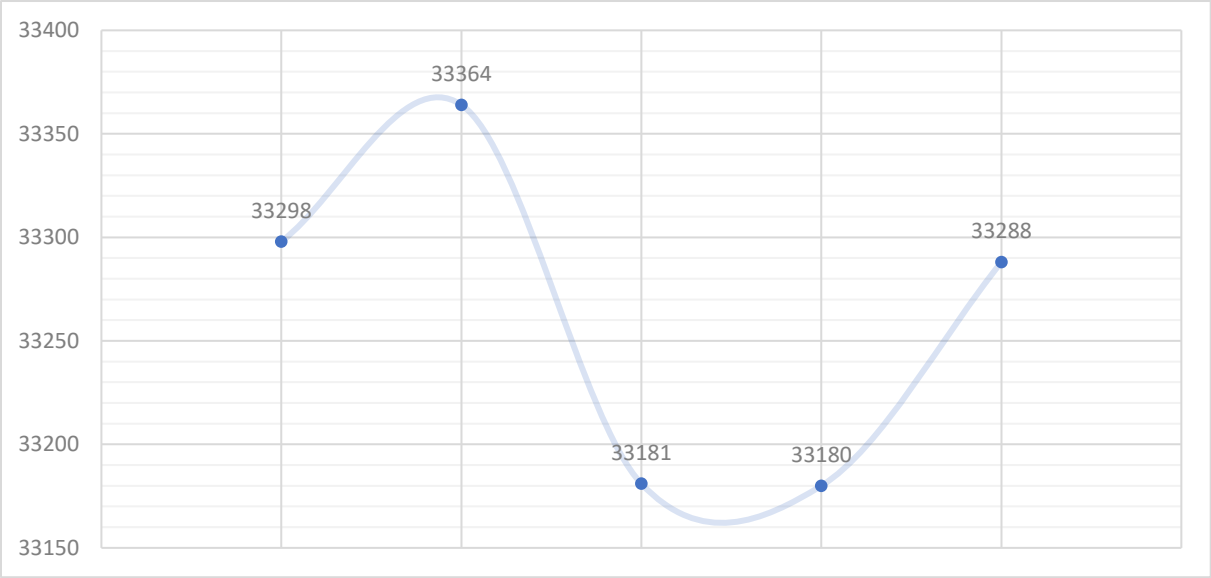
Tablo 4.37 REST XML Büyük Veri C# HttpClient

Tekrar	LOCAL	S4	IDES
1	1083	33298	35658
2	1072	33364	35575
3	1087	33181	35443
4	1077	33180	35423
5	1085	33288	35329
Ortalama	**1081**	**33262**	**35486**



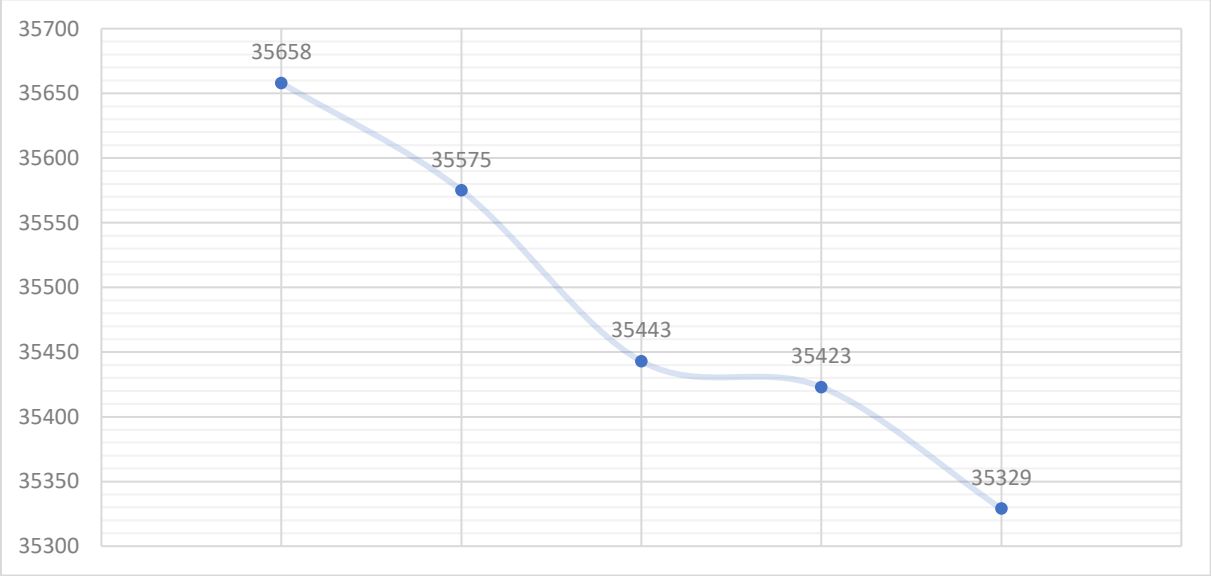
Şekil 4.22 LOCAL REST XML Büyük Veri C# HttpClient

Test ortalaması 1081 ms olup, açıklığın ortalamaya sapması %1,4 ve çeyrekler açıklığının ortalamaya sapması %0,7 olduğu kaydedilmiştir.



Şekil 4.110 S4 REST XML Büyük Veri C# HttpClient

Test ortalaması 33262 ms olup, açıklığın ortalamaya sapması %0,6 ve çeyrekler açıklığının ortalamaya sapması %0,4 olduğu kaydedilmiştir.



Şekil 4.111 IDES REST XML Büyük Veri C# HttpClient

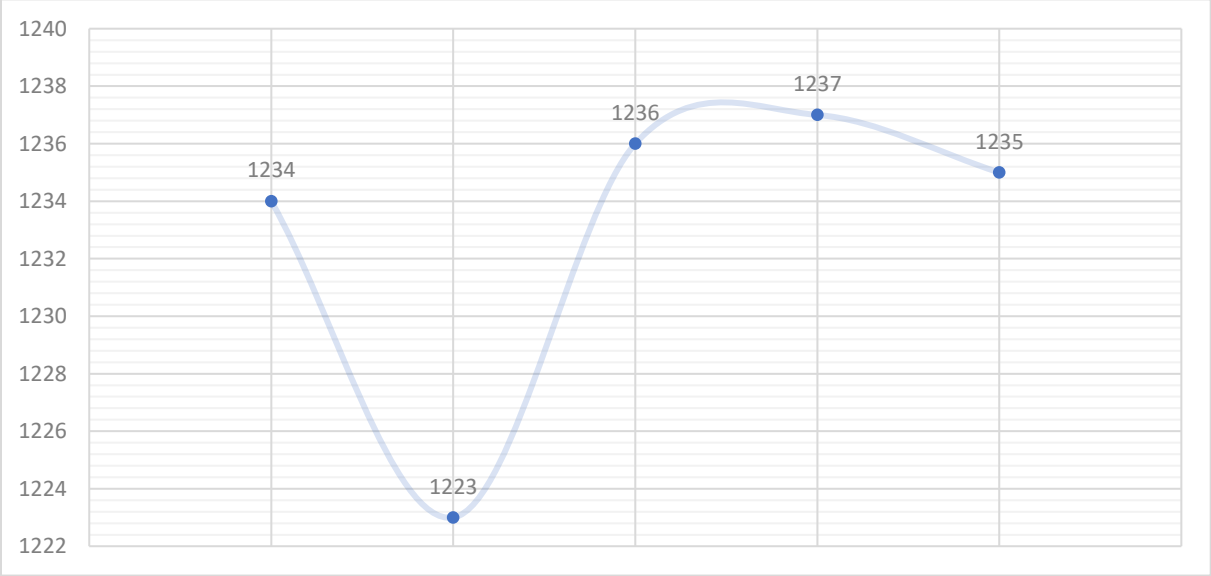
Test ortalaması 35486 ms olup, açıklığın ortalamaya sapması %0,9 ve çeyrekler açıklığının ortalamaya sapması %0,4 olduğu kaydedilmiştir.

C# RestSharp Çalışması

Bu çalışma C# programlama dilinde .NET 7.0 platformunda RestSharp sınıfı kullanılarak Visual Studio ortamında gerçeklenmiş ve Tablo 4.38'deki ve Şekil 4.111-Şekil 4.113 arasındaki sonuçlar elde edilmiştir. Program koduna EK 3.3'te yer verilmiştir.

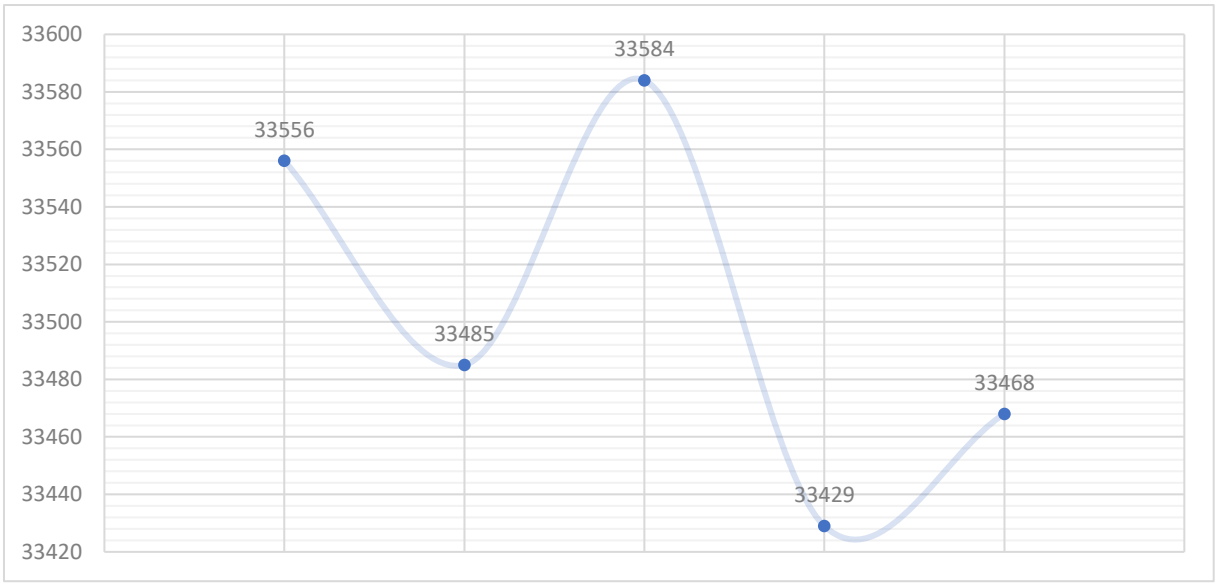
Tablo 4.38 REST XML Büyük Veri C# RestSharp

Tekrar	LOCAL	S4	IDES
1	1234	33556	35740
2	1223	33485	35551
3	1236	33584	35572
4	1237	33429	35678
5	1235	33468	35561
Ortalama	**1233**	**33504**	**35620**



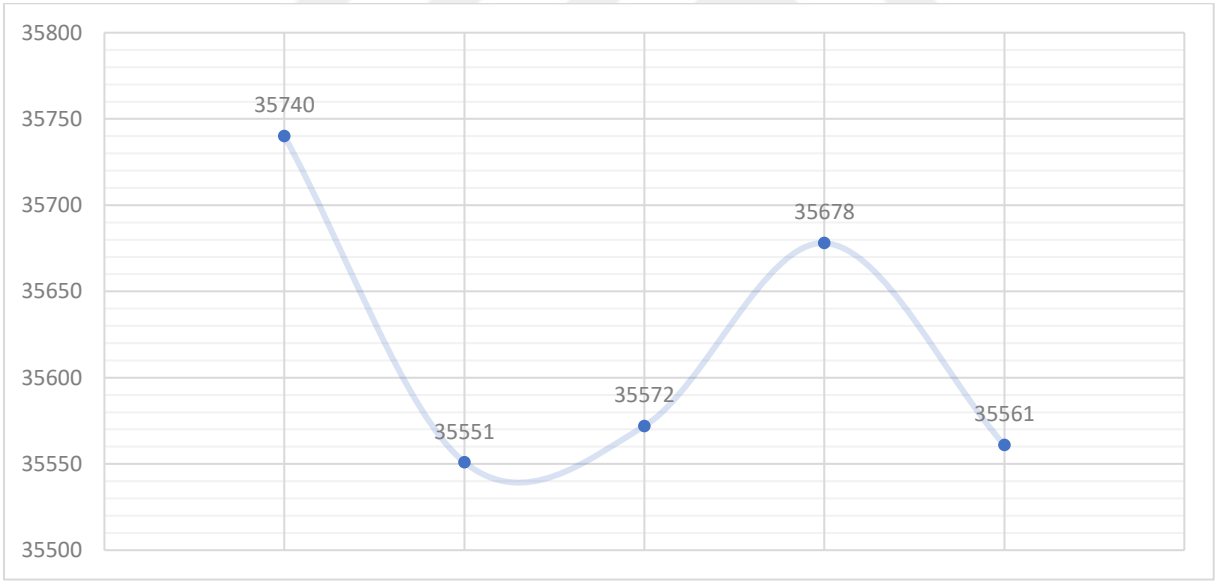
Şekil 4.23 LOCAL REST XML Büyük Veri C# RestSharp

Test ortalaması 1233 ms olup, açıklığın ortalamaya sapması %1,1 ve çeyrekler açıklığının ortalamaya sapması %0,2 olduğu kaydedilmiştir.



Şekil 4.113 S4 REST XML Büyük Veri C# RestSharp

Test ortalaması 33504 ms olup, açıklığın ortalamaya sapması %0,5 ve çeyrekler açıklığının ortalamaya sapması %0,3 olduğu kaydedilmiştir.



Şekil 4.114 IDES REST XML Büyük Veri C# RestSharp

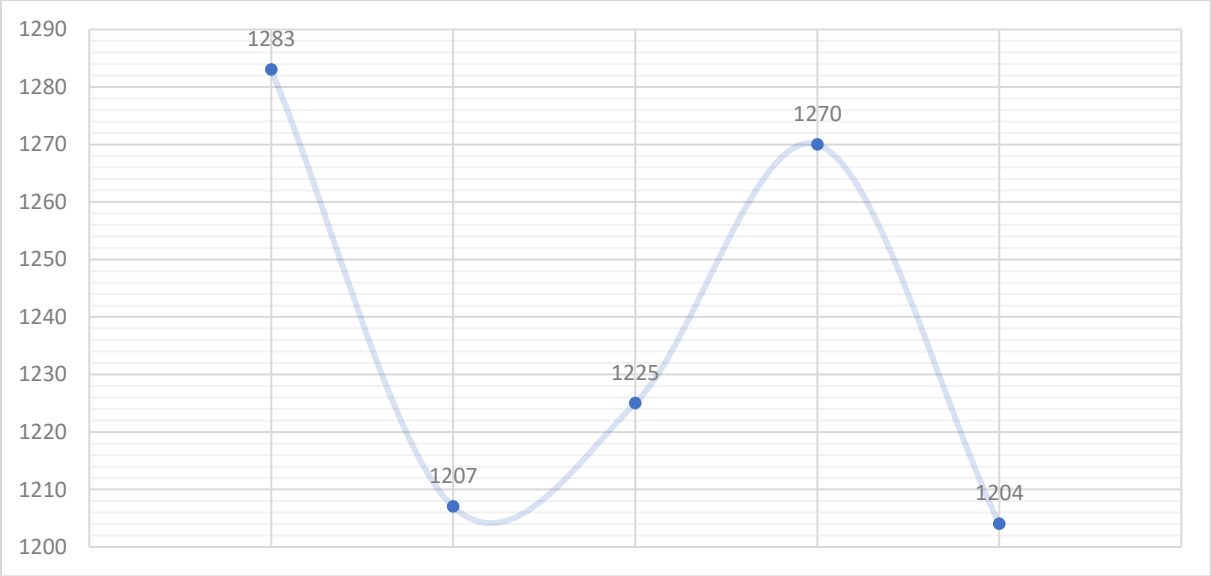
Test ortalaması 35620 ms olup, açıklığın ortalamaya sapması %0,5 ve çeyrekler açıklığının ortalamaya sapması %0,3 olduğu kaydedilmiştir.

Java HttpURLConnection Çalışması

Bu çalışma Java programlama dilinde JDK 19 sürümünde HttpURLConnection sınıfı kullanılarak IntelliJ IDEA ortamında gerçeklenmiş ve Tablo 4.39'daki ve Şekil 4.114-Şekil 4.116 arasındaki sonuçlar elde edilmiştir. Program koduna EK 3.4'te yer verilmiştir.

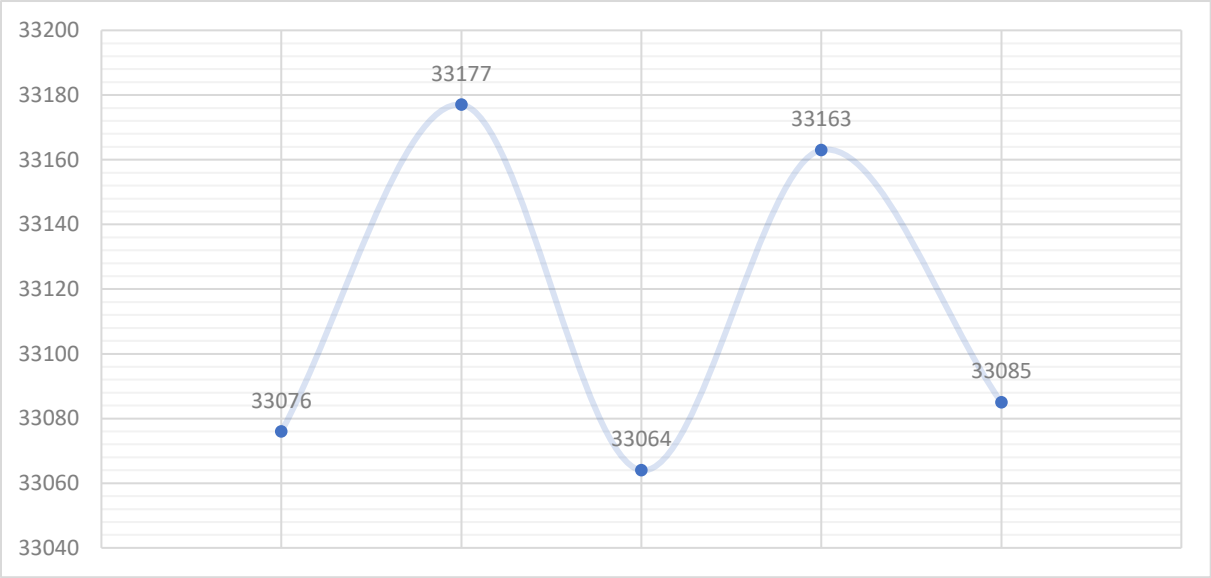
Tablo 4.39 REST XML Büyük Veri Java HttpURLConnection

Tekrar	LOCAL	S4	IDES
1	1283	33076	35317
2	1207	33177	35259
3	1225	33064	35294
4	1270	33163	35174
5	1204	33085	35497
Ortalama	**1238**	**33113**	**35308**



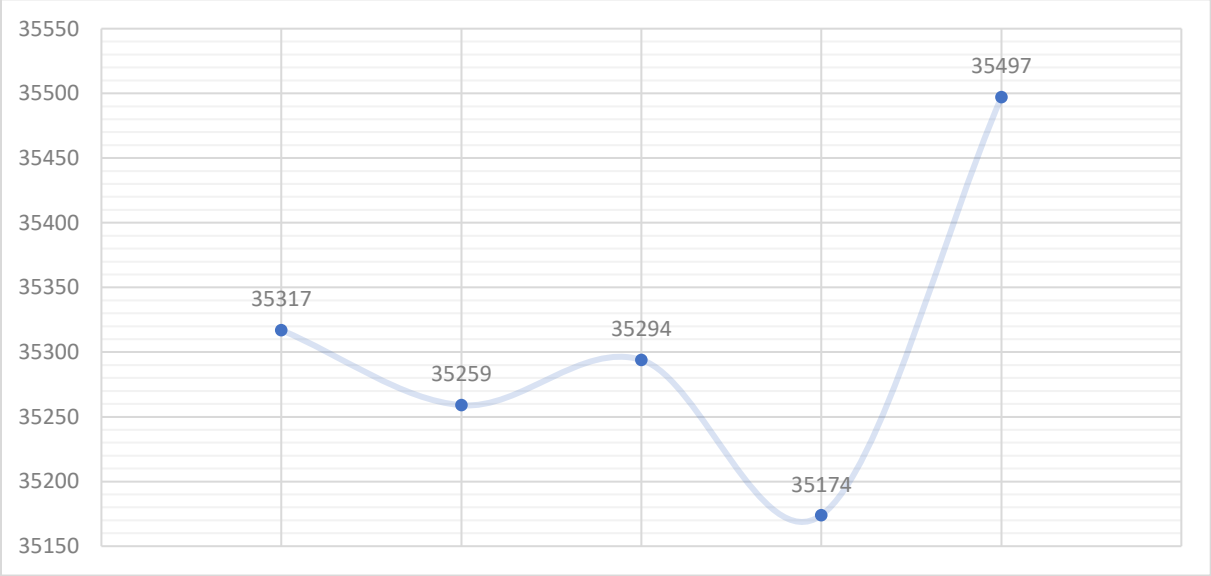
Şekil 4.24 LOCAL REST XML Büyük Veri Java HttpURLConnection

Test ortalaması 1238 ms olup, açıklığın ortalamaya sapması %6,4 ve çeyrekler açıklığının ortalamaya sapması %5,1 olduğu kaydedilmiştir.



Şekil 4.116 S4 REST XML Büyük Veri Java HttpURLConnection

Test ortalaması 33113 ms olup, açıklığın ortalamaya sapması %0,3 ve çeyrekler açıklığının ortalamaya sapması %0,3 olduğu kaydedilmiştir.



Şekil 4.117 IDES REST XML Büyük Veri Java HttpURLConnection

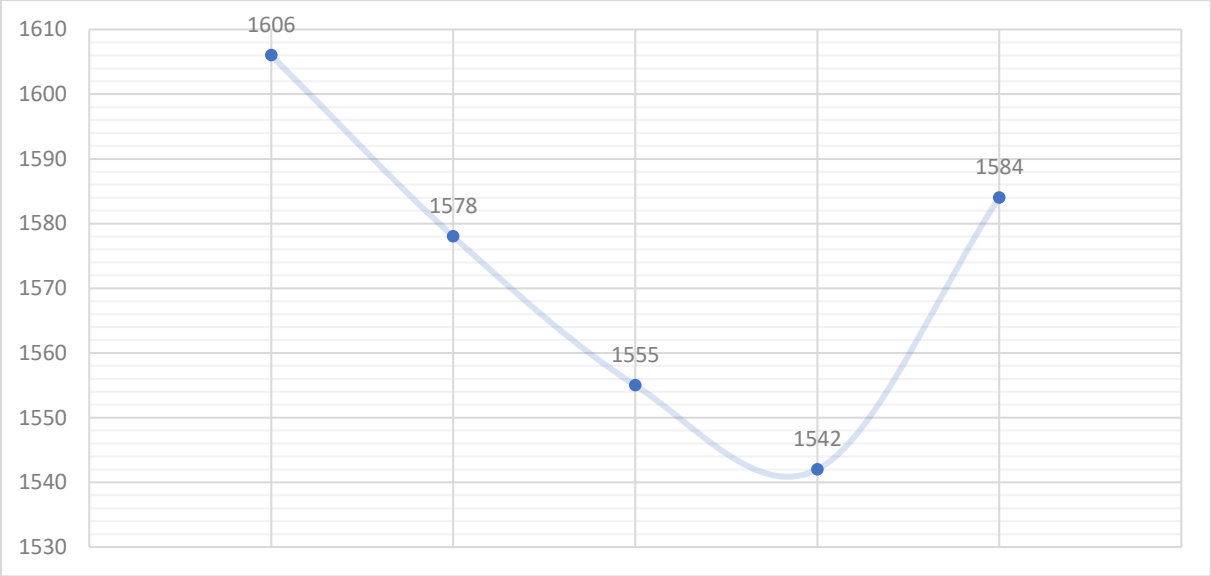
Test ortalaması 35308 ms olup, açıklığın ortalamaya sapması %0,9 ve çeyrekler açıklığının ortalamaya sapması %0,2 olduğu kaydedilmiştir.

Java HttpClient Çalışması

Bu çalışma Java programlama dilinde JDK 19 sürümünde HttpClient sınıfı kullanılarak IntelliJ IDEA ortamında gerçeklenmiş ve Tablo 4.40'teki ve Şekil 4.118-Şekil 4.120 arasındaki sonuçlar elde edilmiştir. Program koduna EK 3.5'te yer verilmiştir.

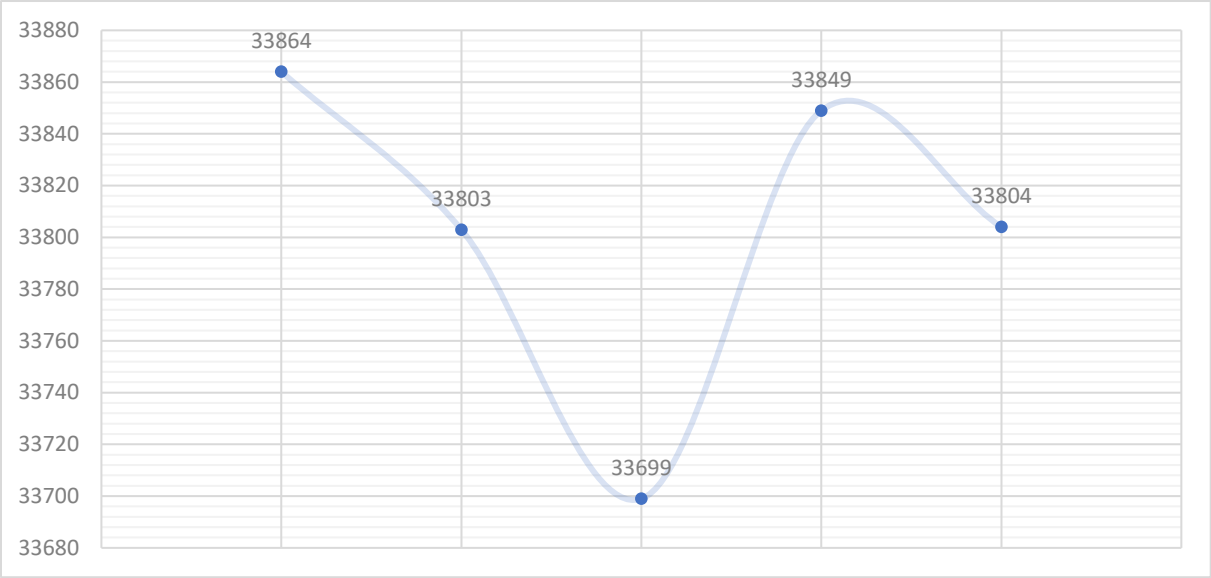
Tablo 4.40 REST XML Büyük Veri Java HttpClient

Tekrar	LOCAL	S4	IDES
1	1606	33864	36079
2	1578	33803	35945
3	1555	33699	35925
4	1542	33849	35905
5	1584	33804	36026
Ortalama	**1573**	**33804**	**35976**



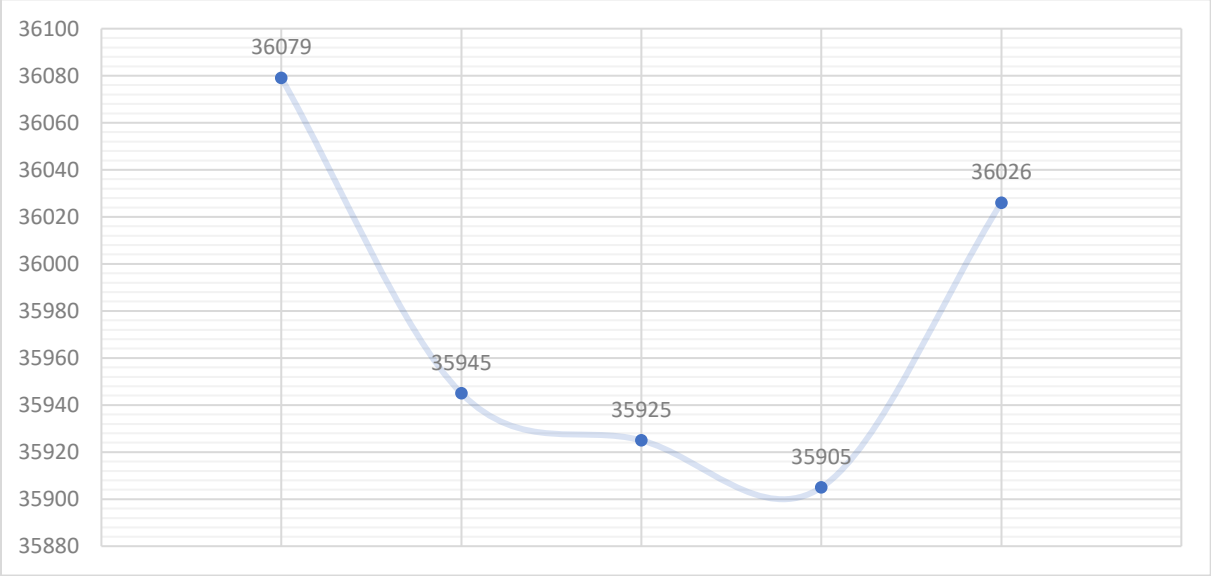
Şekil 4.25 LOCAL REST XML Büyük Veri Java HttpClient

Test ortalaması 1573 ms olup, açıklığın ortalamaya sapması %4,1 ve çeyrekler açıklığının ortalamaya sapması %1,8 olduğu kaydedilmiştir.



Şekil 4.119 S4 REST XML Büyük Veri Java HttpClient

Test ortalaması 33804 ms olup, açıklığın ortalamaya sapması %0,5 ve çeyrekler açıklığının ortalamaya sapması %0,1 olduğu kaydedilmiştir.



Şekil 4.120 IDES REST XML Büyük Veri Java HttpClient

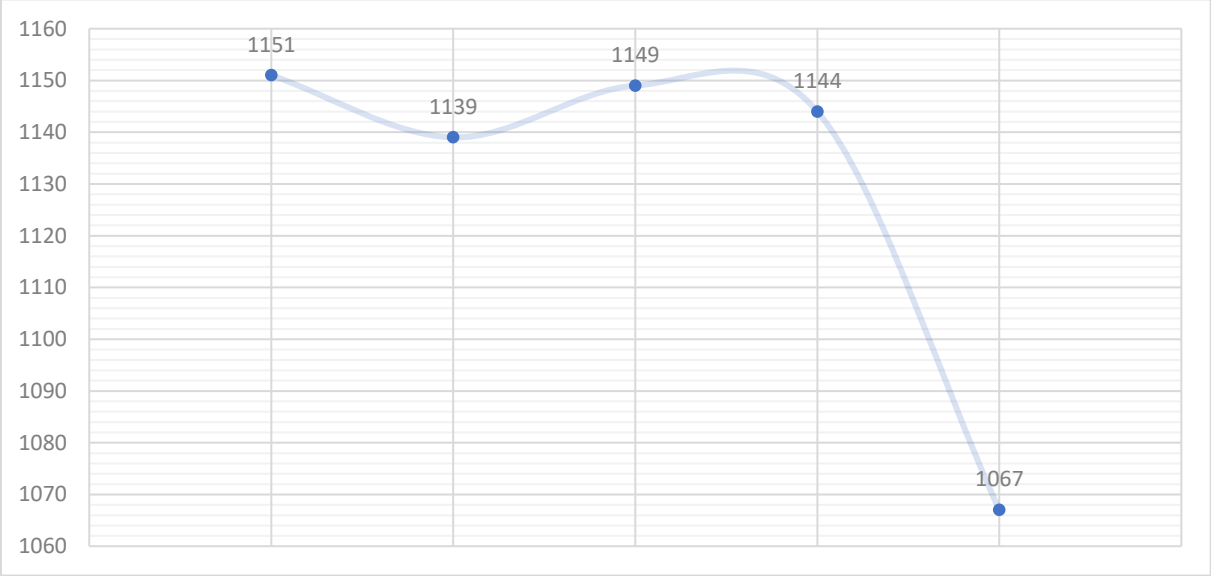
Test ortalaması 35976 ms olup, açıklığın ortalamaya sapması %0,5 ve çeyrekler açıklığının ortalamaya sapması %0,3 olduğu kaydedilmiştir.

Node.js request Çalışması

Bu çalışma JavaScript programlama dilinde Node.js v18 sürümünde request kütüphanesi kullanılarak Visual Studio Code ortamında gerçeklenmiş ve Tablo 4.41'deki ve Şekil 4.121-Şekil 4.123 arasındaki sonuçlar elde edilmiştir. Program koduna EK 3.6'da yer verilmiştir.

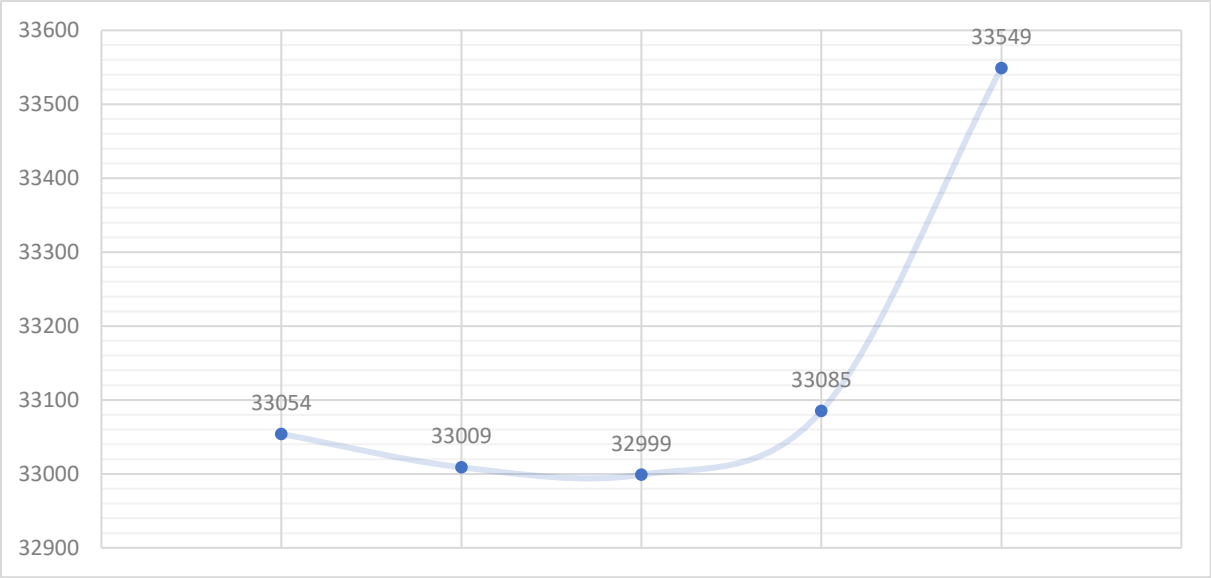
Tablo 4.41 REST XML Büyük Veri Node.js request

Tekrar	LOCAL	S4	IDES
1	1151	33054	35262
2	1139	33009	35289
3	1149	32999	35258
4	1144	33085	35403
5	1067	33549	35181
Ortalama	**1130**	**33139**	**35279**



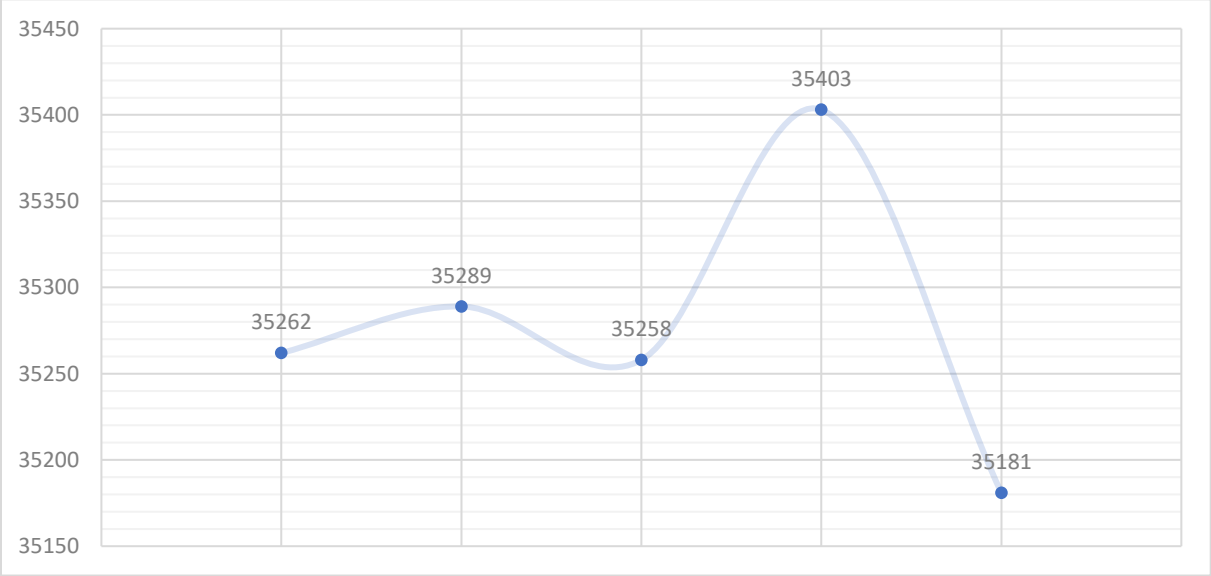
Şekil 4.26 LOCAL REST XML Büyük Veri Node.js request

Test ortalaması 1130 ms olup, açıklığın ortalamaya sapması %7,4 ve çeyrekler açıklığının ortalamaya sapması %0,9 olduğu kaydedilmiştir.



Şekil 4.122 S4 REST XML Büyük Veri Node.js request

Test ortalaması 33139 ms olup, açıklığın ortalamaya sapması %1,7 ve çeyrekler açıklığının ortalamaya sapması %0,2 olduğu kaydedilmiştir.



Şekil 4.123 IDES REST XML Büyük Veri Node.js request

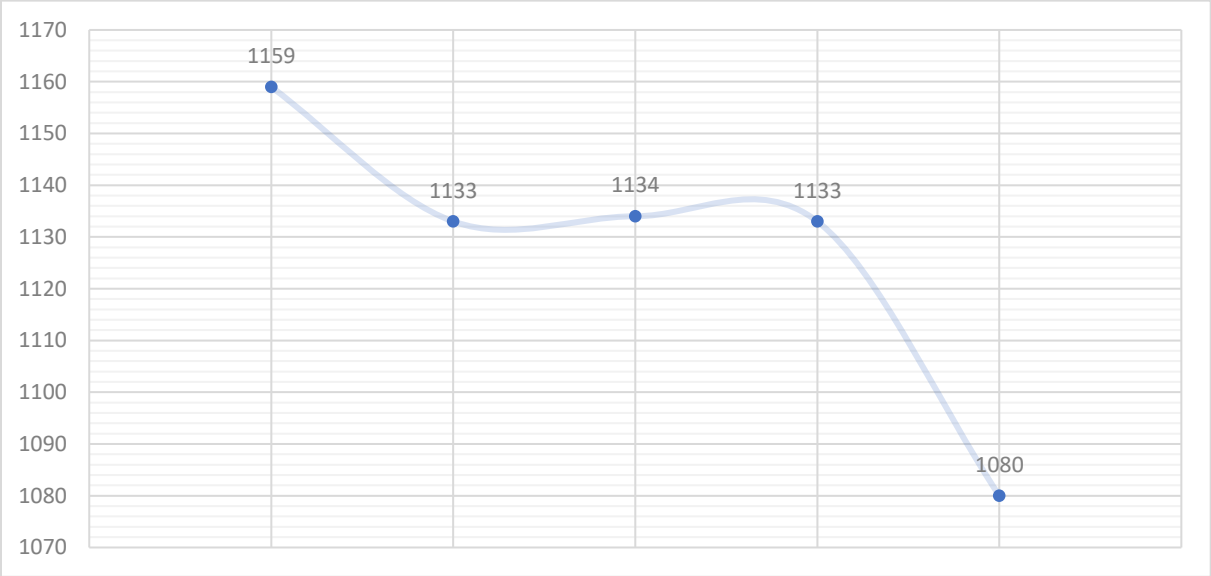
Test ortalaması 35279 ms olup, açıklığın ortalamaya sapması %0,6 ve çeyrekler açıklığının ortalamaya sapması %0,1 olduğu kaydedilmiştir.

Node.js axios Çalışması

Bu çalışma JavaScript programlama dilinde Node.js v18 sürümünde axios kütüphanesi kullanılarak Visual Studio Code ortamında gerçeklenmiş ve Tablo 4.42'deki ve Şekil 4.124-Şekil 4.126 arasındaki sonuçlar elde edilmiştir. Program koduna EK 3.7'de yer verilmiştir.

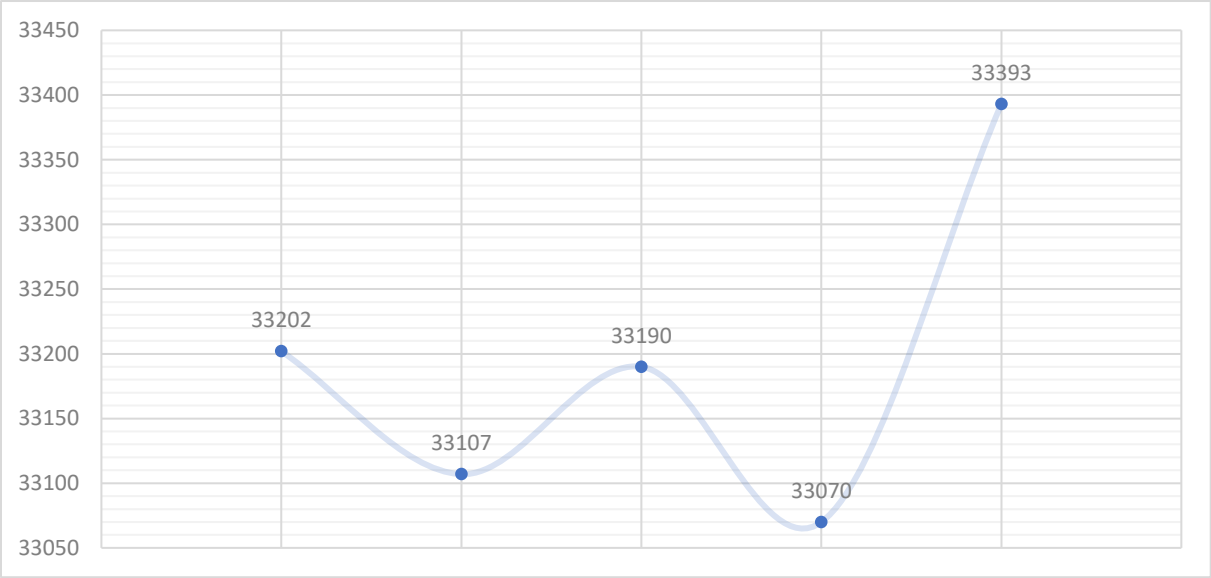
Tablo 4.42 REST XML Büyük Veri Node.js axios

Tekrar	LOCAL	S4	IDES
1	1159	33202	35374
2	1133	33107	35404
3	1134	33190	35363
4	1133	33070	35657
5	1080	33393	35299
Ortalama	**1128**	**33192**	**35419**



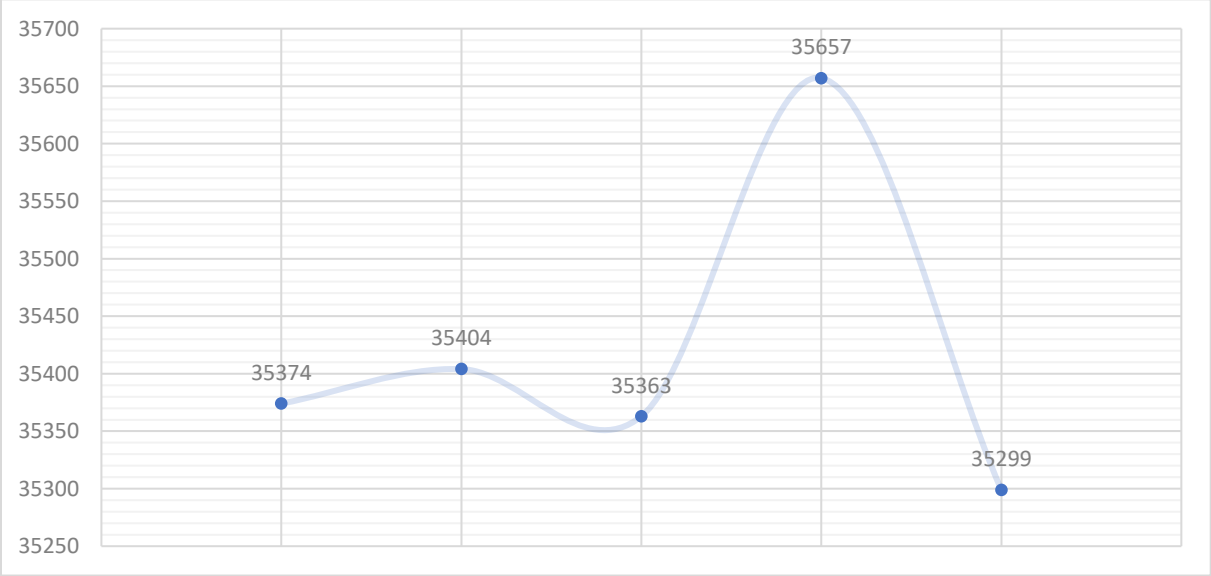
Şekil 4.27 LOCAL REST XML Büyük Veri Node.js axios

Test ortalaması 1128 ms olup, açıklığın ortalamaya sapması %7 ve çeyrekler açıklığının ortalamaya sapması %0,1 olduğu kaydedilmiştir.



Şekil 4.1228 S4 REST XML Büyük Veri Node.js axios

Test ortalaması 33192 ms olup, açıklığın ortalamaya sapması %1 ve çeyrekler açıklığının ortalamaya sapması %0,3 olduğu kaydedilmiştir.



Şekil 4.126 IDES REST XML Büyük Veri Node.js axios

Test ortalaması 35419 ms olup, açıklığın ortalamaya sapması %1 ve çeyrekler açıklığının ortalamaya sapması %0,1 olduğu kaydedilmiştir.

4.2. PERFORMANS KARŞILAŞTIRMALARI

4.2.1. Küçük Veri SOAP Çalışma Analizi

Tablo 4.43 LOCAL Tüm küçük veri SOAP cevap süreleri (ms)

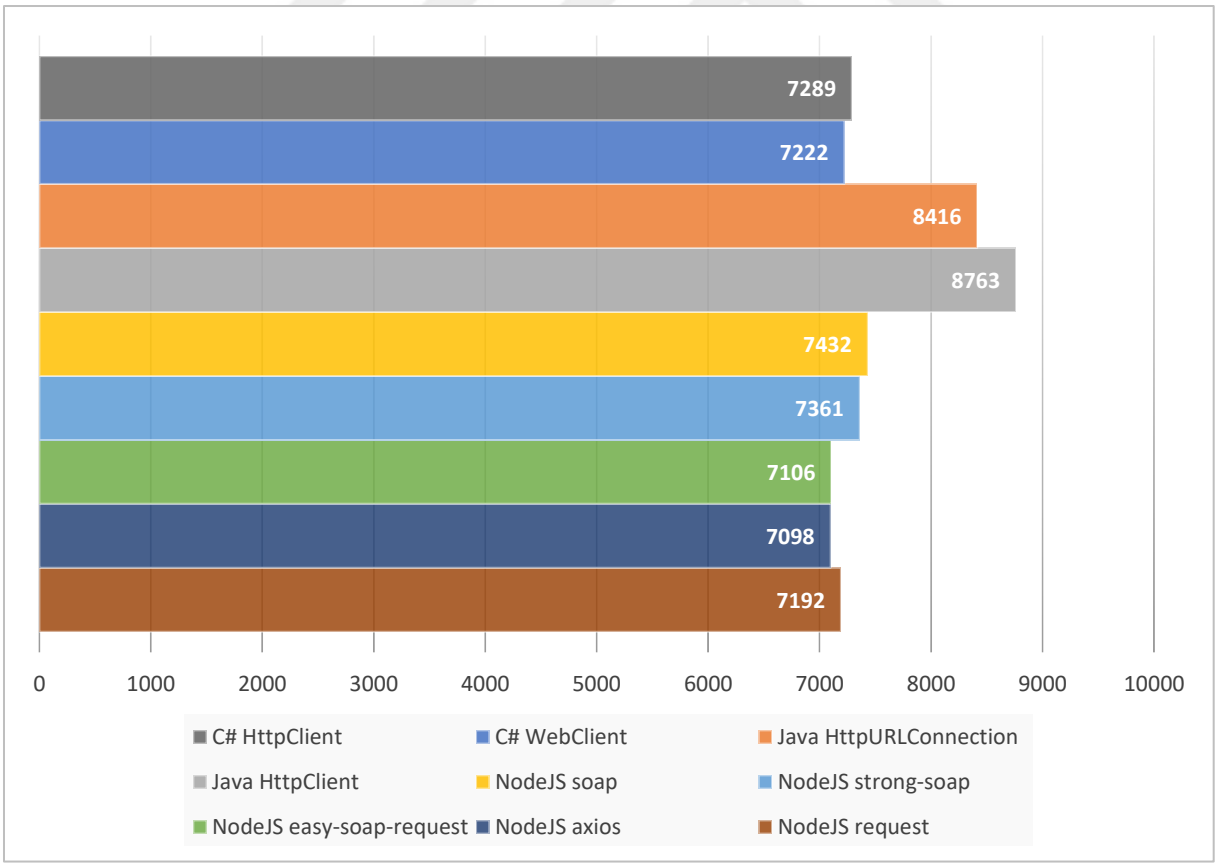
Tekrar	C# HttpClient	C# WebClient	Java HttpURLConnection	Java HttpClient	Node.js soap	Node.js strong-soap	Node.js easy-soap-request	Node.js axios	Node.js request
1	7365	7132	8490	8739	7529	7412	7052	6954	7266
2	7290	7206	8432	8757	7422	7442	7165	7006	7220
3	7268	7215	8490	8753	7492	7445	6942	7050	7112
4	7241	7194	8519	8744	7425	7456	7093	7073	7051
5	7284	7225	8362	8742	7480	7612	7214	7005	7164
6	7344	7205	8523	8799	7366	7254	7176	7235	7213
7	7268	7245	8406	8806	7437	7362	7192	7195	7156
8	7225	7334	8228	8725	7392	7197	7207	7149	7128
9	7268	7245	8370	8738	7368	7300	7120	7082	7174
10	7338	7222	8338	8823	7411	7126	6899	7234	7438
ORTALAMA	**7289**	**7222**	**8416**	**8763**	**7432**	**7361**	**7106**	**7098**	**7192**
AÇIKLIK/ORT. (%)	**1,9**	**2,8**	**3,5**	**1,1**	**2,2**	**6,6**	**4,4**	**4,0**	**5,4**
ÇEY.AÇIK./ORT. (%)	**0,8**	**0,5**	**1,5**	**0,6**	**1,0**	**2,4**	**1,8**	**2,3**	**1,2**

Tablo 4.44 S4 Tüm küçük veri SOAP cevap süreleri (ms)

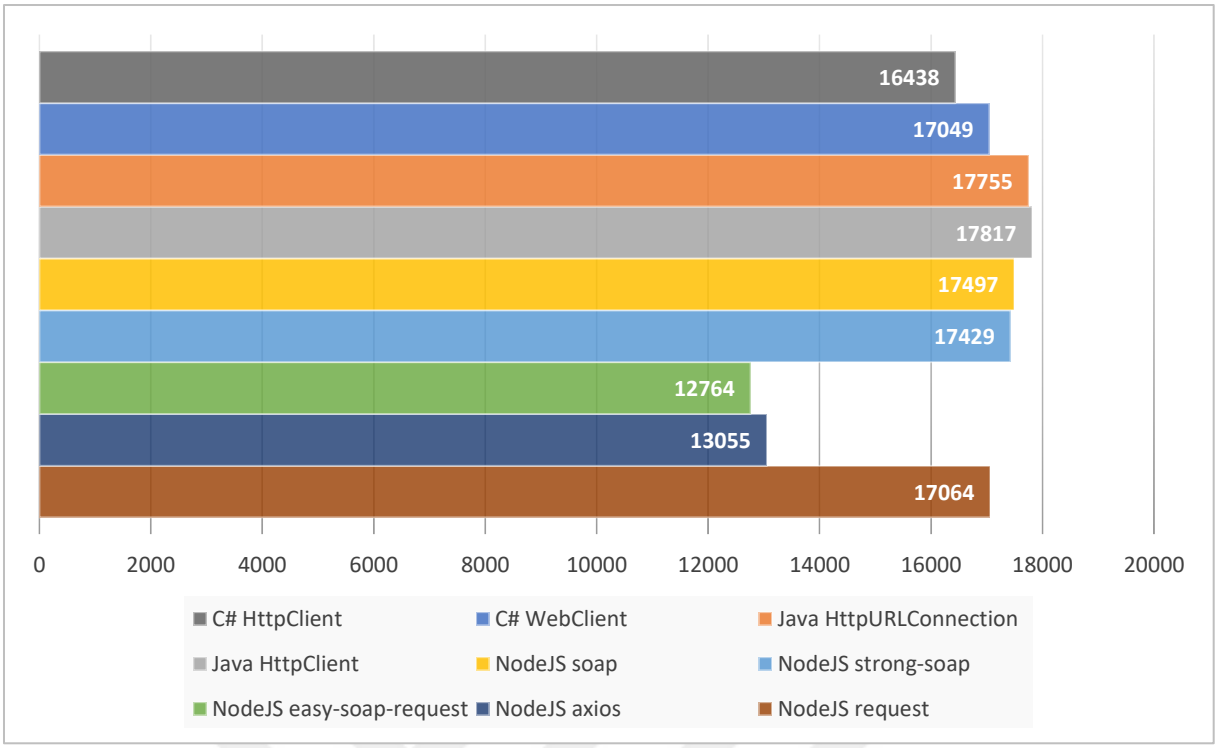
Tekrar	C# HttpClient	C# WebClient	Java HttpURLConnection	Java HttpClient	Node.js soap	Node.js strong-soap	Node.js easy-soap-request	Node.js axios	Node.js request
1	16022	16584	17358	18113	17866	17497	12955	13387	17311
2	16560	16284	17540	17534	17805	17276	12739	12903	17060
3	16488	16941	17648	17204	16844	17505	12823	13027	16496
4	16254	17689	17662	18471	17007	17327	12505	12949	16890
5	16655	16579	17757	17613	17658	17175	12203	13288	17167
6	16360	17468	18684	18147	17356	17844	12563	13212	17280
7	16233	17686	17307	17703	17345	17683	12886	12755	17118
8	16776	17103	18203	17972	17408	17472	12814	13205	17323
9	16498	16870	17741	17789	18208	17200	12788	12946	17193
10	16538	17288	17646	17625	17475	17310	13363	12879	16797
ORTALAMA	**16438**	**17049**	**17755**	**17817**	**17497**	**17429**	**12764**	**13055**	**17064**
AÇIKLIK/ORT. (%)	**4,6**	**8,2**	**7,8**	**7,1**	**7,8**	**3,8**	**9,1**	**4,8**	**4,8**
ÇEY.AÇIK./ORT. (%)	**1,7**	**4,5**	**1,1**	**2,6**	**2,4**	**1,3**	**2,1**	**2,3**	**1,9**

Tablo 4.45 IDES Tüm küçük veri SOAP cevap süreleri (ms)

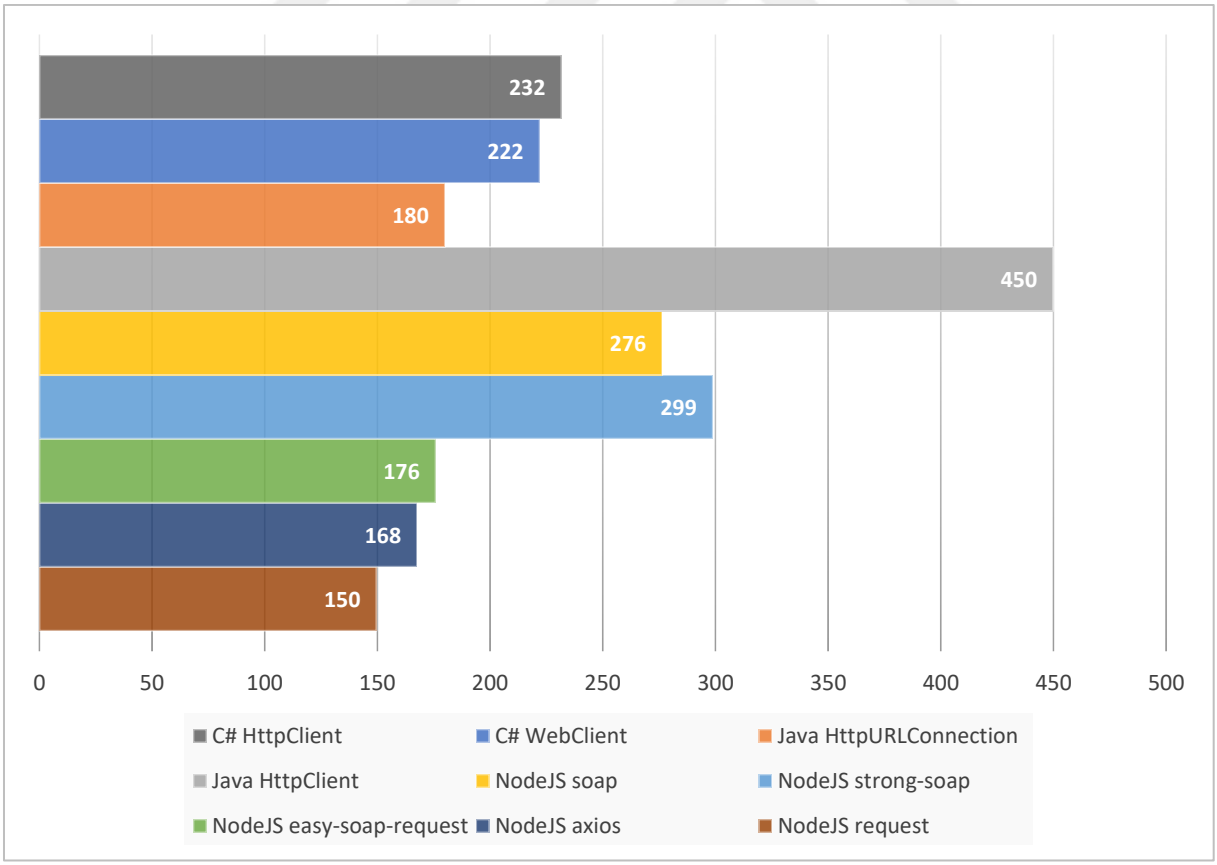
Tekrar	C# HttpClient	C# WebClient	Java HttpURLConnection	Java HttpClient	Node.js soap	Node.js strong-soap	Node.js easy-soap-request	Node.js axios	Node.js request
1	238	216	201	423	293	297	166	178	148
2	223	228	167	428	256	285	164	154	162
3	228	207	174	437	270	288	170	155	139
4	227	227	176	424	282	282	156	146	152
5	225	197	200	504	269	315	153	144	142
6	245	222	165	461	274	292	167	167	154
7	222	243	176	485	279	293	183	161	151
8	246	206	187	452	282	292	198	171	150
9	256	206	184	444	287	320	200	182	145
10	207	268	169	440	272	324	202	218	153
ORTALAMA	**232**	**222**	**180**	**450**	**276**	**299**	**176**	**168**	**150**
AÇIKLIK/ORT. (%)	**21,1**	**32,0**	**20,0**	**18,0**	**13,4**	**14,1**	**27,9**	**44,2**	**15,4**
ÇEY.AÇIK./ORT. (%)	**8,5**	**9,7**	**8,9**	**6,3**	**4,2**	**7,2**	**16,9**	**13,1**	**4,7**



Şekil 4.129 LOCAL küçük veri SOAP ortalama cevap süreleri (ms)



Şekil 4.128 S4 küçük veri SOAP ortalama cevap süreleri (ms)



Şekil 4.130 IDES küçük veri SOAP ortalama cevap süreleri (ms)

4.2.2. Küçük Veri REST JSON Çalışma Analizi

Tablo 4.46 LOCAL Tüm küçük veri REST JSON cevap süreleri (ms)

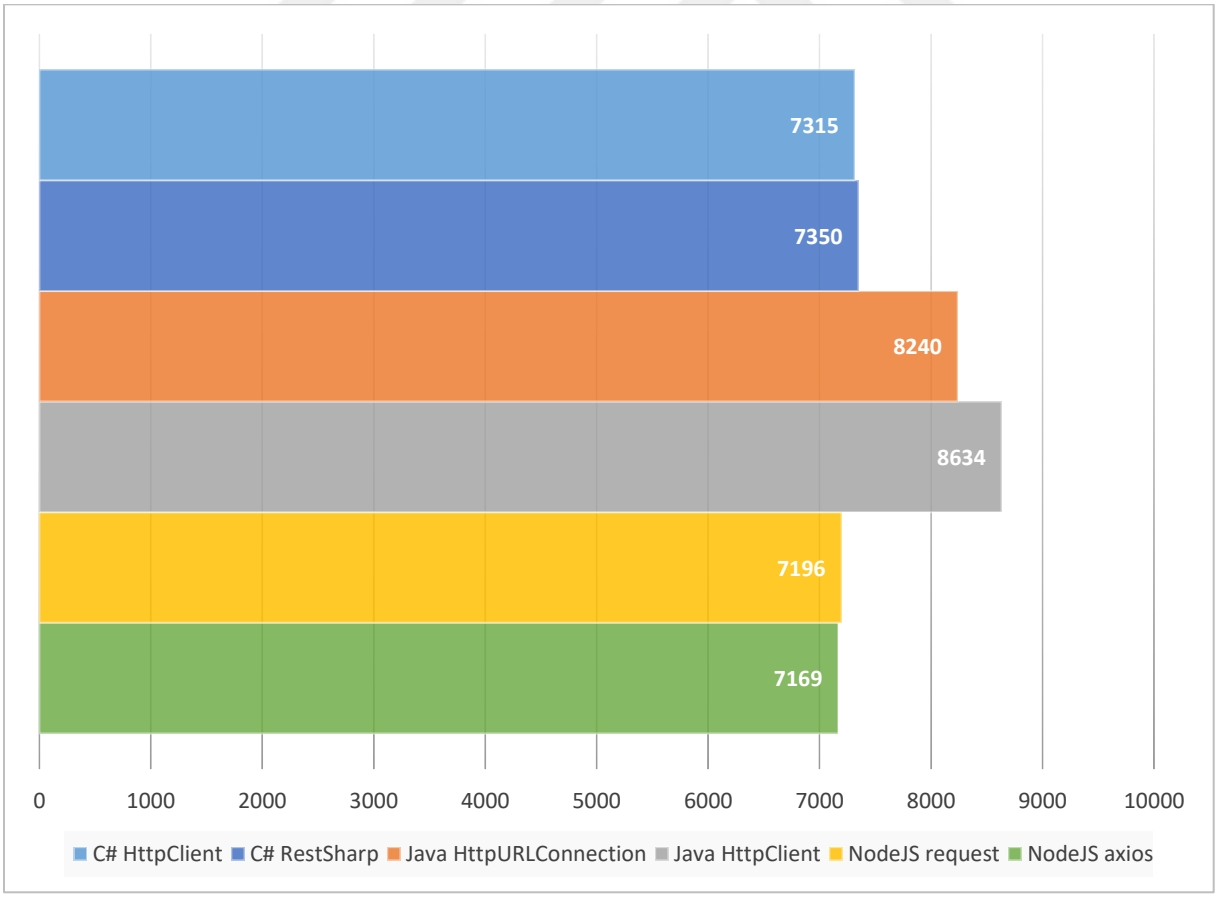
Tekrar	C# HttpClient	C# RestSharp	Java HttpURLConnection	Java HttpClient	Node.js request	Node.js axios
1	7219	7383	8085	8441	7261	7211
2	7309	7292	8053	8522	7149	7200
3	7366	7359	8095	8357	7426	7151
4	7433	7350	8154	9366	7401	7146
5	7257	7344	8397	8929	7060	7172
6	7340	7331	8317	8575	7116	7124
7	7266	7431	8238	8638	7071	7181
8	7228	7288	8306	8572	7114	7186
9	7418	7416	8376	8381	7134	7098
10	7311	7303	8382	8562	7231	7218
ORTALAMA	**7315**	**7350**	**8240**	**8634**	**7196**	**7169**
AÇIKLIK/ORT. (%)	**2,9**	**1,9**	**4,2**	**11,7**	**5,1**	**1,7**
ÇEY.AÇIK./ORT. (%)	**1,4**	**0,9**	**3,1**	**1,9**	**1,9**	**0,7**

Tablo 4.47 S4 Tüm küçük veri REST JSON cevap süreleri (ms)

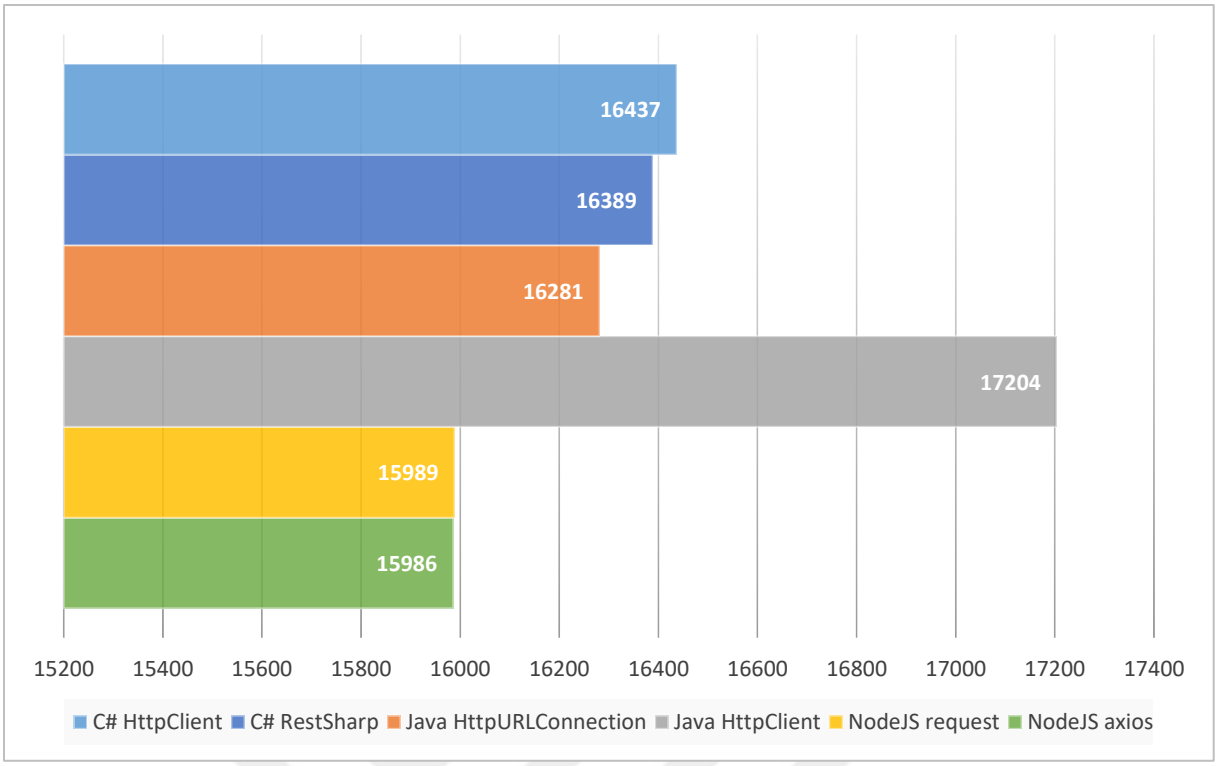
Tekrar	C# HttpClient	C# RestSharp	Java HttpURLConnection	Java HttpClient	Node.js request	Node.js axios
1	16511	16188	16960	16790	16060	15687
2	16888	16516	16248	17665	16174	16021
3	16449	16195	16706	17173	16311	16072
4	16638	16828	16099	17508	16071	15939
5	16065	16317	16034	16889	15754	15782
6	16081	16424	16155	17375	16033	16143
7	16811	16334	16036	17727	16119	15846
8	16328	16577	16218	16742	16060	16048
9	16365	16045	16272	17030	15649	16326
10	16233	16463	16086	17140	15655	15995
ORTALAMA	**16437**	**16389**	**16281**	**17204**	**15989**	**15986**
AÇIKLIK/ORT. (%)	**5,0**	**4,8**	**5,7**	**5,7**	**4,1**	**4,0**
ÇEY.AÇIK./ORT. (%)	**2,1**	**1,7**	**1,1**	**3,2**	**1,8**	**1,2**

Tablo 4.48 IDES Tüm küçük veri REST JSON cevap süreleri (ms)

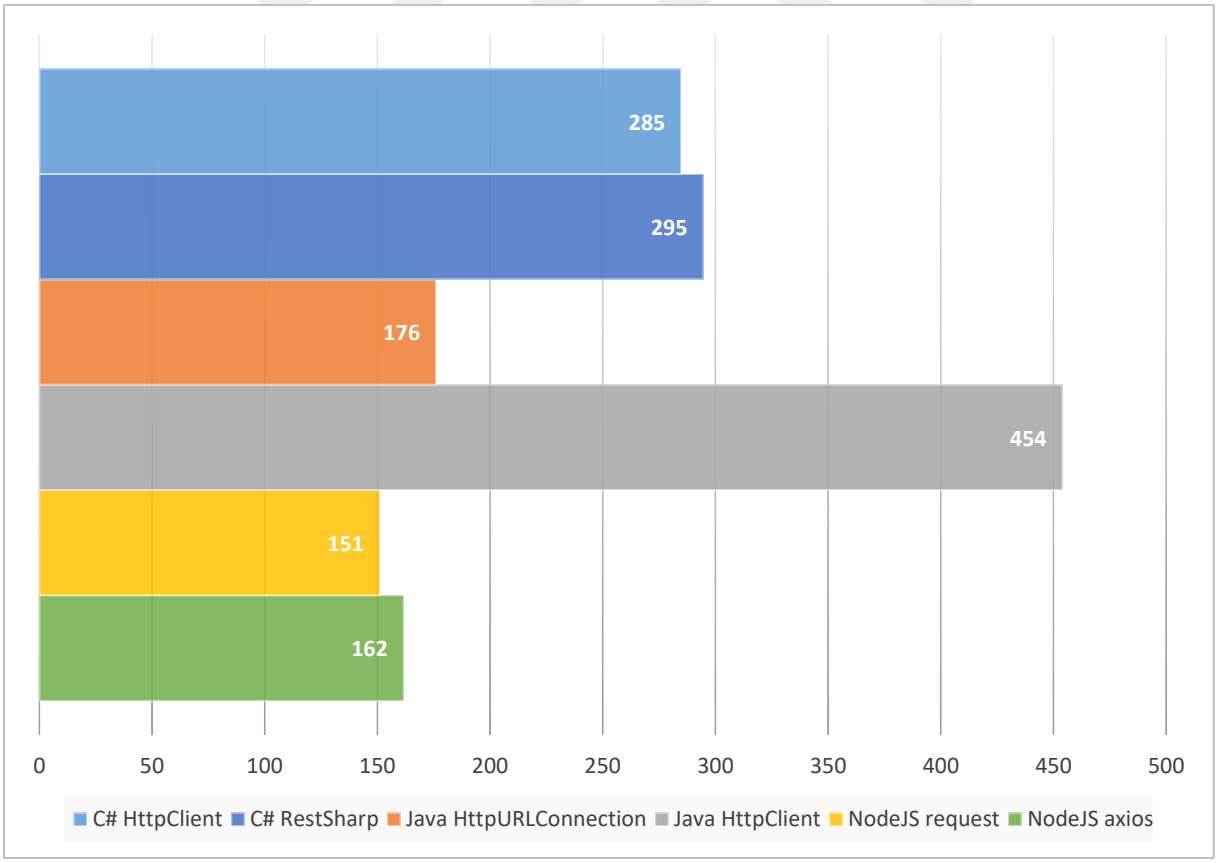
Tekrar	C# HttpClient	C# RestSharp	Java HttpURLConnection	Java HttpClient	Node.js request	Node.js axios
1	255	300	180	487	140	168
2	244	359	170	478	128	154
3	224	290	181	463	154	166
4	230	303	179	447	139	165
5	293	293	180	429	168	143
6	319	288	178	435	152	150
7	280	286	144	431	152	143
8	292	265	172	453	177	147
9	347	275	186	485	154	175
10	364	288	190	432	147	206
ORTALAMA	**285**	**295**	**176**	**454**	**151**	**162**
AÇIKLIK/ORT. (%)	**49,2**	**31,9**	**26,1**	**12,8**	**32,4**	**39,0**
ÇEY.AÇIK./ORT. (%)	**23,1**	**4,0**	**4,1**	**9,1**	**8,1**	**12,2**



Şekil 4.130 LOCAL küçük veri REST JSON ortalama cevap süreleri (ms)



Şekil 4.131 S4 küçük veri REST JSON ortalama cevap süreleri (ms)



Şekil 4.132 IDES küçük veri REST JSON ortalama cevap süreleri (ms)

4.2.3. Küçük Veri REST XML Çalışma Analizi

Tablo 4.49 LOCAL Tüm küçük veri REST XML cevap süreleri (ms)

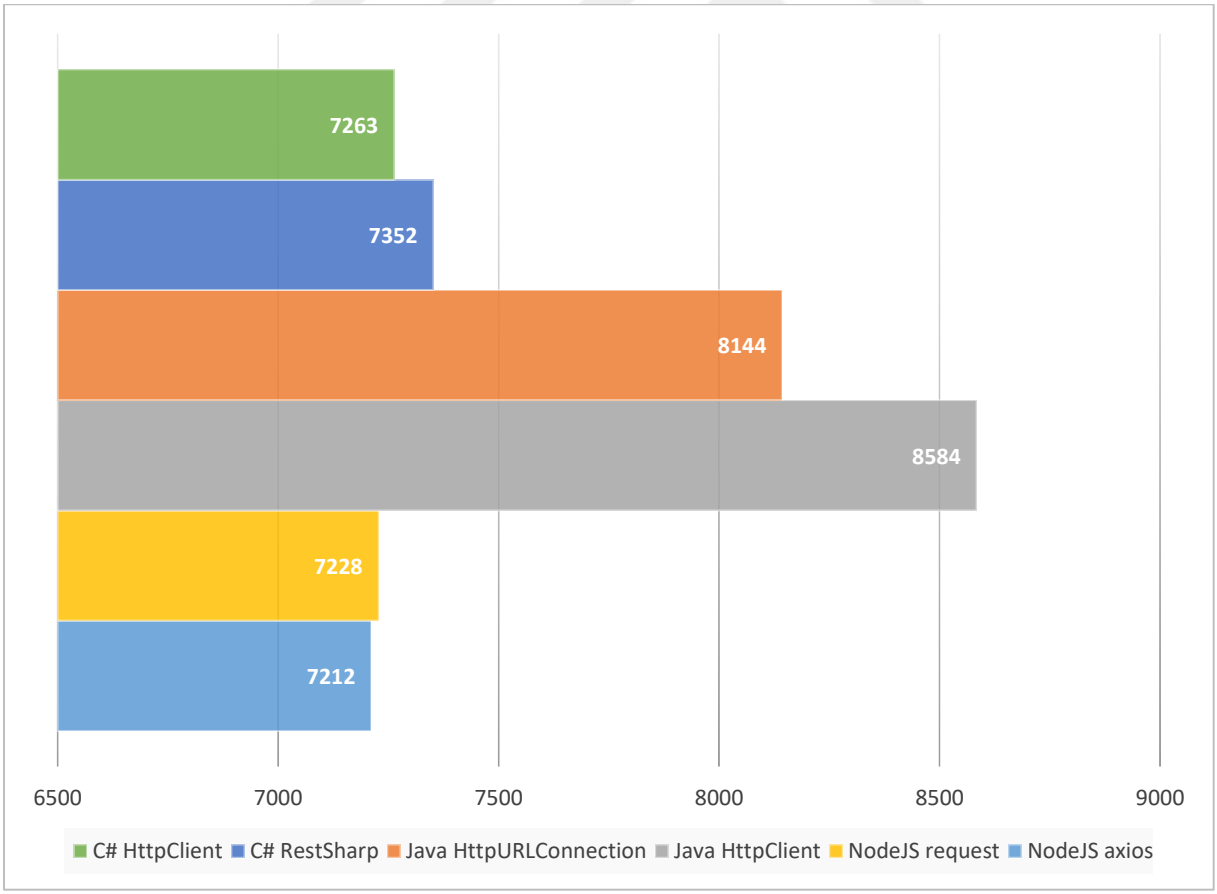
Tekrar	C# HttpClient	C# RestSharp	Java HttpURLConnection	Java HttpClient	Node.js request	Node.js axios
1	7368	7275	8287	8552	7267	7209
2	7323	7269	8206	8601	7159	7193
3	7225	7376	8170	8656	7221	7225
4	7299	7412	8218	8618	7121	7169
5	7246	7427	8242	8595	7164	7260
6	7148	7326	8073	8594	7295	7133
7	7238	7365	7999	8636	7059	7178
8	7210	7325	8080	8494	7295	7350
9	7285	7487	8008	8567	7348	7187
10	7291	7261	8153	8529	7349	7213
ORTALAMA	**7263**	**7352**	**8144**	**8584**	**7228**	**7212**
AÇIKLIK/ORT. (%)	**3,0**	**3,1**	**3,5**	**1,9**	**4,0**	**3,0**
ÇEY.AÇIK./ORT. (%)	**0,9**	**1,6**	**1,7**	**0,7**	**1,9**	**0,6**

Tablo 4.50 S4 Tüm küçük veri REST XML cevap süreleri (ms)

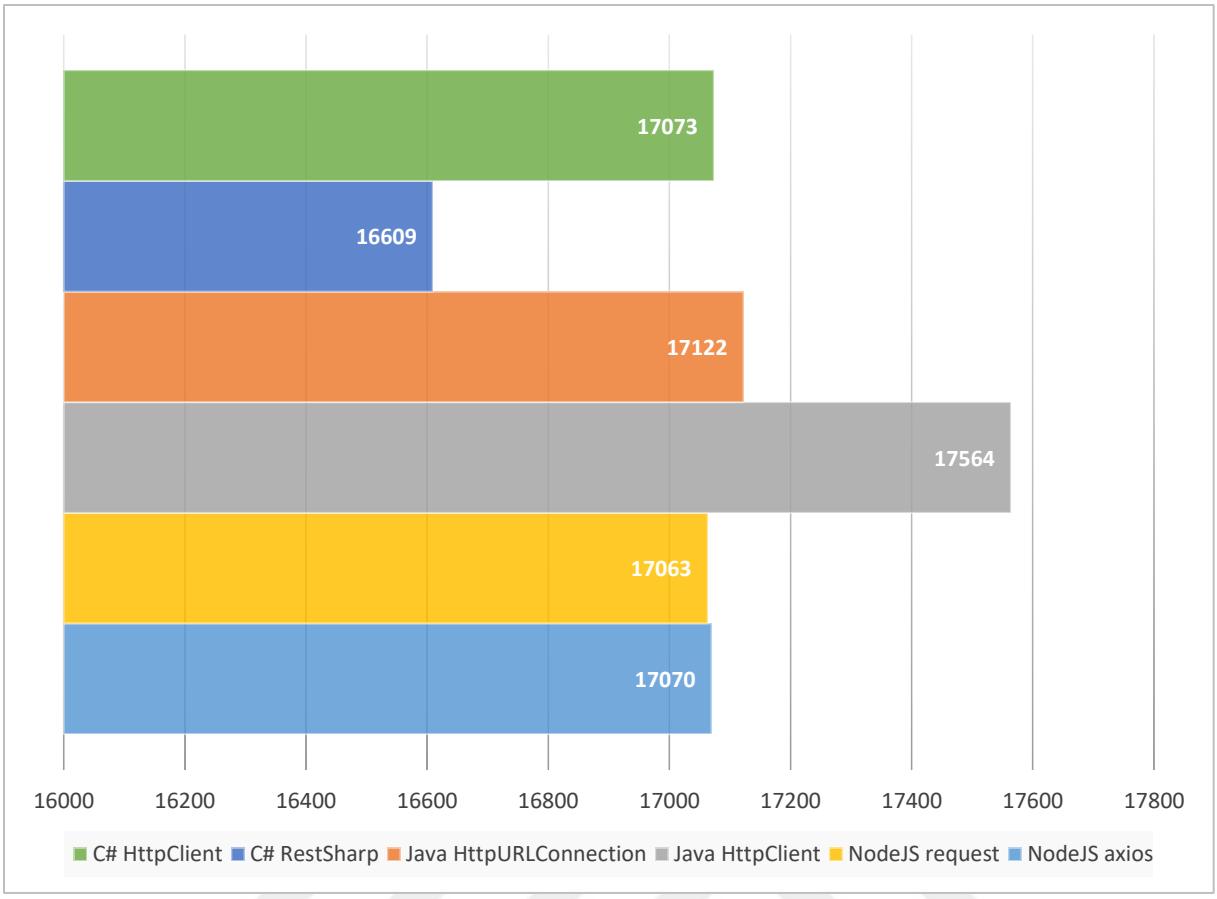
Tekrar	C# HttpClient	C# RestSharp	Java HttpURLConnection	Java HttpClient	Node.js request	Node.js axios
1	16717	16645	17210	17992	16933	16916
2	17201	16732	17342	17765	16935	16817
3	17790	16424	16683	17590	17773	17381
4	17154	16488	17245	17490	16859	17486
5	17095	16671	17510	16920	16909	17021
6	17192	16930	17036	18228	17345	17024
7	17015	16887	16995	16979	16771	17622
8	16447	16237	17023	17877	16457	16946
9	16904	16531	17069	17521	17751	17016
10	17219	16549	17110	17275	16898	16468
ORTALAMA	**17073**	**16609**	**17122**	**17564**	**17063**	**17070**
AÇIKLIK/ORT. (%)	**7,9**	**4,2**	**4,8**	**7,4**	**7,7**	**6,8**
ÇEY.AÇIK./ORT. (%)	**1,6**	**1,3**	**1,2**	**3,0**	**2,2**	**2,2**

Tablo 4.51 IDES Tüm küçük veri REST XML cevap süreleri (ms)

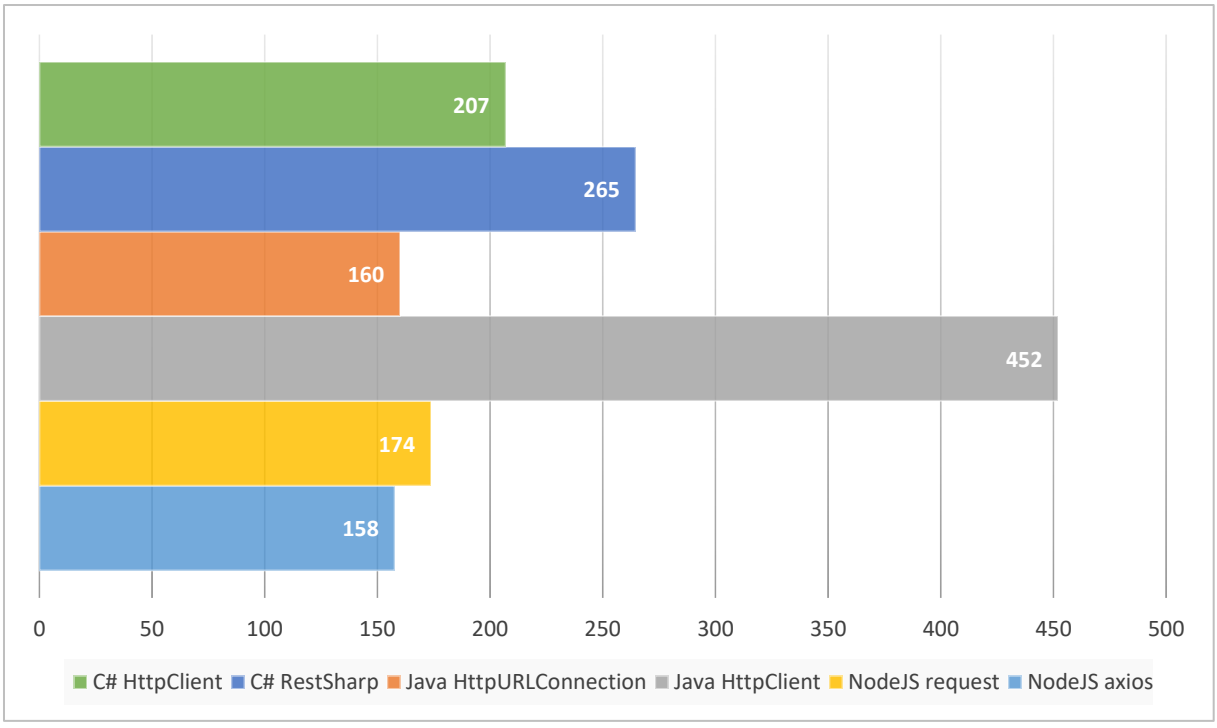
Tekrar	C# HttpClient	C# RestSharp	Java HttpURLConnection	Java HttpClient	Node.js request	Node.js axios
1	187	247	179	508	159	162
2	192	264	168	472	128	149
3	214	260	141	426	142	144
4	195	282	166	431	216	146
5	208	245	170	437	205	158
6	217	258	156	434	167	176
7	205	279	153	456	182	158
8	191	248	169	459	204	146
9	213	246	144	462	156	154
10	247	318	155	437	179	184
ORTALAMA	**207**	**265**	**160**	**452**	**174**	**158**
AÇIKLIK/ORT. (%)	**29,0**	**27,6**	**23,7**	**18,1**	**50,6**	**25,4**
ÇEY.AÇIK./ORT. (%)	**10,1**	**10,6**	**9,5**	**5,9**	**24,0**	**9,0**



Şekil 4.133 LOCAL küçük veri REST XML ortalama cevap süreleri (ms)



Şekil 4.134 S4 küçük veri REST XML ortalama cevap süreleri (ms)



Şekil 4.135 IDES küçük veri REST XML ortalama cevap süreleri (ms)

4.2.4. Büyük Veri SOAP Çalışma Analizi

Tablo 4.52 LOCAL Tüm büyük veri SOAP cevap süreleri (ms)

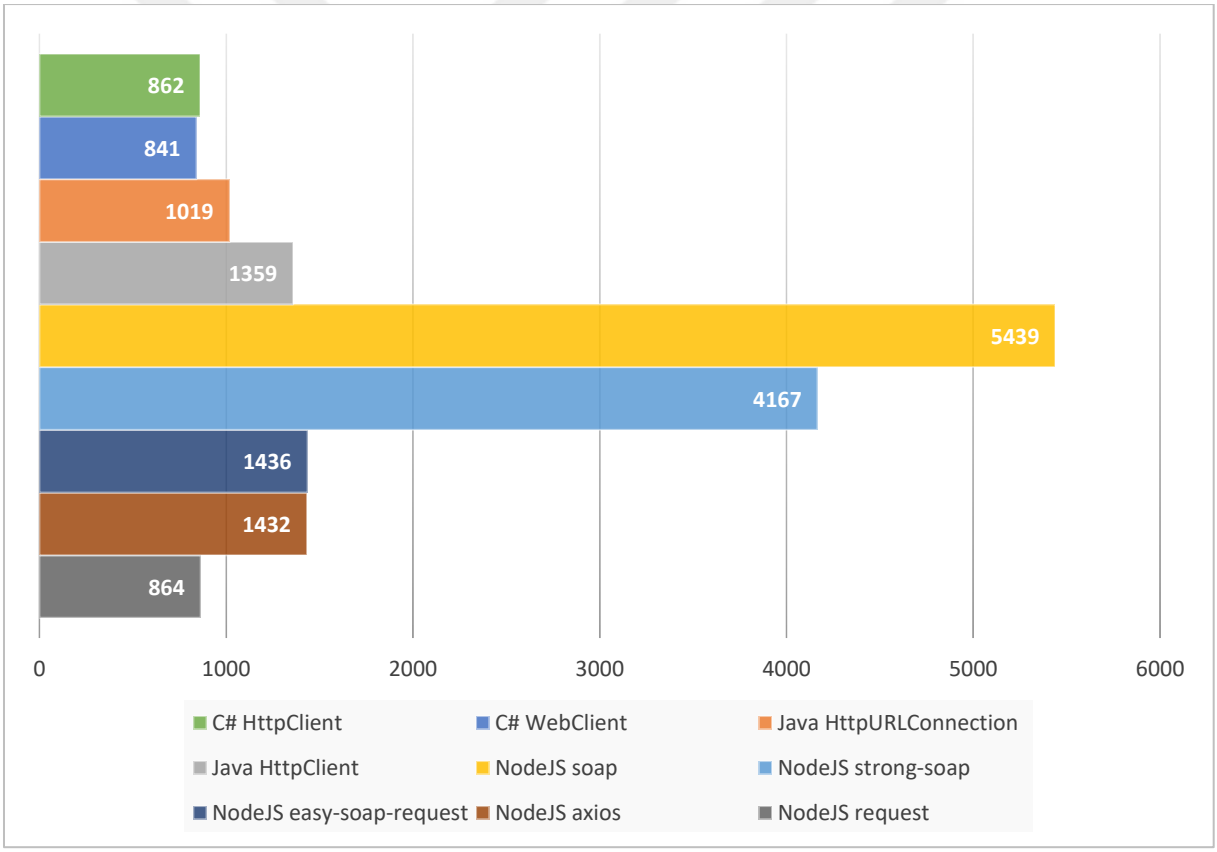
Tekrar	C# HttpClient	C# WebClient	Java HttpURLConnection	Java HttpClient	Node.js soap	Node.js strong-soap	Node.js easy-soap-request	Node.js axios	Node.js request
1	856	850	1006	1376	5422	4188	1425	1461	852
2	861	835	1013	1375	5469	4050	1431	1431	868
3	863	844	993	1338	5467	4235	1424	1418	884
4	860	841	1042	1369	5412	4211	1459	1431	890
5	868	837	1040	1339	5425	4151	1440	1421	826
ORTALAMA	**862**	**841**	**1019**	**1359**	**5439**	**4167**	**1436**	**1432**	**864**
AÇIKLIK/ORT. (%)	**1,4**	**1,8**	**4,8**	**2,8**	**1,0**	**4,4**	**2,4**	**3,0**	**7,4**
ÇEY.AÇIK./ORT. (%)	**0,3**	**0,8**	**3,3**	**2,6**	**0,8**	**1,4**	**1,0**	**0,7**	**3,7**

Tablo 4.53 S4 Tüm büyük veri SOAP cevap süreleri (ms)

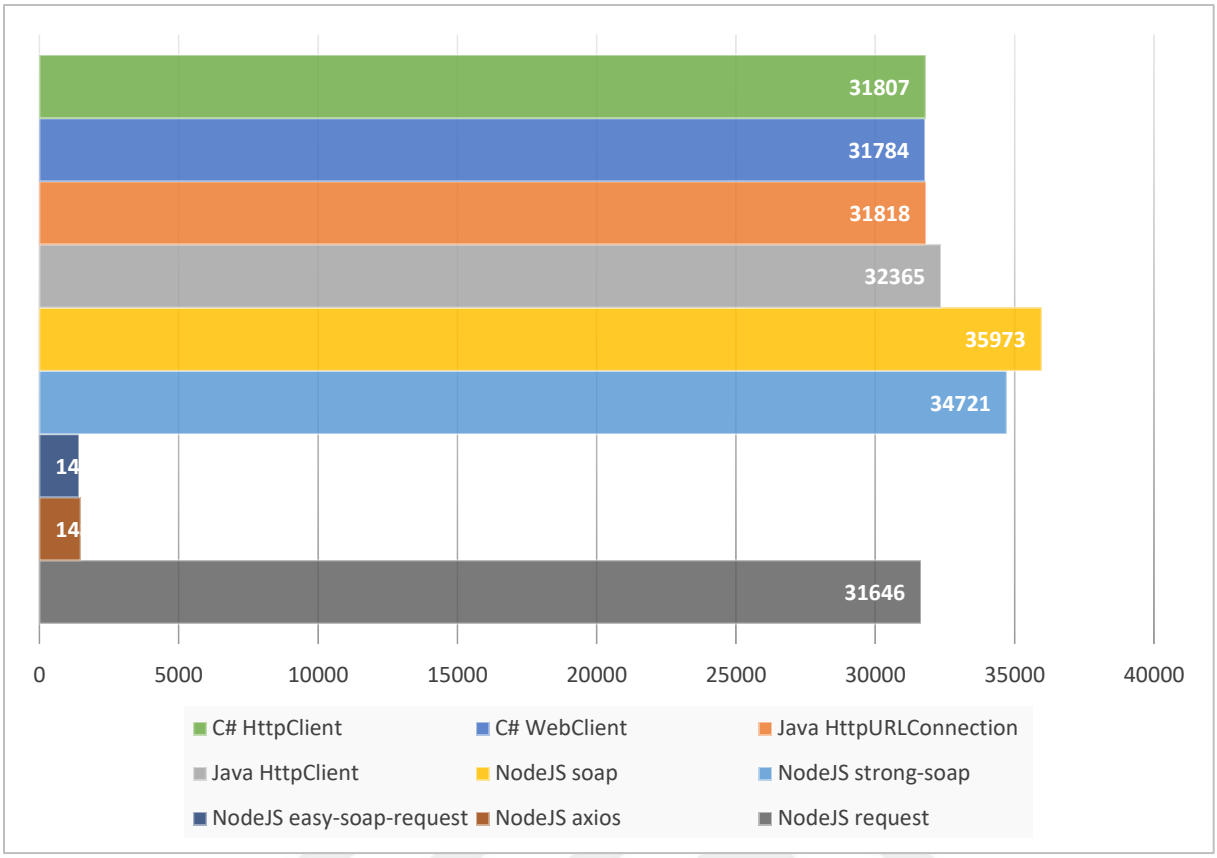
Tekrar	C# HttpClient	C# WebClient	Java HttpURLConnection	Java HttpClient	Node.js soap	Node.js strong-soap	Node.js easy-soap-request	Node.js axios	Node.js request
1	32012	31768	32142	32303	36229	34680	1494	1494	32080
2	31744	31960	31733	32370	35888	34608	1439	1511	31548
3	31772	31624	31674	32488	35812	34874	1423	1509	31576
4	31743	31789	31735	32281	35984	34564	1408	1445	31505
5	31763	31779	31804	32382	35953	34880	1379	1452	31522
ORTALAMA	**31807**	**31784**	**31818**	**32365**	**35973**	**34721**	**1429**	**1482**	**31646**
AÇIKLIK/ORT. (%)	**0,8**	**1,1**	**1,5**	**0,6**	**1,2**	**0,9**	**8,0**	**4,5**	**1,8**
ÇEY.AÇIK./ORT. (%)	**0,1**	**0,1**	**0,2**	**0,2**	**0,3**	**0,8**	**2,2**	**3,8**	**0,2**

Tablo 4.54 IDES Tüm büyük veri SOAP cevap süreleri (ms)

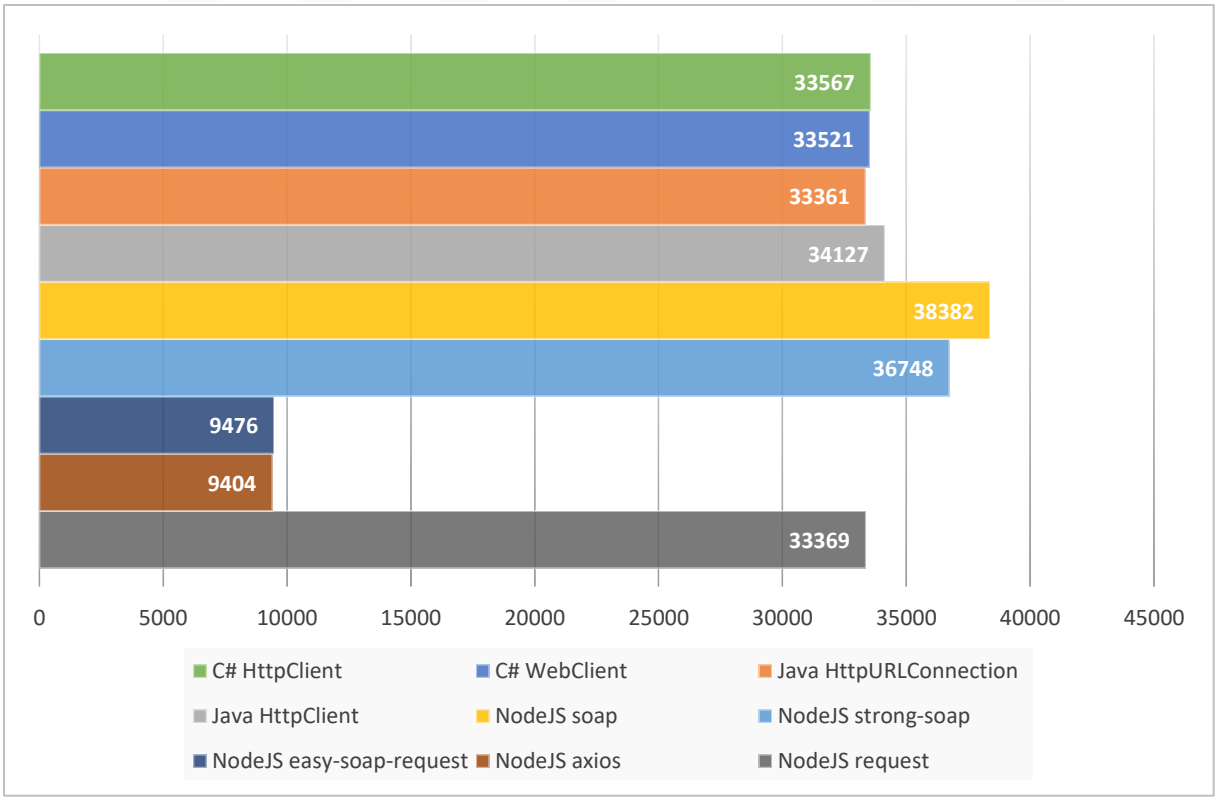
Tekrar	C# HttpClient	C# WebClient	Java HttpURLConnection	Java HttpClient	Node.js soap	Node.js strong-soap	Node.js easy-soap-request	Node.js axios	Node.js request
1	34269	33611	33471	34328	38540	36581	9490	9466	33529
2	33310	33279	33277	34059	38149	36713	9489	9458	33368
3	33530	33401	33373	34039	38341	36515	9472	9359	33347
4	33467	33612	33408	34233	39036	36637	9506	9393	33282
5	33260	33702	33276	33976	37843	37293	9423	9343	33318
ORTALAMA	**33567**	**33521**	**33361**	**34127**	**38382**	**36748**	**9476**	**9404**	**33369**
AÇIKLIK/ORT. (%)	**3,0**	**1,3**	**0,6**	**1,0**	**3,1**	**2,1**	**0,9**	**1,3**	**0,7**
ÇEY.AÇIK./ORT. (%)	**0,7**	**0,6**	**0,4**	**0,6**	**1,0**	**0,4**	**0,2**	**1,1**	**0,1**



Şekil 4.136 LOCAL büyük veri SOAP ortalama cevap süreleri (ms)



Şekil 4.137 S4 büyük veri SOAP ortalama cevap süreleri (ms)



Şekil 4.138 IDES büyük veri SOAP ortalama cevap süreleri (ms)

4.2.5. Büyük Veri REST JSON Çalışma Analizi

Tablo 4.55 LOCAL Tüm büyük veri REST JSON cevap süreleri (ms)

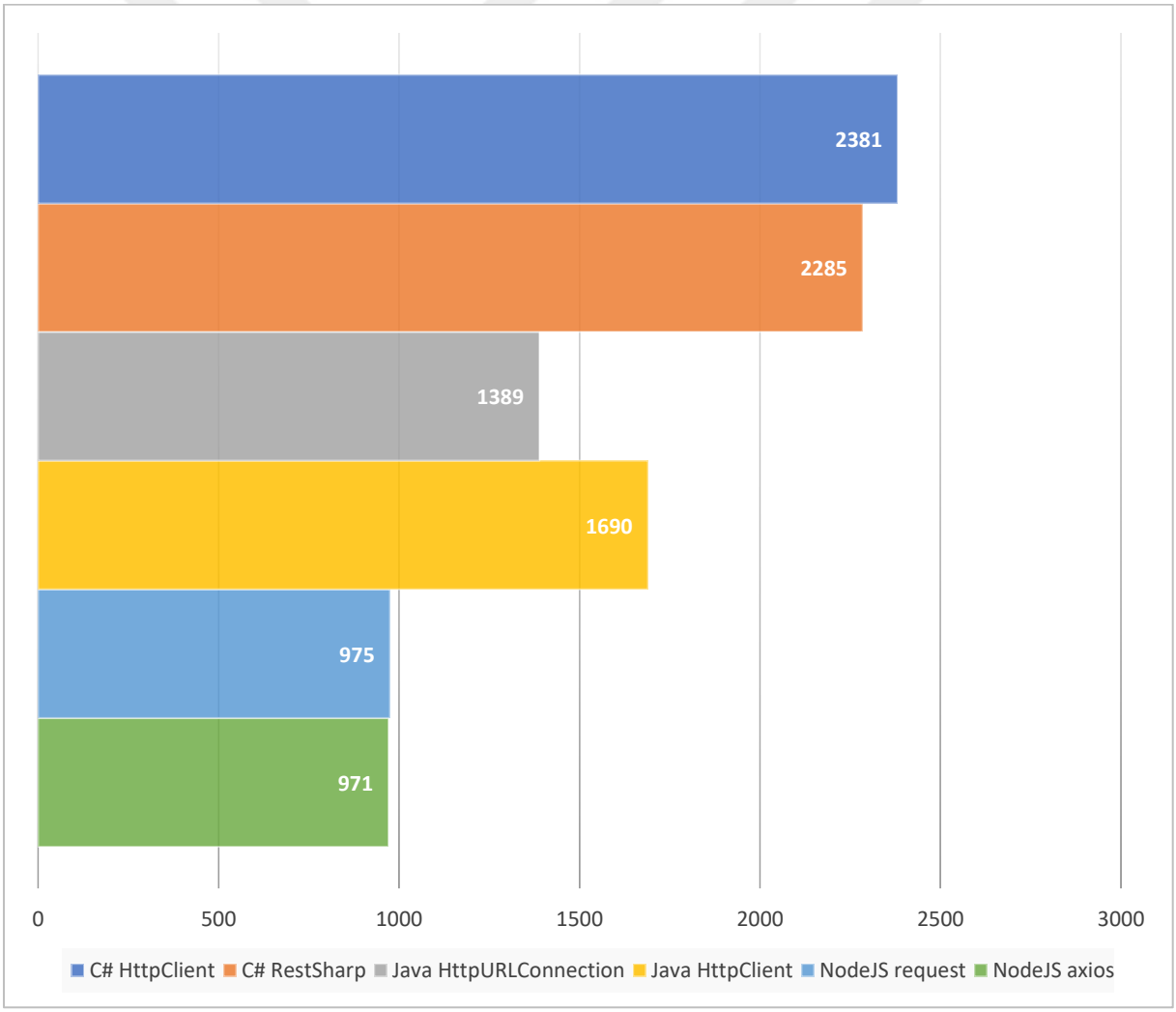
Tekrar	C# HttpClient	C# RestSharp	Java HttpURLConnection	Java HttpClient	Node.js request	Node.js axios
1	2377	2281	1378	1693	1004	992
2	2374	2284	1447	1721	922	964
3	2380	2285	1382	1685	971	961
4	2381	2288	1348	1669	993	971
5	2395	2286	1389	1681	986	967
ORTALAMA	**2381**	**2285**	**1389**	**1690**	**975**	**971**
AÇIKLIK/ORT. (%)	**0,9**	**0,3**	**7,1**	**3,1**	**8,4**	**3,2**
ÇEY.AÇIK./ORT. (%)	**0,2**	**0,1**	**0,8**	**0,7**	**2,3**	**0,7**

Tablo 4.56 S4 Tüm büyük veri REST JSON cevap süreleri (ms)

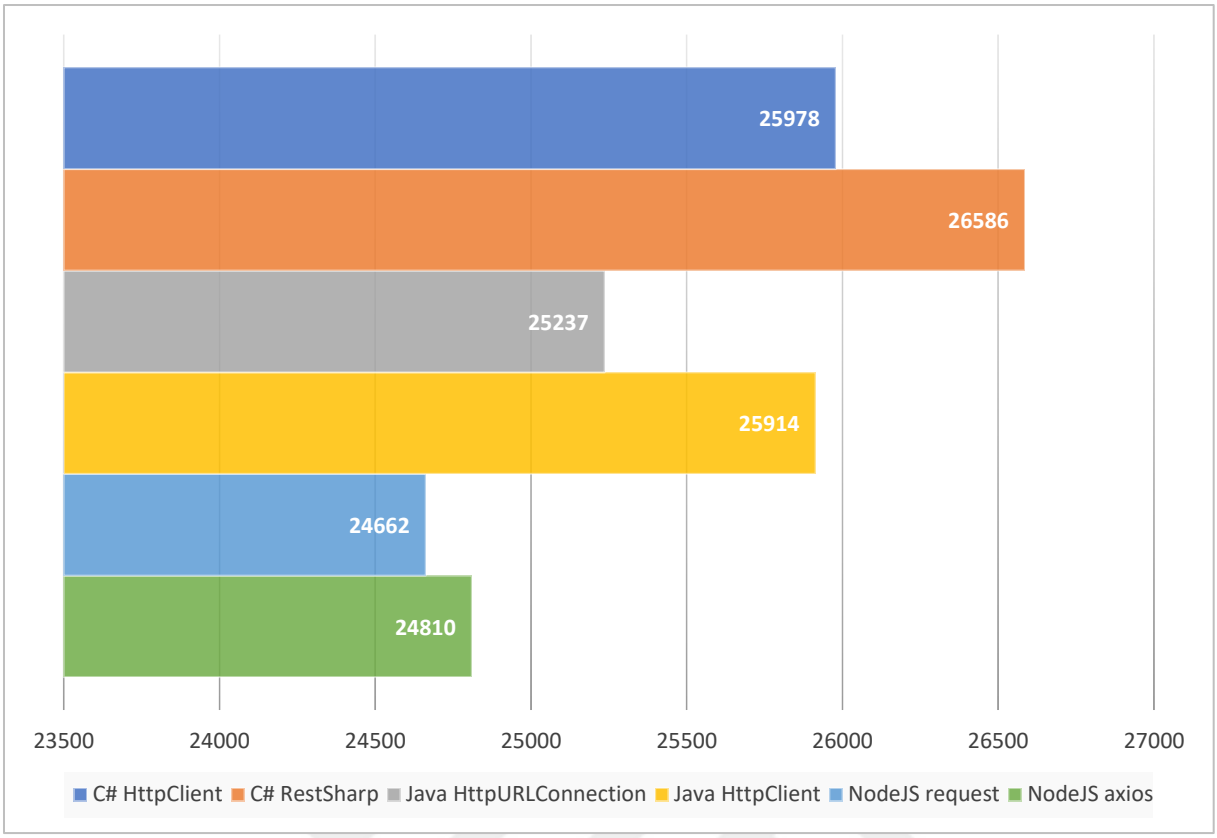
Tekrar	C# HttpClient	C# RestSharp	Java HttpURLConnection	Java HttpClient	Node.js request	Node.js axios
1	26191	26191	25181	26013	24715	24892
2	25949	25989	25183	25966	24620	24615
3	25907	26833	25429	25913	24646	24668
4	25988	26929	25166	25777	24708	24590
5	25857	26987	25225	25899	24623	25284
ORTALAMA	**25978**	**26586**	**25237**	**25914**	**24662**	**24810**
AÇIKLIK/ORT. (%)	**1,3**	**3,8**	**1,0**	**0,9**	**0,4**	**2,8**
ÇEY.AÇIK./ORT. (%)	**0,3**	**2,8**	**0,2**	**0,3**	**0,3**	**1,1**

Tablo 4.57 IDES Tüm büyük veri REST JSON cevap süreleri (ms)

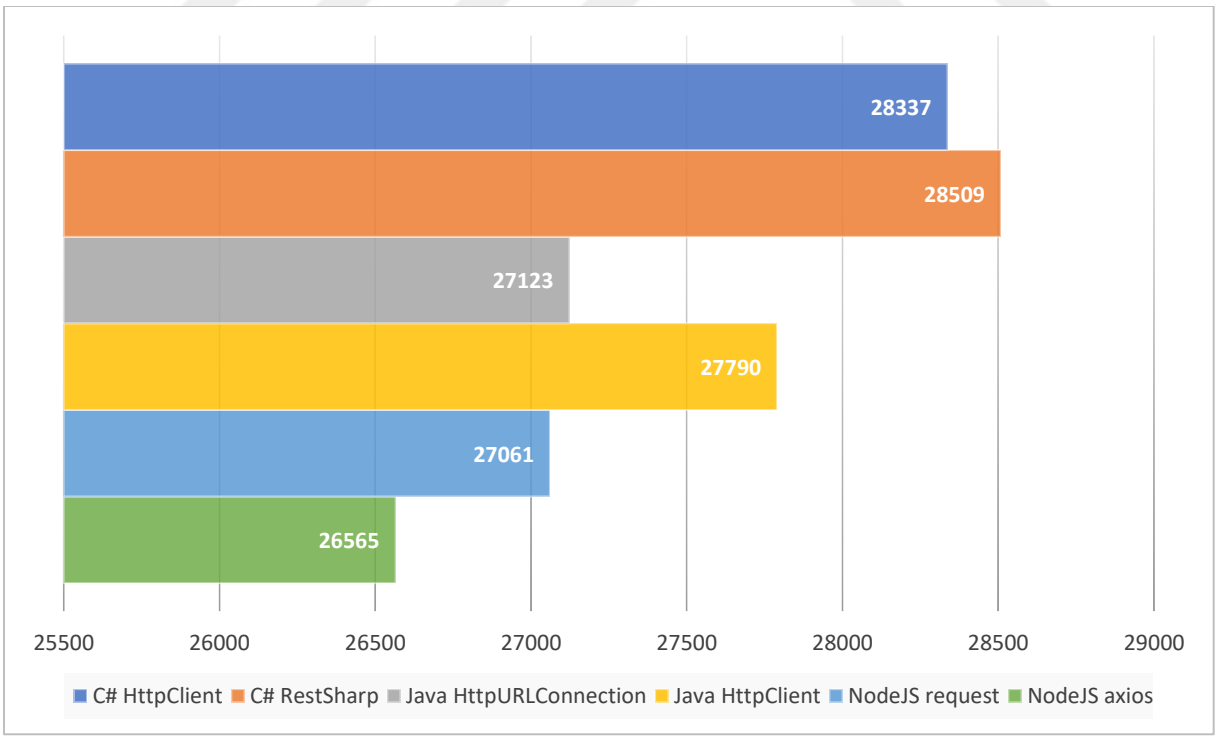
Tekrar	C# HttpClient	C# RestSharp	Java HttpURLConnection	Java HttpClient	Node.js request	Node.js axios
1	28182	28555	27121	28060	26892	26632
2	28315	28556	27392	27690	26705	26536
3	28380	28440	26894	27704	26865	26688
4	28448	28412	27174	27721	27408	26620
5	28359	28581	27035	27774	27437	26350
ORTALAMA	**28337**	**28509**	**27123**	**27790**	**27061**	**26565**
AÇIKLIK/ORT. (%)	**0,9**	**0,6**	**1,8**	**1,3**	**2,7**	**1,3**
ÇEY.AÇIK./ORT. (%)	**0,2**	**0,4**	**0,5**	**0,3**	**2,0**	**0,4**



Şekil 4.139 LOCAL büyük veri REST JSON ortalama cevap süreleri (ms)



Şekil 4.140 S4 büyük veri REST JSON ortalama cevap süreleri (ms)



Şekil 4.141 IDES büyük veri REST JSON ortalama cevap süreleri (ms)

4.2.6. Büyük Veri REST XML Çalışma Analizi

Tablo 4.58 LOCAL Tüm büyük veri REST XML cevap süreleri (ms)

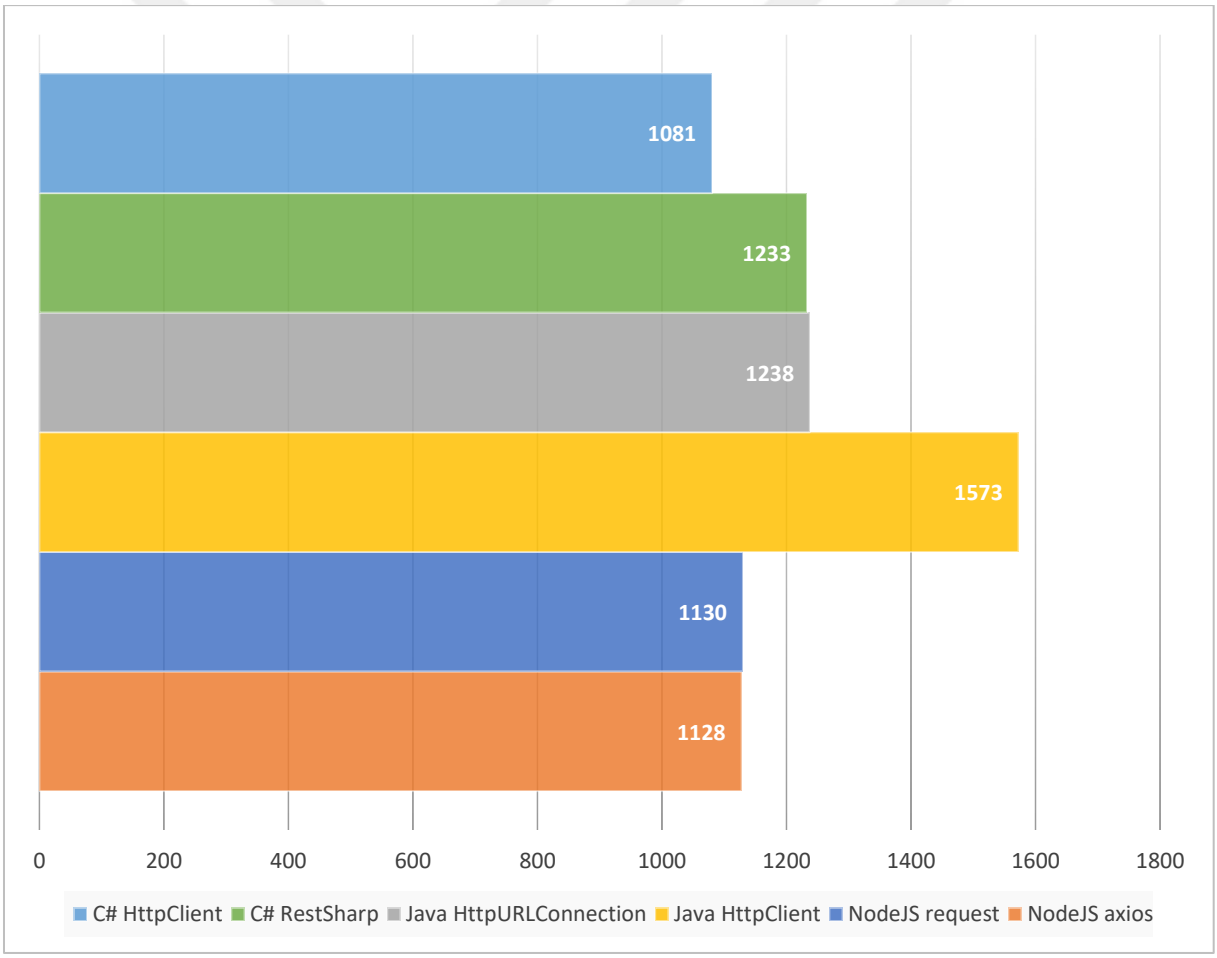
Tekrar	C# HttpClient	C# RestSharp	Java HttpURLConnection	Java HttpClient	Node.js request	Node.js axios
1	1083	1234	1283	1606	1151	1159
2	1072	1223	1207	1578	1139	1133
3	1087	1236	1225	1555	1149	1134
4	1077	1237	1270	1542	1144	1133
5	1085	1235	1204	1584	1067	1080
ORTALAMA	**1081**	**1233**	**1238**	**1573**	**1130**	**1128**
AÇIKLIK/ORT. (%)	**1,4**	**1,1**	**6,4**	**4,1**	**7,4**	**7,0**
ÇEY.AÇIK./ORT. (%)	**0,7**	**0,2**	**5,1**	**1,8**	**0,9**	**0,1**

Tablo 4.59 S4 Tüm büyük veri REST XML cevap süreleri (ms)

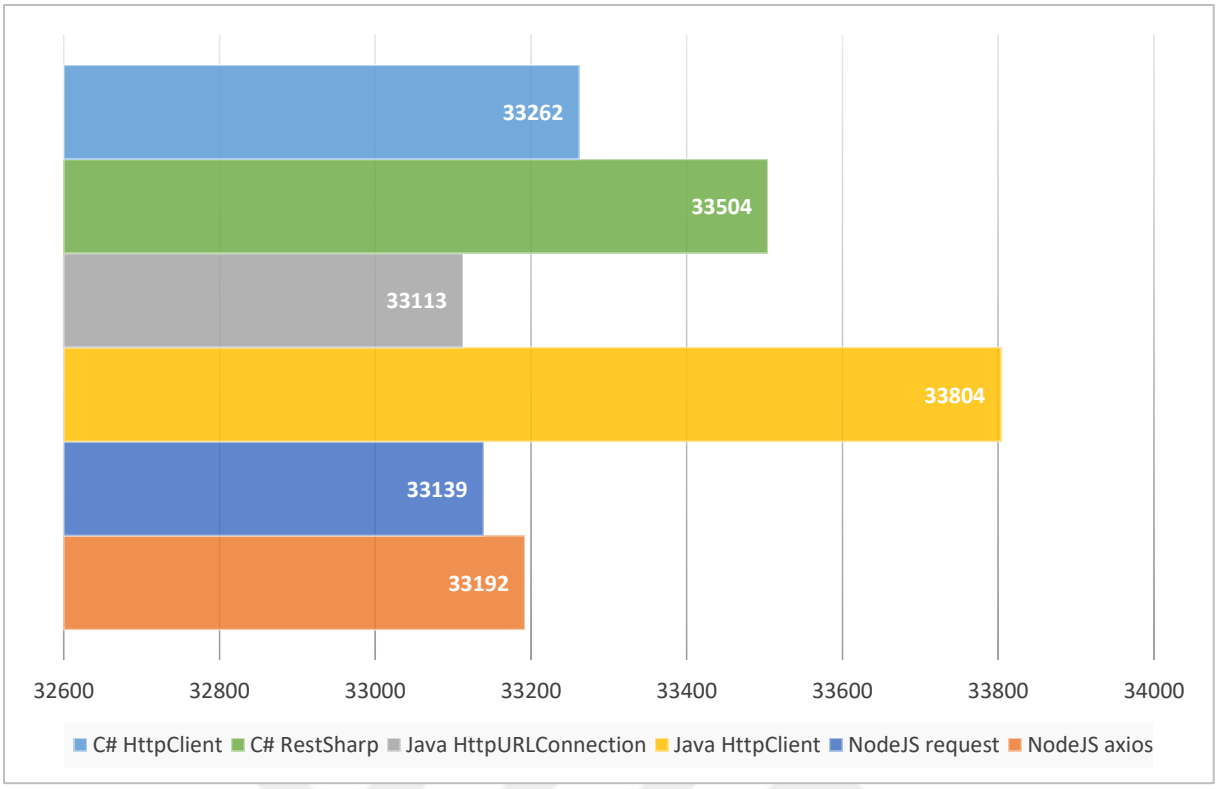
Tekrar	C# HttpClient	C# RestSharp	Java HttpURLConnection	Java HttpClient	Node.js request	Node.js axios
1	33298	33556	33076	33864	33054	33202
2	33364	33485	33177	33803	33009	33107
3	33181	33584	33064	33699	32999	33190
4	33180	33429	33163	33849	33085	33070
5	33288	33468	33085	33804	33549	33393
ORTALAMA	**33262**	**33504**	**33113**	**33804**	**33139**	**33192**
AÇIKLIK/ORT. (%)	**0,6**	**0,5**	**0,3**	**0,5**	**1,7**	**1,0**
ÇEY.AÇIK./ORT. (%)	**0,4**	**0,3**	**0,3**	**0,1**	**0,2**	**0,3**

Tablo 4.60 IDES Tüm büyük veri REST XML cevap süreleri (ms)

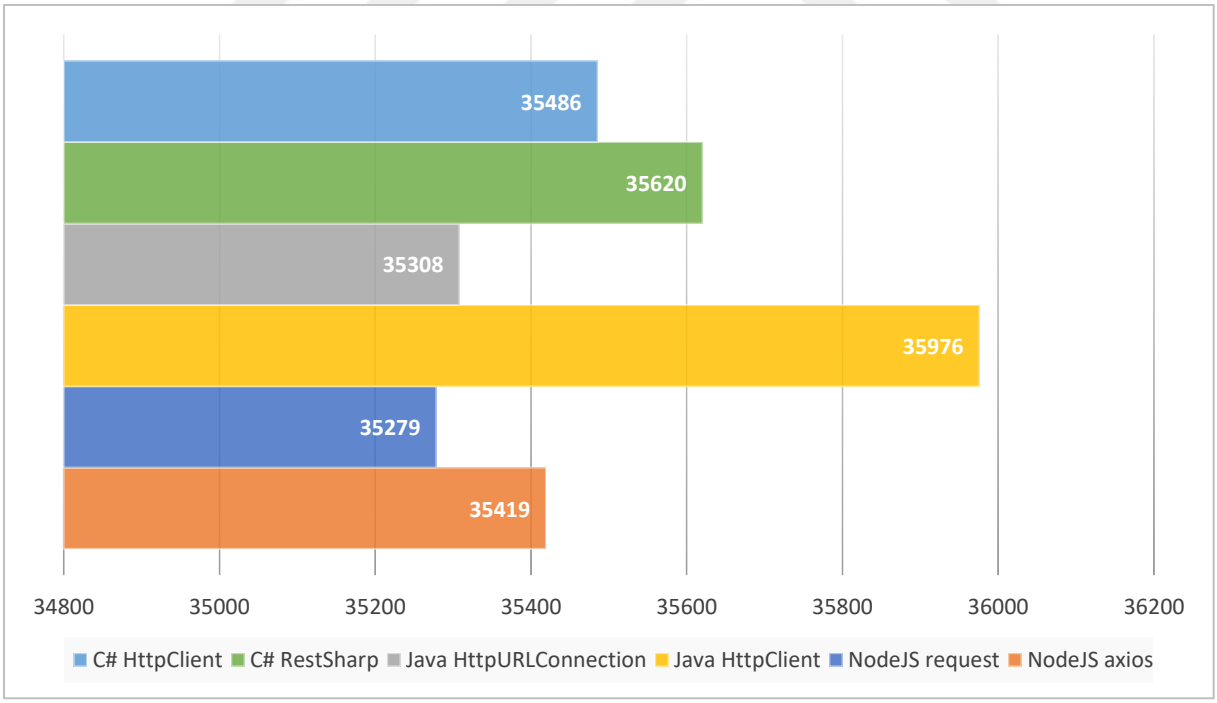
Tekrar	C# HttpClient	C# RestSharp	Java HttpURLConnection	Java HttpClient	Node.js request	Node.js axios
1	35658	35740	35317	36079	35262	35374
2	35575	35551	35259	35945	35289	35404
3	35443	35572	35294	35925	35258	35363
4	35423	35678	35174	35905	35403	35657
5	35329	35561	35497	36026	35181	35299
ORTALAMA	**35486**	**35620**	**35308**	**35976**	**35279**	**35419**
AÇIKLIK/ORT. (%)	**0,9**	**0,5**	**0,9**	**0,5**	**0,6**	**1,0**
ÇEY.AÇIK./ORT. (%)	**0,4**	**0,3**	**0,2**	**0,3**	**0,1**	**0,1**



Şekil 4.142 LOCAL büyük veri REST XML ortalama cevap süreleri (ms)



Şekil 4.143 S4 büyük veri REST XML ortalama cevap süreleri (ms)



Şekil 4.144 IDES büyük veri REST XML ortalama cevap süreleri (ms)

5. TARTIŞMA VE SONUÇ

Bu tez kapsamında yapılan çalışmalarla bir kurumsal kaynak planlama iş yazılımı olan SAP sistemlerinden dışarıya açılması istenen verilerin boyutlarına göre seçilecek istemci (client), kullanıcı programlama arayüzü (API) ve veri formatları taraflı yaşanabilecek performans farkları ortaya konmuştur.

Elde edilen sonuçlar; istemci tarafında istek mesajının oluşturulması ve gönderilmesi, SAP tarafında mesajın alınması, ilgili programların çalıştırılması, veritabanı bağlantılarının yapılması, REST isteklerde verilerin JSON veya XML formatına dönüştürülmesi, verilerin paketlenmesi ve gönderilmesi, tekrar istemci tarafında cevabın alınması ve her satırın en az 1 kez yorumlanması süreciyle elde edilmiştir. Yani SAP tarafında SOAP mesajlar direkt çıkış yapabilirken, REST mesajlar için SAP içerisinde ek olarak her satır veriye XML veya JSON yazım formatına dönüştürme uygulanmaktadır. İstemci tarafında ise her satır veri bir sayaç fonksiyonu ile sayılmakta böylelikle en az 1 kez yorumlanmaktadır. Fakat JavaScript, JSON cevapları doğrudan veya 1 kez parse fonksiyonundan geçirip yorumlayabiliyorken, C# ve Java programlama dillerinde bu işlem için ek olarak JSON Object dönüşümü ve peşinden JSON Array dönüşümü gerçekleştirmek gerekmekte, bu da her bir satır verinin fazladan 2 kere yorumlanması anlamına gelmektedir. Gerçek hayat kullanımlarında bu senaryolar gerçekleşeceği için bu çalışmada da bu dönüşümlere de yer verilmiş, sonuçlar bu şekilde toplanmıştır. Bu dönüşüm işlemlerinin yer aldığı tüm program kodları ilgili çalışma başlığının altında belirtilen EK'lerde paylaşılmıştır.

Genel Değerlendirme

Küçük verilerle yapılan tüm çalışmalarda Java programlama dilindeki HttpClient sınıfı en yavaş sonuçları vermiştir. Büyük verilerle yapılan çalışmalarda da ortalama altı performans göstermiştir. Java programlama dilinde alternatif olarak kullanılan HttpURLConnection sınıfı ise hem SOAP hem REST çalışmalarında ortalama ve çeyrekler ortalaması üzerinde performans sergilemiştir. Java programlama dilinin en eski sınıflarından olan HttpURLConnection sınıfı yapılan her çalışmada HttpClient sınıfından daha performanslı sonuçlar sağlamıştır. HttpURLConnection sınıfı Java 1.1 sürümü ile 1997 yılında ve devam formu HttpsURLConnection sınıfı ise Java 1.4 sürümü ile 2002 yılında desteğe kavuşmuştur. HttpClient sınıfı ise çok daha yeni sayılabilecek Java 11 sürümü ile 2018 yılının son çeyreğinde tam desteğine kavuşmuştur. Günümüzde, topluluk gözünde hantal ve zor bir yazımı olduğu

düşünülen ve bazı günümüz ihtiyaç teknolojilerini karşılayamayan HttpURLConnection sınıfı yerine daha akıcı ve kolay bir yazımı olduğu düşünülen ve HTTP/2 asenkron çalışma gibi diğer ihtiyaç teknolojilerini karşılayabilen HttpClient sınıfı tercih edilmektedir. Fakat bu tez kapsamındaki çalışmalarda SAP sunucularında HTTP/1.1 üzerinden yapılan hem SOAP 1.1 hem de REST çalışmalarının tamamında HttpURLConnection sınıfının kullanımının HttpClient sınıfına oranla daha performanslı sonuçlar veren bir tercih olacağı net bir şekilde görülmüştür. Çalışmalardan elde edilen sonuçların dağılımları ise iki alternatif için de sonucu değiştiremeyecek kadar düzenlidir.

SOAP

SAP sistemlerinden verileri dışarıya SOAP API ile WSDL formatında açmak istediğimizde çoğu senaryoda en iyi tercih Node.js ortamı olacaktır. Veri boyutu küçüldükçe C# programlama dilinde ise WebClient sınıfı kabul edilebilir sonuçlar vermektedir. Java programlama dilinde ise HttpClient sınıfı çoğu çalışmada olduğu gibi bu çalışmalarda da en yavaş veya ortalama altı performans sergilemiştir ve performans odaklı çalışmalarda tercih edilmemelidir.

Bir XML türevi olan WSDL formatındaki küçük verilerle SOAP API üzerinden yapılan çalışmalarda her üç sunucuda da ortak olarak en yüksek değer alınan sonuçlar, yani en yavaş olan çalışma Java programlama dilindeki HttpClient sınıfı kullanılarak yapılan çalışmadır. Çalışmalarda en düşük değer alınan sonuçlar, yani en hızlı olan çalışmalar ise Node.js çalışmaları olmuştur. Node.js ortamında beş alternatif kütüphane ile çalışılmış ve üç farklı sunucuda da ortak olarak veri boyutu büyüdükçe axios ve easy-soap-request çalışmaları en hızlı sonuçları vermiştir. Ama yüksek açıklık ve çeyrekler açıklığı oranlarıyla karşılaşılmıştır. Veri boyutu küçüldükçe request kütüphanesi de aynı performansları benzer hatta daha kararlı dağılımlarla verebilmiştir. C# programlama dilindeki WebClient sınıfı ortalama üzeri performans sunmuştur. Diğer programlama dilleriyle yapılan çalışmalarda rekabetçi sonuçlar elde edilememiştir.

WSDL formatındaki büyük verilerle SOAP API üzerinden yapılan çalışmalarda her üç sunucuda da ortak olarak en yüksek ve de en düşük performans sergileyen çalışmalar Node.js ortamındaki çalışmalar olmuştur. Node.js ortamındaki soap ve strong-soap kütüphaneleri sırasıyla ve diğer çalışmalara oranla fark yaratır düzeyde en yavaş iki performansı göstermiştir. En hızlı performansları ise ortalamaya ve de çeyrekler ortalamasına çok yüksek oransal farklarla Node.js ortamındaki axios ve easy-soap-request kütüphaneleri göstermiştir. Küçük

verilerle olan çalışmalardaki dağılımlarının aksine daha düzenli sonuçlar elde edilmiştir. Node.js ekosistemindeki ilk kütüphanelerden olan ve ilk versiyonunu 2009 yılında tanıtan request kütüphanesi 2020 itibariyle zamanın gerisinde kalan bir çekirdeğe sahip olduğu gerekçesiyle tüm desteğini sonlandırmış(deprecated) ve bakım moduna(maintenance mode) geçmiş durumdadır. 2023 itibariyle 3 senedir güncelleme almamış olmasına karşın hala haftalık indirmeleri 15 milyonun üzerinde kalmakta hatta zaman zaman 20 milyona yaklaşmaktadır. Daha genç bir kütüphane olarak ilk versiyonunu 2014 yılında tanıtan axios ise bu tez çalışması kapsamında gerçeklenen senaryoların neredeyse tamamında request ile yakın sonuçlar elde eden diğer bir Node.js kütüphanesi olmuştur. Düzenli güncellemeler almaya devam eden ve mimari açıdan daha güvenli ve daha fazla kontrol alanı sağlayan bir yapıda olan axios, 2021 yılının son çeyreği itibariyle haftalık indirmelerde request kütüphanesini geçmiş, günümüzde ise aralarında 2 katı aşkın bir fark axios liderliğinde oluşmuş durumdadır. İki kütüphane de hem REST hem SOAP istekleri için kullanılabilmektedir.

Bu tez kapsamında SOAP çalışmaları için Node.js ortamında toplam beş olmak üzere üç alternatif kütüphane daha kullanılmıştır. İlk ikisi REST çalışmalarında da kullanılan axios ve request kütüphaneleri olmak üzere kalan üçü sırasıyla soap, strong-soap ve easy-soap-request kütüphaneleridir. Bunlardan soap, Node.js ortamındaki en eski SOAP isteklerini yönetme amaçlı kütüphanesidir. Eski adı node-soap olan bu kütüphaneden geliştirilerek 2016 yılında ekosisteme kazandırılan diğer bir alternatif ise strong-soap kütüphanesidir. Kullanımları ve performansları birbirleriyle oldukça yakındır. Kullanılan son ve en genç alternatif ise axios kütüphanesinden geliştirilerek 2018 yılında ekosisteme kazandırılan ve performansları da birbirleriyle oldukça yakın olan easy-soap-request kütüphanesidir. Haftalık indirmeleri ise 2023 yılında soap kütüphanesi için 400 bin civarında olup, strong-soap ve easy-soap-request kütüphaneleri için ise on binler seviyesindedir.

Bu tez kapsamında gerçeklenen test senaryolarının büyük çoğunluğunda en performanslı ve birbirleriyle çok yakın sonuçlara sahip iki çalışma Node.js ortamındaki axios ve request kütüphanelerine aittir. Lakin SOAP çalışmalarında veri boyutları büyüdükçe axios çalışmaları, request çalışmalarına karşı ciddi performans kazanımı göstermektedir. Veri boyutu on binlerce ve hatta yüz binlerce satırı geçtiği SOAP çalışmalarında çok ciddi farklar axios ve easy-soap-request kütüphaneleri liderliğinde ortaya çıkmaktadır. Node.js ortamındaki diğer alternatif soap, strong-soap ve request kütüphaneleri veya bu tez kapsamında kullanılan C# ve Java programlama dillerindeki diğer alternatif çalışmalar veri boyutu büyüdükçe SOAP çalışmalarında birbirleriyle benzer sonuçları getirmiş olup, üç sistemde de aynı veri setiyle

gerçeklenen “Büyük Veri SOAP Çalışmaları”nda axios ve easy-soap-request kütüphanelerine kıyasla ortalamada ve çeyrekler ortalamasında 2’den 22 kata kadar geride kaldıkları görülmektedir. Bu ciddi farklar nedeniyle performans odaklı SOAP çalışmalarında Node.js ortamında axios veya easy-soap-request kütüphaneleri tercih edilmelidir. Node.js ortamındaki soap ve strong-soap kütüphaneleri kolay kullanımları açısından tercih edilebilecek iken performans odaklı olarak tercih edilmemelidir.

JSON

JSON formatındaki küçük verilerle RESTful API üzerinden yapılan çalışmalarda her üç sunucuda da ortak olarak en yüksek değer alınan sonuçlar, yani en yavaş olan çalışma Java programlama dilindeki HttpClient sınıfı kullanılarak yapılan çalışmadır. JSON formatındaki büyük verilerle RESTful API üzerinden yapılan çalışmalarda her üç sunucuda da ortak olarak en yüksek değer alınan sonuçlar, yani en yavaş olan çalışma C# programlama dili kullanılarak yapılan çalışmadır. REST çalışmaları için .NET ortamında hazır olarak gelen HttpClient sınıfı ile onun geliştirilmesiyle ortaya çıkmış günümüzde popüler bir açık kaynak kütüphanesi olan RestSharp kullanılmıştır. Daha kolay kullanımı olmasına karşın RestSharp, HttpClient sınıfıyla olan çalışmalara kıyasla daha fazla bellek kullanımı gerçekleştirmektedir. Bu tez kapsamında hem XML hem JSON veri formatındaki çalışmalarda birbirlerine çok yakın sonuçlar elde etmiş olup, dağılımları diğer çalışmalardan düzensiz değildir. HttpClient sınıfı küçük farklarla daha performanslıdır. Tüm JSON çalışmalarında en düşük değer alınan sonuçlar, yani en hızlı olan çalışmalar Node.js çalışmaları olmuştur.

JSON formatındaki verilerin yorumlanması için C# programlama dilinde Json.NET çatısı, Java programlama dilinde JSON-Java kütüphanesi kullanılmıştır. Node.js ortamında ek bir kullanıma gerek yoktur. Bu durum da performansa oldukça etki etmektedir.

XML

XML formatındaki verilerle RESTful API üzerinden yapılan tüm çalışmalarda her üç sunucuda da ortak olarak en yüksek değer alınan sonuçlar, yani en yavaş olan çalışma Java programlama dilindeki HttpClient sınıfı kullanılarak yapılan çalışmadır.

Bu tez çalışmasında sonuçların birbirine en yakın olduğu ve açıklık veya çeyrekler açıklığı oranlarıyla en düzenli dağılım gösteren çalışmalar XML çalışmaları olmuştur. XML çalışmalarında en düşük değer alınan sonuçlar, yani en hızlı olan çalışmalar genelde Node.js

çalışmaları olmuştur. Fakat Java programlama dilindeki HttpClient sınıfı hariç diğer çalışmalar birbirlerine yakın değerlerle sonuçlanmıştır, bu yüzden performans odaklı çalışmalarda Node.js ortamındaki request veya axios kütüphanesini tavsiye etmekle beraber diğer programlama dillerinin de kullanılabileceğini söyleyebiliriz.

Sonuç

SAP sistemlerinden dışarıya veri açılan performans odaklı çalışmalarda API ve veri formatı seçiminde bu tez kapsamında yapılan çalışmalar en performanslı sonuçlara göre sıralandığında SOAP API seçiminin özellikle veri boyutu büyüdükçe net bir şekilde tek tercih olduğunu görüyoruz. Bu sonuçların da büyük çoğunluğu Node.js çalışmalarına aittir. Ama WSDL formatındaki veriler, XML veya JSON formatındaki aynı verilerden daha büyük dosya boyutlarına sahiptir. Buna rağmen ortaya çıkan performans farkı, SAP tarafında WSDL dönüştürücüsünün sistemin dahili bileşenleri tarafından yapılmasından ve JSON veya XML dönüştürme işlemlerinin ise kullanıcının seçeceği veya yazacağı fonksiyon veya sınıf metotlarıyla yapılmasından kaynaklıdır.

REST cevapları için SAP tarafında doğrudan bir desteğin olmaması, verilerin farklı yöntemlerle XML veya JSON formatına dönüştürülüp, sunucuya gelen REST isteklerinin bu dönüştürülmüş verilerle cevaplanması ile çözülüyor. XML formatındaki veriler, JSON formatındaki aynı verilerden daha fazla etiket ve elementler ile yaklaşık yüzde elli daha büyük dosya boyutuna sahip olabiliyor. İstemci tarafından alınan bu verilerin yorumlanmasında ise JSON obje ve dizilerine dönüştürme işlemleri bu farkı kapatıyor hatta JSON çalışmalarını geriye düşürüyor. Aynı işlemler tam tersinden hem XML hem JSON için SAP içerisinde de gerçekleştirildiği için performans kaybı SOAP çalışmalarıyla kıyasla giderek artıyor. Genel değerlendirme olarak performans odaklı çalışmalarda SAP sistemlerinden dışarıya SOAP API ile WSDL formatında veya RESTful API ile XML veya JSON formatında verileri açmak istediğimizde çoğu senaryoda en iyi tercih Node.js ortamı olacaktır. JSON formatıyla yapılan çalışmalarda C# programlama diliyle yapılan çalışmalar yavaş kalmakta, bunun sebebi JSON veri formatının yorumlanmasındaki gecikmedir. XML formatıyla yapılan çalışmalarda Java programlama dilindeki HttpClient çalışmaları yavaş kalmakta, diğer tüm alternatif çalışmalar birbiriyle yakın sonuçlar sergilemektedir. Tüm sonuçlar analiz edildiğinde özellikle veriler büyüdükçe SAP içerisinde web servisler ve geliştirici ekranlarıyla doğrudan desteklenen SOAP API üzerinden WSDL veri formatıyla Node.js ortamında axios veya easy-soap-request kütüphanesiyle çalışmak en performanslı tercih olacaktır.

KAYNAKLAR

Arslan, E., "Kurumsal Kaynak Planlaması (ERP) Microsoft Dynamics Ax Yazılımının Şirketlere Ve Kamu Kuruluşlarına Uyarlanması", İstanbul Aydın Üniversitesi Fen Bilimleri Enstitüsü Yüksek Lisans Tezi, 2015

Bal, B.M., "Kurumsal Kaynak Planlama Yazılımı Seçimi İçin Analitik Hiyerarşik Proses ve Apriori Algoritması Uygulanması", Hacettepe Üniversitesi Sosyal Bilimler Enstitüsü Yüksek Lisans Tezi, 2020

Başar, R., Arslan, H.M., "Kurumsal Kaynak Planlaması (ERP) Yazılımının En Uygun Uzlaşık Çözüm (Vikor) İle Seçimi", Süleyman Demirel Üniversitesi İktisadi ve İdari Bilimler Fakültesi Dergisi, Sayı. 22(4), Sayfa. 1065-1080, 2017

Bayraktar, E., Efe, M., "Kurumsal Kaynak Planlaması ERP Ve Yazılım Seçim Süreci", The Journal of Selcuk University Social Sciences Institute, Sayı. 15, Sayfa. 689 - 709, 2006

Boztaş, M., Özmızrak, M., "Kurumsal Kaynak Planlaması (ERP) Yazılımları Kurulum ve Kullanım Sürecinin Bilgi Yönetimi Kavramıyla Etkileşimi", İstanbul Commerce University Journal of Science, Sayı. 11(21), Sayfa. 65-79, 2012

Çakır, B.Ö., Bedük, A., "Çalışanların Kurumsal Kaynak Planlaması ERP Değerlendirmeleri ve Kurumsallaşma Algıları", The Journal of Selcuk University Social Sciences Institute, Sayı. 30, Sayfa. 81-91, 2013

Dikmen, F.C., Kavakcı, I., "Bütünleşik Olarak Kullanılan Macbeth, Topsis Ve Copras Yöntemleri İle Kurumsal Kaynak Planlama Yazılım Seçimi", Ağrı İbrahim Çeçen Üniversitesi Sosyal Bilimler Enstitüsü Dergisi, Sayı. 8(1), Sayfa. 205-238, 2022

Dülgerler, M., "Kurumsal Kaynak Planlaması Ve Web Servisleri İle Bir Erp Uygulaması", Beykent Üniversitesi Fen Bilimleri Enstitüsü Yüksek Lisans Tezi, 2007

Kılıçaslan, Ş., "Bir Kurumsal Kaynak Planlama Yazılımı Uygulaması ve Başarımının Değerlendirilmesi", Kocaeli Üniversitesi Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, 2012

Madenoğlu, F.S., "Bütünleşik Entropi-Copras Yaklaşımı İle Kurumsal Kaynak Planlama (KKP) Yazılımının Seçimi", Journal of Management and Economics Research, Sayı. 19(4), Sayfa. 14-29, 2021

Özkan Özen, Y.D., Koçak, A., "Bulanık Analitik Hiyerarşi ve Bulanık Dematel Yöntemleri Kullanılarak Kurumsal Kaynak Planlaması Yazılım Seçimi ve Değerlendirilmesi", Yönetim ve Ekonomi Dergisi, Sayı. 24(3), Sayfa. 929-957, 2017

Perçin, S., Gök, A.C., "ERP Yazılımı Seçiminde İki Aşamalı AAS-TOPSIS Yaklaşımı", Eskişehir Osmangazi University Journal of Economics and Administrative Sciences, Sayı. 8(2), Sayfa. 93-114, 2013

Tarhan, H.H., "Bir Kurumsal Kaynak Planlama Yazılımı Ve Akıllı Karar Destek Sistemi Araçlarının Geliştirilmesi", Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Doktora Tezi, 2017

Topbaş, E., “Yalın Kurumsal Kaynak Planlaması Yazılımının Geliştirilmesi Ve Kahramanmaraş'ta Yalın Üretim Yapan İşletmelerde Uygulanması”, Kahramanmaraş Sütçü İmam Üniversitesi Sosyal Bilimler Enstitüsü Yüksek Lisans Tezi, 2019

Turan, S.,“Kobi’lerin Kurumsal Kaynak Planlama Yazılımlarından Beklentileri ve Sektörel Bazda Yazılım Geliştirilmesi”, İstanbul Ticaret Üniversitesi Fen Bilimleri Enstitüsü Yüksek Lisans Tezi, 2011

Vatansever, K., Uluköy, M., “Kurumsal Kaynak Planlaması Sistemlerinin Bulanık AHP ve Bulanık MOORA Yöntemleriyle Seçimi: Üretim Sektöründe Bir Uygulama”, Manisa Celal Bayar Üniversitesi Sosyal Bilimler Dergisi, Sayı.11(2), Sayfa. 274-293, 2013

Yeşildağ, B., “Muğla İlinde Küçük Ve Orta Büyüklükteki İşletmelerde Kurumsal Kaynak Planlama (ERP) Yazılımları Kullanım Düzeyi Ve Verimliliğinin Araştırılması”, Muğla Üniversitesi Sosyal Bilimler Enstitüsü Yüksek Lisans Tezi, 2010

Yıldız, A., Yıldız, D., “Bulanık TOPSIS Yöntemiyle Kurumsal Kaynak Planlaması Yazılım Seçimi”, Business and Economics Research Journal, Sayı. 5(1), Sayfa. 87-106, 2014

EKLER

EK 1. Küçük Veriyle SOAP Uygulamaları

EK 1.1. İstek atılan SAP sunucusunun SOAP WSDL tanımlaması

```
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:wsoap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" xmlns:tns="urn:sap-
com:document:sap:soap:functions:mc-style" xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy" xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:n1="urn:sap-com:document:sap:rfc:functions"
targetNamespace="urn:sap-com:document:sap:soap:functions:mc-style">
    <wsdl:documentation>
        <sidl:sidl xmlns:sidl="http://www.sap.com/2007/03/sidl"/>
    </wsdl:documentation>
    <wsp:UsingPolicy wsdl:required="true"/>
    <wsp:Policy wsu:Id="BN__Z_BIND_BACKGROUND_JOBS">
        <wsp:ExactlyOne>
            <wsp:All>
                <sapattahnd:Enabled xmlns:sapattahnd="http://www.sap.com/710/features/attachment/">false</sapattahnd:Enabled>
                <saptrnbnd:OptimizedMimeSerialization
xmlns:saptrnbnd="http://schemas.xmlsoap.org/ws/2004/09/policy/optimizedmimeserialization" wsp:Optional="true"/>
                <wsaw:UsingAddressing xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl" wsp:Optional="true"/>
            </wsp:All>
            <wsp:All>
                <sapattahnd:Enabled xmlns:sapattahnd="http://www.sap.com/710/features/attachment/">false</sapattahnd:Enabled>
                <saptrnbnd:OptimizedXMLTransfer xmlns:saptrnbnd="http://www.sap.com/webas/710/soap/features/transportbinding/"
uri="http://xml.sap.com/2006/11/esi/esp/binxml" wsp:Optional="true"/>
                <wsaw:UsingAddressing xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl" wsp:Optional="true"/>
            </wsp:All>
        </wsp:ExactlyOne>
    </wsp:Policy>
    <wsp:Policy wsu:Id="BN__Z_BIND_BACKGROUND_JOBS_soap12">
        <wsp:ExactlyOne>
            <wsp:All>
                <sapattahnd:Enabled xmlns:sapattahnd="http://www.sap.com/710/features/attachment/">false</sapattahnd:Enabled>
                <saptrnbnd:OptimizedMimeSerialization
xmlns:saptrnbnd="http://schemas.xmlsoap.org/ws/2004/09/policy/optimizedmimeserialization" wsp:Optional="true"/>
                <wsaw:UsingAddressing xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl" wsp:Optional="true"/>
            </wsp:All>
            <wsp:All>
                <sapattahnd:Enabled xmlns:sapattahnd="http://www.sap.com/710/features/attachment/">false</sapattahnd:Enabled>
                <saptrnbnd:OptimizedXMLTransfer xmlns:saptrnbnd="http://www.sap.com/webas/710/soap/features/transportbinding/"
uri="http://xml.sap.com/2006/11/esi/esp/binxml" wsp:Optional="true"/>
                <wsaw:UsingAddressing xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl" wsp:Optional="true"/>
            </wsp:All>
        </wsp:ExactlyOne>
    </wsp:Policy>
    <wsp:Policy wsu:Id="IF__Z_WS_BACKGROUND_JOBS">
        <wsp:ExactlyOne>
            <wsp:All>
                <sapsession:Session xmlns:sapsession="http://www.sap.com/webas/630/soap/features/session/">
                    <sapsession:enableSession>false</sapsession:enableSession>
                </sapsession:Session>
                <sapcentraladmin:CentralAdministration
xmlns:sapcentraladmin="http://www.sap.com/webas/700/soap/features/CentralAdministration/" wsp:Optional="true">
                    <sapcentraladmin:BusinessApplicationID>2F6FDBF8D9621EED98EF79C6E9C6B348</sapcentraladmin:BusinessApplicationID>
                </sapcentraladmin:CentralAdministration>
            </wsp:All>
        </wsp:ExactlyOne>
```

```
    </wsp:Policy>
    <wsp:Policy wsu:Id="OP__ZFmBackgroundJobs">
        <wsp:ExactlyOne>
            <wsp:All>
                <saptrhnw05:required xmlns:saptrhnw05="http://www.sap.com/NW05/soap/features/transaction/">no</saptrhnw05:required>
                <sapcomhnd:enableCommit xmlns:sapcomhnd="http://www.sap.com/NW05/soap/features/commit/">false</sapcomhnd:enableCommit>
                <sapblock:enableBlocking
xmlns:sapblock="http://www.sap.com/NW05/soap/features/blocking/">true</sapblock:enableBlocking>
                <saprmnw05:enableWSRM xmlns:saprmnw05="http://www.sap.com/NW05/soap/features/wsrm/">false</saprmnw05:enableWSRM>
            </wsp:All>
        </wsp:ExactlyOne>
    </wsp:Policy>
    <wsdl:types>
        <xsd:schema attributeFormDefault="qualified" targetNamespace="urn:sap-com:document:sap:rfc:functions">
            <xsd:simpleType name="char1">
                <xsd:restriction base="xsd:string">
                    <xsd:maxLength value="1"/>
                </xsd:restriction>
            </xsd:simpleType>
            <xsd:simpleType name="char10">
                <xsd:restriction base="xsd:string">
                    <xsd:maxLength value="10"/>
                </xsd:restriction>
            </xsd:simpleType>
            <xsd:simpleType name="char12">
                <xsd:restriction base="xsd:string">
                    <xsd:maxLength value="12"/>
                </xsd:restriction>
            </xsd:simpleType>
            <xsd:simpleType name="char20">
                <xsd:restriction base="xsd:string">
                    <xsd:maxLength value="20"/>
                </xsd:restriction>
            </xsd:simpleType>
            <xsd:simpleType name="char200">
                <xsd:restriction base="xsd:string">
                    <xsd:maxLength value="200"/>
                </xsd:restriction>
            </xsd:simpleType>
            <xsd:simpleType name="char32">
                <xsd:restriction base="xsd:string">
                    <xsd:maxLength value="32"/>
                </xsd:restriction>
            </xsd:simpleType>
            <xsd:simpleType name="char8">
                <xsd:restriction base="xsd:string">
                    <xsd:maxLength value="8"/>
                </xsd:restriction>
            </xsd:simpleType>
            <xsd:simpleType name="clnt3">
                <xsd:restriction base="xsd:string">
                    <xsd:maxLength value="3"/>
                </xsd:restriction>
            </xsd:simpleType>
            <xsd:simpleType name="date10">
```

```
                <xsd:restriction base="xsd:string">
                    <xsd:maxLength value="10"/>
                    <xsd:pattern value="\d\d\d\d-\d\d-\d\d"/>
                </xsd:restriction>
            </xsd:simpleType>
            <xsd:simpleType name="time">
                <xsd:restriction base="xsd:time">
                    <xsd:pattern value="[0-9]{2}:[0-9]{2}:[0-9]{2}"/>
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:schema>
        <xsd:schema xmlns:n0="urn:sap-com:document:sap:rfc:functions" attributeFormDefault="qualified" targetNamespace="urn:sap-com:document:sap:soap:functions:mc-style">
            <xsd:import namespace="urn:sap-com:document:sap:rfc:functions"/>
            <xsd:complexType name="ZBackgroundjobs">
                <xsd:sequence>
                    <xsd:element name="Jobname" type="n0:char32"/>
                    <xsd:element name="Jobcount" type="n0:char8"/>
                    <xsd:element name="Joblog" type="n0:char20"/>
                    <xsd:element name="Authckman" type="n0:clnt3"/>
                    <xsd:element name="Authcknam" type="n0:char12"/>
                    <xsd:element name="Status" type="n0:char1"/>
                    <xsd:element name="Sdlstrtdt" type="n0:date10"/>
                    <xsd:element name="Sdlstrttm" type="n0:time"/>
                    <xsd:element name="Date" type="n0:char10"/>
                    <xsd:element name="Time" type="n0:char8"/>
                    <xsd:element name="Text" type="n0:char200"/>
                </xsd:sequence>
            </xsd:complexType>
            <xsd:complexType name="TableOfZBackgroundjobs">
                <xsd:sequence>
                    <xsd:element name="item" type="tns:ZBackgroundjobs" minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
            </xsd:complexType>
            <xsd:element name="ZFmBackgroundJobs">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="InputDate" type="n0:char10"/>
                        <xsd:element name="Output" type="tns:TableOfZBackgroundjobs" minOccurs="0"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="ZFmBackgroundJobsResponse">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="Output" type="tns:TableOfZBackgroundjobs" minOccurs="0"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
        </xsd:schema>
    </wsdl:types>
    <wsdl:message name="ZFmBackgroundJobs">
        <wsdl:part name="parameters" element="tns:ZFmBackgroundJobs"/>
    </wsdl:message>
    <wsdl:message name="ZFmBackgroundJobsResponse">
        <wsdl:part name="parameter" element="tns:ZFmBackgroundJobsResponse"/>
    </wsdl:message>
    <wsdl:portType name="Z_WS_BACKGROUND_JOBS">
        <wsdl:documentation>
            <sapdoc:sapdoc xmlns:sapdoc="urn:sap:esi:documentation">
                <sapdoc:docitem docURL="http://tez.aliburak.local:8000/sap/bc/esdt/docu/sd_text?sap-client=100&sd_name=Z_WS_BACKGROUND_JOBS"/>
            </sapdoc:sapdoc>
        </wsdl:documentation>
        <wsp:Policy>
            <wsp:PolicyReference URI="#IF__Z_WS_BACKGROUND_JOBS"/>
        </wsp:Policy>
        <wsdl:operation name="ZFmBackgroundJobs">
            <wsp:Policy>
                <wsp:PolicyReference URI="#OP__ZFmBackgroundJobs"/>
            </wsp:Policy>
            <wsdl:input message="tns:ZFmBackgroundJobs"/>
            <wsdl:output message="tns:ZFmBackgroundJobsResponse"/>
        </wsdl:operation>
    </wsdl:portType>
    <wsdl:binding name="Z_BIND_BACKGROUND_JOBS" type="tns:Z_WS_BACKGROUND_JOBS">
        <wsp:Policy>
            <wsp:PolicyReference URI="#BN__Z_BIND_BACKGROUND_JOBS"/>
        </wsp:Policy>
        <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
        <wsdl:operation name="ZFmBackgroundJobs">
            <soap:operation soapAction="urn:sap-com:document:sap:soap:functions:mc-style:Z_WS_BACKGROUND_JOBS:ZFmBackgroundJobsRequest" style="document"/>
```

EK 1.2. Küçük veri seti bulunan artalan işleri hata kayıtları için SAP sunucusuna atılan SOAP istek mektubu

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:sap-com:document:sap:soap:functions:mc-style">
  <soapenv:Header/>
  <soapenv:Body>
    <urn:ZFmBackgroundJobs>
      <InputDate>01012023</InputDate>
      <!--Optional:-->
      <Output>
      </Output>
    </urn:ZFmBackgroundJobs>
  </soapenv:Body>
</soapenv:Envelope>
```

EK 1.3. EK 1.2'de yer verilen istek mektubuna SAP sunucusunun cevap mektubu

```
<soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/">
       <soap-env:Header/>
       <soap-env:Body>
              <n0:ZFmBackgroundJobsResponse xmlns:n0="urn:sap-
com:document:sap:soap:functions:mc-style">
                     <Output>
                            <soap-env:Envelope xmlns:soap-
env="http://schemas.xmlsoap.org/soap/envelope/">
   <soap-env:Header/>
   <soap-env:Body>
      <n0:ZFmBackgroundJobsResponse xmlns:n0="urn:sap-com:document:sap:soap:functions:mc-
style">
         <Output>
                     <item>
                            <Jobname>EU_INIT</Jobname>
                            <Jobcount>00100000</Jobcount>
                            <Joblog>JOBLGX00100000X39277</Joblog>
                            <Authckman>100</Authckman>
                            <Authcknam>ALIBURAK</Authcknam>
                            <Status>A</Status>
                            <Sdlstrtdt>2022-12-09</Sdlstrtdt>
                            <Sdlstrttm>20:00:00</Sdlstrttm>
                            <Date>09.12.2022</Date>
                            <Time>20:00:00</Time>
                            <Text>Job canceled due to system shutdown.</Text>
                     </item>
                     <item>
                            <Jobname>EU_REORG</Jobname>
                            <Jobcount>01400100</Jobcount>
                            <Joblog>JOBLGX01400100X79070</Joblog>
                            <Authckman>100</Authckman>
                            <Authcknam>ALIBURAK</Authcknam>
                            <Status>A</Status>
                            <Sdlstrtdt>2022-12-10</Sdlstrtdt>
                            <Sdlstrttm>01:40:00</Sdlstrttm>
                            <Date>10.12.2022</Date>
                            <Time>01:40:00</Time>
                            <Text>Internal session terminated with a runtime error
DBSQL_TABLE_UNKNOWN (see ST22).Job canceled.</Text>
                     </item>
                     <item>
                            <Jobname>EU_REORG</Jobname>
                            <Jobcount>00475400</Jobcount>
                            <Joblog>JOBLGX00475400X35332</Joblog>
```

```
                        <Authckman>100</Authckman>
                        <Authcknam>ALIBURAK</Authcknam>
                        <Status>A</Status>
                        <Sdlstrtdt>2023-01-02</Sdlstrtdt>
                        <Sdlstrttm>01:40:00</Sdlstrttm>
                        <Date>02.01.2023</Date>
                        <Time>01:40:00</Time>
                        <Text>Internal session terminated with a runtime error
DBSQL_TABLE_UNKNOWN (see ST22).Job canceled.</Text>
                    </item>
                    <item>
                        <Jobname>EU_INIT</Jobname>
                        <Jobcount>00474900</Jobcount>
                        <Joblog>JOBLGX00474900X12732</Joblog>
                        <Authckman>100</Authckman>
                        <Authcknam>ALIBURAK</Authcknam>
                        <Status>A</Status>
                        <Sdlstrtdt>2023-01-02</Sdlstrtdt>
                        <Sdlstrttm>20:00:00</Sdlstrttm>
                        <Date>02.01.2023</Date>
                        <Time>20:00:00</Time>
                        <Text>Job canceled due to system shutdown.</Text>
                    </item>
                    <item>
                        <Jobname>EU_REORG</Jobname>
                        <Jobcount>13452600</Jobcount>
                        <Joblog>JOBLGX13452600X68526</Joblog>
                        <Authckman>100</Authckman>
                        <Authcknam>ALIBURAK</Authcknam>
                        <Status>A</Status>
                        <Sdlstrtdt>2023-01-04</Sdlstrtdt>
                        <Sdlstrttm>01:40:00</Sdlstrttm>
                        <Date>04.01.2023</Date>
                        <Time>01:40:00</Time>
                        <Text>Internal session terminated with a runtime error
DBSQL_TABLE_UNKNOWN (see ST22).Job canceled.</Text>
                    </item>
                    ...
                    ...örnekler sunucuya göre binlerce <item> olarak devam etmektedir...
                    ...
                    </Output>
            </n0:ZFmBackgroundJobsResponse>
        </soap-env:Body>
</soap-env:Envelope>
```

EK 1.4. C# programlama dilinde HttpClient sınıfı kullanarak SOAP isteği oluşturup, gönderen ve cevabı yorumlayan program kodu

```
//SOAP HttpClient
using System.Net.Http.Headers;
using System.Text;
using System.Diagnostics;

namespace MyApp
{
    public class Program
    {
        public static async Task Main()
        {
            Stopwatch stopwatch = new Stopwatch();
            stopwatch.Start();

            string xmlSOAP = """
                <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:sap-com:document:sap:soap:functions:mc-style">
                    <soapenv:Header/>
                    <soapenv:Body>
                        <urn:ZFmBackgroundJobs>
                            <InputDate>01012023</InputDate>
                            <!--Optional:-->
                            <Output>
                            </Output>
                        </urn:ZFmBackgroundJobs>
                    </soapenv:Body>
                </soapenv:Envelope>
                """;
            string url = "http://tez.aliburak.local:8000/sap/bc/srt/rfc/sap/z_ws_background_jobs/100/z_s_background_jobs/z_bind_background_jobs";
            //string soapAction = "urn:sap-com:document:sap:soap:functions:mc-style:Z_WS_BACKGROUND_JOBS:ZFmBackgroundJobsRequest";

            HttpClient httpClient = new HttpClient();
            HttpContent httpContent = new StringContent(xmlSOAP);
            HttpResponseMessage response;

            // Set the basic authentication credentials
            var byteArray = Encoding.ASCII.GetBytes("aliburak:Password1");
            httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Basic", Convert.ToBase64String(byteArray));

            HttpRequestMessage req = new HttpRequestMessage(HttpMethod.Post, url);
            req.Method = HttpMethod.Post;
            req.Content = httpContent;
            req.Content.Headers.ContentType = MediaTypeHeaderValue.Parse("text/xml; charset=utf-8");

            // Here you will get the Reponse from service
            response = await httpClient.SendAsync(req);

            // Converting the response into text format
            var content = await response.Content.ReadAsStringAsync();
            //Console.WriteLine(content);
            Console.WriteLine(content.Split("<item>").Length - 1);

            stopwatch.Stop();
            Console.WriteLine("Elapsed Milliseconds: " + stopwatch.ElapsedMilliseconds);
        }
    }
}
```

EK 1.5. C# programlama dilinde WebClient sınıfı kullanarak SOAP isteği oluşturup, gönderen ve cevabı yorumlayan program kodu

```
//SOAP WebClient
using System.Net;
using System.Diagnostics;

namespace MyApp
{
    public class Program
    {
        public static async Task Main()
        {
            Stopwatch stopwatch = new Stopwatch();
            stopwatch.Start();

            string xmlSOAP = """
                <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:sap-com:document:sap:soap:functions:mc-style">
                    <soapenv:Header/>
                    <soapenv:Body>
                        <urn:ZFmBackgroundJobs>
                            <InputDate>01012023</InputDate>
                            <!--Optional:-->
                            <Output>
                            </Output>
                        </urn:ZFmBackgroundJobs>
                    </soapenv:Body>
                </soapenv:Envelope>
                """;
            string url = "http://tez.aliburak.local:8000/sap/bc/srt/rfc/sap/z_ws_background_jobs/100/z_s_background_jobs/z_bind_background_jobs";

            var client = new WebClient();
            client.Headers.Add("Content-Type", "text/xml;charset=utf-8");
            //client.Headers.Add("SOAPAction", soapAction);
            client.Credentials = new NetworkCredential("aliburak", "Password1");
            var response = client.UploadString(url, xmlSOAP);

            //Console.WriteLine(response);
            Console.WriteLine(response.Split("<item>").Length - 1);

            stopwatch.Stop();
            Console.WriteLine("Elapsed Milliseconds: " + stopwatch.ElapsedMilliseconds);
        }
    }
}
```

EK 1.6. Java programlama dilinde HttpURLConnection sınıfı kullanarak SOAP isteği oluşturup, gönderen ve cevabı yorumlayan program kodu

```java
//SOAP HttpURLConnection
import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.Base64;
import java.util.concurrent.TimeUnit;
import com.google.common.base.Stopwatch;

public class Main5 {
    public static void main(String args[]) {
        try {
            Stopwatch stopwatch = Stopwatch.createStarted();

            String xmlSOAP =  """
                <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:sap-com:document:sap:soap:functions:mc-style">
                   <soapenv:Header/>
                   <soapenv:Body>
                      <urn:ZFmBackgroundJobs>
                         <InputDate>01012023</InputDate>
                         <!--Optional:-->
                         <Output>
                         </Output>
                      </urn:ZFmBackgroundJobs>
                   </soapenv:Body>
                </soapenv:Envelope>""";
            String url="http://tez.aliburak.local:8000/sap/bc/srt/rfc/sap/z_ws_background_jobs/100/z_s_background_jobs/z_bind_background_jobs";
            URL obj = new URL(url);
            HttpURLConnection con = (HttpURLConnection) obj.openConnection();

            con.setRequestMethod("POST");
            con.setRequestProperty("Content-Type","text/xml; charset=utf-8");
            con.setDoOutput(true);
            String valueToEncode = "aliburak" + ":" + "Password1";
            con.setRequestProperty("Authorization", "Basic " + Base64.getEncoder().encodeToString(valueToEncode.getBytes()));
            DataOutputStream wr = new DataOutputStream(con.getOutputStream());
            wr.writeBytes(xmlSOAP);
            wr.flush();
            wr.close();
            String responseStatus = con.getResponseMessage();
            System.out.println(responseStatus);
            BufferedReader in = new BufferedReader(new InputStreamReader(con.getInputStream()));
            String inputLine;
            StringBuffer response = new StringBuffer();
            while ((inputLine = in.readLine()) != null) {
                response.append(inputLine);
            }
            in.close();

            //print result
//            System.out.println(response.toString());
            System.out.println(response.toString().split("<item>").length - 1);

            stopwatch.stop();
            System.out.println("Time elapsed: "+ stopwatch.elapsed(TimeUnit.MILLISECONDS));
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

EK 1.7. Java programlama dilinde HttpClient sınıfı kullanarak SOAP isteği oluşturup, gönderen ve cevabı yorumlayan program kodu

```java
//SOAP HttpClient
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.Base64;
import java.util.concurrent.TimeUnit;
import com.google.common.base.Stopwatch;

public class Main6 {
    public static void main(String args[]) {
        try {
            Stopwatch stopwatch = Stopwatch.createStarted();

            String xmlSOAP =  """
                <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:sap-com:document:sap:soap:functions:mc-style">
                    <soapenv:Header/>
                    <soapenv:Body>
                        <urn:ZFmBackgroundJobs>
                            <InputDate>01012023</InputDate>
                            <!--Optional:-->
                            <Output>
                            </Output>
                        </urn:ZFmBackgroundJobs>
                    </soapenv:Body>
                </soapenv:Envelope>""";
            String url="http://tez.aliburak.local:8000/sap/bc/srt/rfc/sap/z_ws_background_jobs/100/z_s_background_jobs/z_bind_background_jobs";
            String soapAction = "urn:sap-com:document:sap:soap:functions:mc-style:Z_WS_BACKGROUND_JOBS:ZFmBackgroundJobsRequest";
            String valueToEncode = "aliburak" + ":" + "Password1";

            HttpClient client = HttpClient.newHttpClient();
            HttpRequest request = HttpRequest.newBuilder()
                    .POST(HttpRequest.BodyPublishers.ofString(xmlSOAP))
                    .version(HttpClient.Version.HTTP_1_1)
                    .uri(URI.create(url))
                    .header("Content-Type", "text/xml; charset=utf-8")
                    .header("SOAPAction", soapAction)
                    .header("Authorization", "Basic " + Base64.getEncoder().encodeToString(valueToEncode.getBytes()))
                    .build();
            HttpResponse<String> response = client.send(request, HttpResponse.BodyHandlers.ofString());

            //print result
//            System.out.println(response.statusCode());
//            System.out.println(response.body());
            System.out.println(response.body().split("<item>").length - 1);

            stopwatch.stop();
            System.out.println("Time elapsed: "+ stopwatch.elapsed(TimeUnit.MILLISECONDS));
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

EK 1.8. Node.js ortamında soap kütüphanesi kullanarak SOAP isteği oluşturup, gönderen ve cevabı yorumlayan program kodu

```
// SOAP + WSDL + soap
var soap = require('soap');

performance.mark('A');

var input = '01012023';
console.log("background-jobs Input: " + input);

const url = 'http://tez.aliburak.local:8000/sap/bc/srt/wsdl/flv_10002A111AD1/bndg_url/sap/bc/srt/rfc/sap/z_ws_background_jobs/100
    /z_s_background_jobs/z_bind_background_jobs?sap-client=100';
const uname = 'aliburak' ;
const pass = 'Password1' ;
const auth = "Basic " + new Buffer.from(uname + ":" + pass).toString("base64");

const foo = async () => {
    await soap.createClient(url, { wsdl_headers: {Authorization: auth}}, function(err, client) {
        try{
            client.setSecurity(new soap.BasicAuthSecurity(uname, pass));
            client.ZFmBackgroundJobs({InputDate: input, Output: ""}, function(err, result) {
                if (typeof result === 'undefined') {
                    // Başarılı connection sonrası VPN bağlantısı kesilirse response alınamaz ve undefined döner ama catch'e girmez
                    console.log("Connection failed");
                    // res.send(false);
                }else{
                    // console.log(result.Output);
                    console.log(Object.keys(result.Output.item).length);
                    console.log("Başarılı");
                    performance.mark('B');
                    performance.measure('A to B', 'A', 'B');
                    const measure = performance.getEntriesByName('A to B')[0];
                    console.log(measure.duration);
                }
            });
        }catch(err){
            console.log(err);
        }
    });
};

foo();
```

EK 1.9. Node.js ortamında strong-soap kütüphanesi kullanarak SOAP isteği oluşturup, gönderen ve cevabı yorumlayan program kodu

```
// SOAP + WSDL + strong-soap
const soap = require('strong-soap').soap;

performance.mark('A');

var input = '01012023';
console.log("background-jobs Input: " + input);

const url = 'http://tez.aliburak.local:8000/sap/bc/srt/wsdl/flv_10002A111AD1/bndg_url/sap/bc/srt/rfc/sap/z_ws_background_jobs/100
    /z_s_background_jobs/z_bind_background_jobs?sap-client=100';
const uname = 'aliburak' ;
const pass = 'Password1' ;
const auth = "Basic " + new Buffer.from(uname + ":" + pass).toString("base64");

const foo = async () => {
    await soap.createClient(url, { wsdl_headers: {Authorization: auth} }, function(err, client) {
        try{
            client.setSecurity(new soap.BasicAuthSecurity(uname, pass));
            client.ZFmBackgroundJobs({InputDate: input, Output: ""}, function(err, result) {
                if (typeof result === 'undefined') {
                    // Başarılı connection sonrası VPN bağlantısı kesilirse response alınamaz ve undefined döner ama catch'e girmez
                    console.log("Connection failed");
                    // res.send(false);
                }else{
                    // console.log(result.Output);
                    console.log(Object.keys(result.Output.item).length);
                    console.log("Başarılı");
                    performance.mark('B');
                    performance.measure('A to B', 'A', 'B');
                    const measure = performance.getEntriesByName('A to B')[0];
                    console.log(measure.duration);
                }
            });
        }catch(err){
            console.log(err);
        }
    });
};

foo();
```

EK 1.10. Node.js ortamında easy-soap-request kütüphanesi kullanarak SOAP isteği oluşturup, gönderen ve cevabı yorumlayan program kodu

```
// SOAP + WSDL + easy-soap-request
const soapRequest = require('easy-soap-request');

performance.mark('A');

const url = 'http://tez.aliburak.local:8000/sap/bc/srt/rfc/sap/z_ws_background_jobs/100/z_s_background_jobs/z_bind_background_jobs';
const uname = 'aliburak' ;
const pass = 'Password1' ;
const auth = "Basic " + new Buffer.from(uname + ":" + pass).toString("base64");

const Headers = {
    'Authorization' : auth,
    'Content-Type': 'text/xml;charset=UTF-8',
    'SOAPAction': 'urn:sap-com:document:sap:soap:functions:mc-style:Z_WS_BACKGROUND_JOBS:ZFmBackgroundJobsRequest'
};
const xml = `<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:sap-com:document:sap:soap:functions
    :mc-style">
                <soapenv:Header/>
                <soapenv:Body>
                    <urn:ZFmBackgroundJobs>
                        <InputDate>01012023</InputDate>
                        <!--Optional:-->
                        <Output>
                        </Output>
                    </urn:ZFmBackgroundJobs>
                </soapenv:Body>
            </soapenv:Envelope>`;

(async () => {
    const { response } = await soapRequest({ url: url, headers: Headers, xml: xml }); // Optional timeout parameter(milliseconds)
    // const { headers, body, statusCode } = response;
    // console.log(response);
    console.log(response.body.split("<item>").length - 1);
    console.log("Başarılı");
    performance.mark('B');
    performance.measure('A to B', 'A', 'B');
    const measure = performance.getEntriesByName('A to B')[0];
    console.log(measure.duration);
})();
```

EK 1.11. Node.js ortamında axios kütüphanesi kullanarak SOAP isteği oluşturup, gönderen ve cevabı yorumlayan program kodu

```
// SOAP + axios
var axios = require('axios');

performance.mark('A');

const url = 'http://tez.aliburak.local:8000/sap/bc/srt/rfc/sap/z_ws_background_jobs/100/z_s_background_jobs/z_bind_background_jobs';
const uname = 'aliburak' ;
const pass = 'Password1' ;
const auth = "Basic " + new Buffer.from(uname + ":" + pass).toString("base64");

const xml = `<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:sap-com:document:sap:soap:functions
    :mc-style">
                <soapenv:Header/>
                <soapenv:Body>
                    <urn:ZFmBackgroundJobs>
                        <InputDate>01012023</InputDate>
                        <!--Optional:-->
                        <Output>
                        </Output>
                    </urn:ZFmBackgroundJobs>
                </soapenv:Body>
            </soapenv:Envelope>`;

const foo = async () => {
    axios.post(url,
        xml,{
        headers : {
            'Authorization' : auth,
            'Content-Type': 'text/xml;charset=UTF-8',
            'SOAPAction': 'urn:sap-com:document:sap:soap:functions:mc-style:Z_WS_BACKGROUND_JOBS:ZFmBackgroundJobsRequest'
        }
    })
    .then(function (response) {
        // handle success
        // console.log(response);
        // console.log(response.data)
        console.log(response.data.split("<item>").length - 1);;
        console.log("Başarılı");
        performance.mark('B');
        performance.measure('A to B', 'A', 'B');
        const measure = performance.getEntriesByName('A to B')[0];
        console.log(measure.duration);
    })
    .catch(function (error) {
        // handle error
        console.log(error);
    })
    .then(function () {
        // always executed
    });
};

foo();
```

EK 1.12. Node.js ortamında request kütüphanesi kullanarak SOAP isteği oluşturup, gönderen ve cevabı yorumlayan program kodu

```
// SOAP + request
var request = require('request');

performance.mark('A');

const url = 'http://tez.aliburak.local:8000/sap/bc/srt/rfc/sap/z_ws_background_jobs/100/z_s_background_jobs/z_bind_background_jobs';
const uname = 'aliburak' ;
const pass = 'Password1' ;
const auth = "Basic " + new Buffer.from(uname + ":" + pass).toString("base64");

const xml = `<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:sap-com:document:sap:soap:functions
    :mc-style">
                <soapenv:Header/>
                <soapenv:Body>
                    <urn:ZFmBackgroundJobs>
                        <InputDate>01012023</InputDate>
                        <!--Optional:-->
                        <Output>
                        </Output>
                    </urn:ZFmBackgroundJobs>
                </soapenv:Body>
            </soapenv:Envelope>`;

const foo = async () => {
    request.post( {
        url : url,
        body: xml,
        headers : {
            'Authorization' : auth,
            'Content-Type': 'text/xml;charset=UTF-8',
            'SOAPAction': 'urn:sap-com:document:sap:soap:functions:mc-style:Z_WS_BACKGROUND_JOBS:ZFmBackgroundJobsRequest'
        }
      }, function(error, response, body) {
        //   console.log(response);
        //   console.log(body);
          console.log(body.split("<item>").length - 1);
          console.log("Başarılı");
          performance.mark('B');
          performance.measure('A to B', 'A', 'B');
          const measure = performance.getEntriesByName('A to B')[0];
          console.log(measure.duration);
      } );
};

foo();
```

EK 2. Küçük Veriyle REST JSON Uygulamaları

EK 2.1. Küçük veri seti bulunan artalan işleri hata kayıtları için SAP sunucusuna atılan REST isteğine SAP sunucusunun JSON formatındaki cevabı


```
{
    "ITEM": [
        {
            "JOBNAME": "EU_REORG",
            "JOBCOUNT": "14253200",
            "JOBLOG": "JOBLGX14253200X06010",
            "AUTHCKMAN": "100",
            "AUTHCKNAM": "ALIBURAK",
            "STATUS": "A",
```

```
            "SDLSTRTDT": "2023-01-05",
            "SDLSTRTTM": "01:40:00",
            "DATE": "05.01.2023",
            "TIME": "01:40:00",
            "TEXT": "Internal session terminated with a runtime error DBSQL_TABLE_UNKNOWN
(see ST22).\nJob canceled.\n"
        },
        {
            "JOBNAME": "EU_REORG",
            "JOBCOUNT": "23302700",
            "JOBLOG": "JOBLGX23302700X48590",
            "AUTHCKMAN": "100",
            "AUTHCKNAM": "ALIBURAK",
            "STATUS": "A",
            "SDLSTRTDT": "2023-01-11",
            "SDLSTRTTM": "01:40:00",
            "DATE": "11.01.2023",
            "TIME": "01:40:00",
            "TEXT": "Internal session terminated with a runtime error DBSQL_TABLE_UNKNOWN
(see ST22).\nJob canceled.\n"
        },
        {
            "JOBNAME": "EU_INIT",
            "JOBCOUNT": "23302500",
            "JOBLOG": "JOBLGX23302500X79799",
            "AUTHCKMAN": "100",
            "AUTHCKNAM": "ALIBURAK",
            "STATUS": "A",
            "SDLSTRTDT": "2023-01-11",
            "SDLSTRTTM": "20:00:00",
            "DATE": "11.01.2023",
            "TIME": "20:00:00",
            "TEXT": "Job canceled due to system shutdown.\n"
        },
        {
            "JOBNAME": "EU_REORG",
            "JOBCOUNT": "19302600",
            "JOBLOG": "JOBLGX19302600X71003",
            "AUTHCKMAN": "100",
            "AUTHCKNAM": "ALIBURAK",
            "STATUS": "A",
            "SDLSTRTDT": "2023-01-12",
            "SDLSTRTTM": "01:40:00",
            "DATE": "12.01.2023",
            "TIME": "01:40:00",
            "TEXT": "Internal session terminated with a runtime error DBSQL_TABLE_UNKNOWN
(see ST22).\nJob canceled.\n"
        },
        {
            "JOBNAME": "EU_INIT",
            "JOBCOUNT": "18214800",
            "JOBLOG": "JOBLGX18214800X04855",
            "AUTHCKMAN": "100",
            "AUTHCKNAM": "ALIBURAK",
            "STATUS": "A",
            "SDLSTRTDT": "2023-01-12",
            "SDLSTRTTM": "20:00:00",
            "DATE": "12.01.2023",
            "TIME": "20:00:00",
            "TEXT": ""
        },
        {
            "JOBNAME": "EU_REORG",
            "JOBCOUNT": "18215000",
            "JOBLOG": "JOBLGX18215000X54295",
            "AUTHCKMAN": "100",
```

```
            "AUTHCKNAM": "ALIBURAK",
            "STATUS": "A",
            "SDLSTRTDT": "2023-01-13",
            "SDLSTRTTM": "01:40:00",
            "DATE": "13.01.2023",
            "TIME": "01:40:00",
            "TEXT": "Internal session terminated with a runtime error DBSQL_TABLE_UNKNOWN
(see ST22).\nJob canceled.\n"
        },
              ...
              ...örnekler sunucuya göre binlerce tekrar devam etmektedir...
              ...
    ]
}
```

EK 2.2. C# programlama dilinde HttpClient sınıfı kullanarak REST isteği oluşturup, gönderen ve JSON formatındaki cevabı yorumlayan program kodu

```
1  //REST+JSON HttpClient
2  using System.Net.Http.Headers;
3  using System.Text;
4  using System.Diagnostics;
5  using Newtonsoft.Json.Linq;
6
7  namespace MyApp
8  {
9      public class Program
10     {
11         public static void Main()
12         {
13             Stopwatch stopwatch = new Stopwatch();
14             stopwatch.Start();
15
16             // Create an HttpClient with a base address
17             var client = new HttpClient { BaseAddress = new Uri("http://tez.aliburak.local:8000/sap/bc/zrest_json_sm37?DATE=01012023") };
18
19             // Set the basic authentication credentials
20             var byteArray = Encoding.ASCII.GetBytes("aliburak:Password1");
21             client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Basic", Convert.ToBase64String(byteArray));
22
23             // Make the GET request
24             var response = client.GetAsync("").Result;
25
26             // Read the response content
27             var content = response.Content.ReadAsStringAsync().Result;
28
29             // Print result
30             //Console.WriteLine(content);
31             JObject jObject = JObject.Parse(content);
32             JArray jArray = (JArray)jObject["ITEM"];
33             Console.WriteLine("Array length: " + jArray.Count);
34
35             stopwatch.Stop();
36             Console.WriteLine("Elapsed Milliseconds: " + stopwatch.ElapsedMilliseconds);
37         }
38     }
39 }
```

EK 2.3. C# programlama dilinde RestSharp sınıfı kullanarak REST isteği oluşturup, gönderen ve JSON formatındaki cevabı yorumlayan program kodu

```
//REST+JSON RestSharp
using Newtonsoft.Json.Linq;
using RestSharp;
using System.Diagnostics;
using System.Text;

namespace MyApp
{
    public class Program
    {
        public static void Main()
        {
            Stopwatch stopwatch = new Stopwatch();
            stopwatch.Start();

            var client = new RestClient("http://tez.aliburak.local:8000");
            var request = new RestRequest("/sap/bc/zrest_json_sm37?DATE=01012023", Method.Get);
            string credentials = Convert.ToBase64String(Encoding.ASCII.GetBytes("aliburak:Password1"));
            request.AddHeader("Authorization", "Basic " + credentials);
            RestResponse response = client.Execute(request);
            var content = response.Content;

            // Print result
            //Console.WriteLine(content);
            JObject jObject = JObject.Parse(content);
            JArray jArray = (JArray)jObject["ITEM"];
            Console.WriteLine("Array length: " + jArray.Count);

            stopwatch.Stop();
            Console.WriteLine("Elapsed Milliseconds: " + stopwatch.ElapsedMilliseconds);
        }
    }
}
```

EK 2.4. Java programlama dilinde HttpURLConnection sınıfı kullanarak REST isteği oluşturup, gönderen ve JSON formatındaki cevabı yorumlayan program kodu

```java
//REST+JSON HttpURLConnection
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.Base64;
import java.util.concurrent.TimeUnit;
import com.google.common.base.Stopwatch;
import org.json.JSONArray;
import org.json.JSONObject;

public class Main2 {
    public static void main(String[] args) {
        try {
            Stopwatch stopwatch = Stopwatch.createStarted();

            String url = "http://tez.aliburak.local:8000/sap/bc/zrest_json_sm37?DATE=01012023";
            URL obj = new URL(url);
            HttpURLConnection con = (HttpURLConnection) obj.openConnection();

            // optional default is GET
            con.setRequestMethod("GET");
            // con.setRequestProperty("User-Agent", "Mozilla/5.0 (Windows NT 6.3; Win64; x64; rv:27.0) Gecko/20100101 Firefox/27.0.2
                Waterfox/27.0");
            // con.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");
            String valueToEncode = "aliburak" + ":" + "Password1";
            con.setRequestProperty("Authorization", "Basic " + Base64.getEncoder().encodeToString(valueToEncode.getBytes()));
            // int responseCode = con.getResponseCode();
            // System.out.println("\nSending 'GET' request to URL : " + url);
            // System.out.println("Response Code : " + responseCode);
            BufferedReader in = new BufferedReader(new InputStreamReader(con.getInputStream()));
            String inputLine;
            StringBuffer response = new StringBuffer();
            while ((inputLine = in.readLine()) != null) {
                response.append(inputLine);
            }
            in.close();

            //print result
//            System.out.println(response.toString());
            JSONObject jObject = new JSONObject(response.toString());
            JSONArray jArray = jObject.getJSONArray("ITEM");
            System.out.println("Array length: "+jArray.length());

            stopwatch.stop();
            System.out.println("Time elapsed: "+ stopwatch.elapsed(TimeUnit.MILLISECONDS));
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

EK 2.5. Java programlama dilinde HttpClient sınıfı kullanarak REST isteği oluşturup, gönderen ve JSON formatındaki cevabı yorumlayan program kodu

```java
//REST+JSON HttpClient
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.Base64;
import java.util.concurrent.TimeUnit;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.net.URI;
import com.google.common.base.Stopwatch;
import org.json.JSONArray;
import org.json.JSONObject;

public class Main4 {
    public static void main(String[] args) {
        try {
            Stopwatch stopwatch = Stopwatch.createStarted();

            String valueToEncode = "aliburak" + ":" + "Password1";

            HttpClient client = HttpClient.newHttpClient();
            HttpRequest request = HttpRequest.newBuilder()
                    .GET()
                    .version(HttpClient.Version.HTTP_1_1)
                    .uri(URI.create("http://tez.aliburak.local:8000/sap/bc/zrest_json_sm37?DATE=01012023"))
                    .header("Authorization", "Basic " + Base64.getEncoder().encodeToString(valueToEncode.getBytes()))
                    .build();
            HttpResponse<String> response = client.send(request, HttpResponse.BodyHandlers.ofString());

            //print result
//            System.out.println(response.statusCode());
//            System.out.println(response.body());
            JSONObject jObject = new JSONObject(response.body());
            JSONArray jArray = jObject.getJSONArray("ITEM");
            System.out.println("Array length: "+jArray.length());

            stopwatch.stop();
            System.out.println("Time elapsed: "+ stopwatch.elapsed(TimeUnit.MILLISECONDS));
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

EK 2.6. Node.js ortamında request kütüphanesi kullanarak REST isteği oluşturup, gönderen ve JSON formatındaki cevabı yorumlayan program kodu

```
// REST + JSON + request
var request = require('request');

performance.mark('A');

var input = '01012023';
console.log("background-jobs Input: " + input);

const url = "http://tez.aliburak.local:8000/sap/bc/zrest_json_sm37?DATE="+input;
const uname = 'aliburak' ;
const pass = 'Password1' ;
const auth = "Basic " + new Buffer.from(uname + ":" + pass).toString("base64");

const foo = async () => {
    request.get( {
        url : url,
        headers : {
            "Authorization" : auth
        }
      }, function(error, response, body) {
        //   console.log(JSON.parse(body).ITEM);
          console.log(JSON.parse(body).ITEM.length);
          console.log("Başarılı");
          performance.mark('B');
          performance.measure('A to B', 'A', 'B');
          const measure = performance.getEntriesByName('A to B')[0];
          console.log(measure.duration);
      } );
};

foo();
```

EK 2.7. Node.js ortamında axios kütüphanesi kullanarak REST isteği oluşturup, gönderen ve JSON formatındaki cevabı yorumlayan program kodu

```
// REST + JSON + axios
var axios = require('axios');

performance.mark('A');

var input = '01012023';
console.log("background-jobs Input: " + input);

const url = "http://tez.aliburak.local:8000/sap/bc/zrest_json_sm37?DATE="+input;
const uname = 'aliburak' ;
const pass = 'Password1' ;
const auth = "Basic " + new Buffer.from(uname + ":" + pass).toString("base64");

const foo = async () => {
    axios.get(url,{
        headers : {
            "Authorization" : auth
        }
    })
    .then(function (response) {
        // handle success
        // console.log(response.data);
        console.log(response.data.ITEM.length);
        console.log("Başarılı");
        performance.mark('B');
        performance.measure('A to B', 'A', 'B');
        const measure = performance.getEntriesByName('A to B')[0];
        console.log(measure.duration);
    })
    .catch(function (error) {
        // handle error
        console.log(error);
    })
};

foo();
```

EK 3. Küçük Veriyle REST XML Uygulamaları

EK 3.1. Küçük veri seti bulunan artalan işleri hata kayıtları için SAP sunucusuna atılan REST isteğine SAP sunucusunun XML formatındaki cevabı

```
<asx:abap xmlns:asx="http://www.sap.com/abapxml" version="1.0">
       <asx:values>
              <TAB>
                     <Z_BACKGROUNDJOBS>
                            <JOBNAME>EU_REORG</JOBNAME>
                            <JOBCOUNT>14253200</JOBCOUNT>
                            <JOBLOG>JOBLGX14253200X06010</JOBLOG>
                            <AUTHCKMAN>100</AUTHCKMAN>
                            <AUTHCKNAM>ALIBURAK</AUTHCKNAM>
                            <STATUS>A</STATUS>
                            <SDLSTRTDT>2023-01-05</SDLSTRTDT>
                            <SDLSTRTTM>01:40:00</SDLSTRTTM>
                            <DATE>05.01.2023</DATE>
                            <TIME>01:40:00</TIME>
                            <TEXT>Internal session terminated with a runtime error
DBSQL_TABLE_UNKNOWN (see ST22). Job canceled. </TEXT>
                     </Z_BACKGROUNDJOBS>
                     <Z_BACKGROUNDJOBS>
                            <JOBNAME>EU_REORG</JOBNAME>
                            <JOBCOUNT>23302700</JOBCOUNT>
                            <JOBLOG>JOBLGX23302700X48590</JOBLOG>
                            <AUTHCKMAN>100</AUTHCKMAN>
                            <AUTHCKNAM>ALIBURAK</AUTHCKNAM>
                            <STATUS>A</STATUS>
                            <SDLSTRTDT>2023-01-11</SDLSTRTDT>
                            <SDLSTRTTM>01:40:00</SDLSTRTTM>
                            <DATE>11.01.2023</DATE>
                            <TIME>01:40:00</TIME>
                            <TEXT>Internal session terminated with a runtime error
DBSQL_TABLE_UNKNOWN (see ST22). Job canceled. </TEXT>
                     </Z_BACKGROUNDJOBS>
                     <Z_BACKGROUNDJOBS>
                            <JOBNAME>EU_INIT</JOBNAME>
                            <JOBCOUNT>23302500</JOBCOUNT>
                            <JOBLOG>JOBLGX23302500X79799</JOBLOG>
                            <AUTHCKMAN>100</AUTHCKMAN>
                            <AUTHCKNAM>ALIBURAK</AUTHCKNAM>
                            <STATUS>A</STATUS>
                            <SDLSTRTDT>2023-01-11</SDLSTRTDT>
                            <SDLSTRTTM>20:00:00</SDLSTRTTM>
                            <DATE>11.01.2023</DATE>
                            <TIME>20:00:00</TIME>
                            <TEXT>Job canceled due to system shutdown. </TEXT>
                     </Z_BACKGROUNDJOBS>
                     <Z_BACKGROUNDJOBS>
                            <JOBNAME>EU_REORG</JOBNAME>
                            <JOBCOUNT>19302600</JOBCOUNT>
                            <JOBLOG>JOBLGX19302600X71003</JOBLOG>
                            <AUTHCKMAN>100</AUTHCKMAN>
                            <AUTHCKNAM>ALIBURAK</AUTHCKNAM>
                            <STATUS>A</STATUS>
                            <SDLSTRTDT>2023-01-12</SDLSTRTDT>
                            <SDLSTRTTM>01:40:00</SDLSTRTTM>
                            <DATE>12.01.2023</DATE>
                            <TIME>01:40:00</TIME>
                            <TEXT>Internal session terminated with a runtime error
DBSQL_TABLE_UNKNOWN (see ST22). Job canceled. </TEXT>
                     </Z_BACKGROUNDJOBS>
```

```
                        <Z_BACKGROUNDJOBS>
                                <JOBNAME>EU_INIT</JOBNAME>
                                <JOBCOUNT>18214800</JOBCOUNT>
                                <JOBLOG>JOBLGX18214800X04855</JOBLOG>
                                <AUTHCKMAN>100</AUTHCKMAN>
                                <AUTHCKNAM>ALIBURAK</AUTHCKNAM>
                                <STATUS>A</STATUS>
                                <SDLSTRTDT>2023-01-12</SDLSTRTDT>
                                <SDLSTRTTM>20:00:00</SDLSTRTTM>
                                <DATE>12.01.2023</DATE>
                                <TIME>20:00:00</TIME>
                                <TEXT/>
                        </Z_BACKGROUNDJOBS>
                        <Z_BACKGROUNDJOBS>
                                <JOBNAME>EU_REORG</JOBNAME>
                                <JOBCOUNT>18215000</JOBCOUNT>
                                <JOBLOG>JOBLGX18215000X54295</JOBLOG>
                                <AUTHCKMAN>100</AUTHCKMAN>
                                <AUTHCKNAM>ALIBURAK</AUTHCKNAM>
                                <STATUS>A</STATUS>
                                <SDLSTRTDT>2023-01-13</SDLSTRTDT>
                                <SDLSTRTTM>01:40:00</SDLSTRTTM>
                                <DATE>13.01.2023</DATE>
                                <TIME>01:40:00</TIME>
                                <TEXT>Internal session terminated with a runtime error
DBSQL_TABLE_UNKNOWN (see ST22). Job canceled. </TEXT>
                        </Z_BACKGROUNDJOBS>
                        <Z_BACKGROUNDJOBS>
                                <JOBNAME>EU_INIT</JOBNAME>
                                <JOBCOUNT>00100100</JOBCOUNT>
                                <JOBLOG>JOBLGX00100100X61056</JOBLOG>
                                <AUTHCKMAN>100</AUTHCKMAN>
                                <AUTHCKNAM>ALIBURAK</AUTHCKNAM>
                                <STATUS>A</STATUS>
                                <SDLSTRTDT>2023-01-13</SDLSTRTDT>
                                <SDLSTRTTM>20:00:00</SDLSTRTTM>
                                <DATE>13.01.2023</DATE>
                                <TIME>20:00:00</TIME>
                                <TEXT>Job canceled due to system shutdown. </TEXT>
                        </Z_BACKGROUNDJOBS>
                        <Z_BACKGROUNDJOBS>
                                <JOBNAME>EU_REORG</JOBNAME>
                                <JOBCOUNT>01400000</JOBCOUNT>
                                <JOBLOG>JOBLGX01400000X55169</JOBLOG>
                                <AUTHCKMAN>100</AUTHCKMAN>
                                <AUTHCKNAM>ALIBURAK</AUTHCKNAM>
                                <STATUS>A</STATUS>
                                <SDLSTRTDT>2023-01-14</SDLSTRTDT>
                                <SDLSTRTTM>01:40:00</SDLSTRTTM>
                                <DATE>14.01.2023</DATE>
                                <TIME>01:40:00</TIME>
                                <TEXT>Internal session terminated with a runtime error
DBSQL_TABLE_UNKNOWN (see ST22). Job canceled. </TEXT>
                        </Z_BACKGROUNDJOBS>
                        <Z_BACKGROUNDJOBS>
                                <JOBNAME>EU_INIT</JOBNAME>
                                <JOBCOUNT>01361700</JOBCOUNT>
                                <JOBLOG>JOBLGX01361700X07721</JOBLOG>
                                <AUTHCKMAN>100</AUTHCKMAN>
                                <AUTHCKNAM>ALIBURAK</AUTHCKNAM>
                                <STATUS>A</STATUS>
                                <SDLSTRTDT>2023-01-14</SDLSTRTDT>
                                <SDLSTRTTM>20:00:00</SDLSTRTTM>
                                <DATE>14.01.2023</DATE>
                                <TIME>20:00:00</TIME>
                                <TEXT>Job canceled due to system shutdown. </TEXT>
```

```
                    </Z_BACKGROUNDJOBS>
        ...
        ...örnekler sunucuya göre binlerce <Z_BACKGROUNDJOBS> olarak devam etmektedir...
        ...
               </TAB>
        </asx:values>
</asx:abap>
```

EK 3.2. C# programlama dilinde HttpClient sınıfı kullanarak REST isteği oluşturup, gönderen ve XML formatındaki cevabı yorumlayan program kodu

```
1  //REST+XML HttpClient
2  using System.Net.Http.Headers;
3  using System.Text;
4  using System.Diagnostics;
5
6  namespace MyApp
7  {
8      public class Program
9      {
10         public static void Main()
11         {
12             Stopwatch stopwatch = new Stopwatch();
13             stopwatch.Start();
14
15             // Create an HttpClient with a base address
16             var client = new HttpClient { BaseAddress = new Uri("http://tez.aliburak.local:8000/sap/bc/zrest_xml_sm37/01012023") };
17
18             // Set the basic authentication credentials
19             var byteArray = Encoding.ASCII.GetBytes("aliburak:Password1");
20             client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Basic", Convert.ToBase64String(byteArray));
21
22             // Make the GET request
23             var response = client.GetAsync("").Result;
24
25             // Read the response content
26             var content = response.Content.ReadAsStringAsync().Result;
27
28             // Print result
29             //Console.WriteLine(content);
30             Console.WriteLine(content.Split("<ZBACKGROUNDJOBS>").Length - 1);
31
32             stopwatch.Stop();
33             Console.WriteLine("Elapsed Milliseconds: " + stopwatch.ElapsedMilliseconds);
34         }
35     }
36 }
```

EK 3.3. C# programlama dilinde RestSharp sınıfı kullanarak REST isteği oluşturup, gönderen ve XML formatındaki cevabı yorumlayan program kodu

```
//REST+XML RestSharp
using RestSharp;
using System.Diagnostics;
using System.Text;

namespace MyApp
{
    public class Program
    {
        public static void Main()
        {
            Stopwatch stopwatch = new Stopwatch();
            stopwatch.Start();

            var client = new RestClient("http://tez.aliburak.local:8000");
            var request = new RestRequest("/sap/bc/zrest_xml_sm37/01012023", Method.Get);
            string credentials = Convert.ToBase64String(Encoding.ASCII.GetBytes("aliburak:Password1"));
            request.AddHeader("Authorization", "Basic " + credentials);
            RestResponse response = client.Execute(request);
            var content = response.Content;

            // Print result
            //Console.WriteLine(content);
            Console.WriteLine(content.Split("<ZBACKGROUNDJOBS>").Length - 1);

            stopwatch.Stop();
            Console.WriteLine("Elapsed Milliseconds: " + stopwatch.ElapsedMilliseconds);
        }
    }
}
```

EK 3.4. Java programlama dilinde HttpURLConnection sınıfı kullanarak REST isteği oluşturup, gönderen ve XML formatındaki cevabı yorumlayan program kodu

```java
//REST+XML HttpURLConnection
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.Base64;
import java.util.concurrent.TimeUnit;
import com.google.common.base.Stopwatch;

public class Main {
    public static void main(String[] args) {
        try {
            Stopwatch stopwatch = Stopwatch.createStarted();

            String url = "http://tez.aliburak.local:8000/sap/bc/zrest_xml_sm37/01012023";
            URL obj = new URL(url);
            HttpURLConnection con = (HttpURLConnection) obj.openConnection();

            // optional default is GET
            con.setRequestMethod("GET");
            // con.setRequestProperty("User-Agent", "Mozilla/5.0 (Windows NT 6.3; Win64; x64; rv:27.0) Gecko/20100101 Firefox/27.0.2 Waterfox
                /27.0");
            // con.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");
            String valueToEncode = "aliburak" + ":" + "Password1";
            con.setRequestProperty("Authorization", "Basic " + Base64.getEncoder().encodeToString(valueToEncode.getBytes()));
            // int responseCode = con.getResponseCode();
            // System.out.println("\nSending 'GET' request to URL : " + url);
            // System.out.println("Response Code : " + responseCode);
            BufferedReader in = new BufferedReader(new InputStreamReader(con.getInputStream()));
            String inputLine;
            StringBuffer response = new StringBuffer();
            while ((inputLine = in.readLine()) != null) {
                response.append(inputLine);
            }
            in.close();

            //print result
//            System.out.println(response.toString());
            System.out.println(response.toString().split("<ZBACKGROUNDJOBS>").length - 1);

            stopwatch.stop();
            System.out.println("Time elapsed: "+ stopwatch.elapsed(TimeUnit.MILLISECONDS));
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

EK 3.5. Java programlama dilinde HttpClient sınıfı kullanarak REST isteği oluşturup, gönderen ve XML formatındaki cevabı yorumlayan program kodu

```
//REST+XML HttpClient
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.Base64;
import java.util.concurrent.TimeUnit;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.net.URI;
import com.google.common.base.Stopwatch;

public class Main3 {
    public static void main(String[] args) {
        try {
            Stopwatch stopwatch = Stopwatch.createStarted();

            String valueToEncode = "aliburak" + ":" + "Password1";

            HttpClient client = HttpClient.newHttpClient();
            HttpRequest request = HttpRequest.newBuilder()
                    .GET()
                    .version(HttpClient.Version.HTTP_1_1)
                    .uri(URI.create("http://tez.aliburak.local:8000/sap/bc/zrest_xml_sm37/01012023"))
                    .header("Authorization", "Basic " + Base64.getEncoder().encodeToString(valueToEncode.getBytes()))
                    .build();
            HttpResponse<String> response = client.send(request, HttpResponse.BodyHandlers.ofString());

            //print result
//            System.out.println(response.statusCode());
//            System.out.println(response.body());
            System.out.println(response.body().split("<BACKGROUNDJOBS>").length - 1);

            stopwatch.stop();
            System.out.println("Time elapsed: "+ stopwatch.elapsed(TimeUnit.MILLISECONDS));
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

EK 3.6. Node.js ortamında request kütüphanesi kullanarak REST isteği oluşturup, gönderen ve XML formatındaki cevabı yorumlayan program kodu

```
// REST + XML + request
var request = require('request');

performance.mark('A');

var input = '01012023';
console.log("background-jobs Input: " + input);

const url = "http://tez.aliburak.local:8000/sap/bc/zrest_xml_sm37/"+input;
const uname = 'aliburak' ;
const pass = 'Password1' ;
const auth = "Basic " + new Buffer.from(uname + ":" + pass).toString("base64");

const foo = async () => {
    request.get( {
        url : url,
        headers : {
            "Authorization" : auth
        }
      }, function(error, response, body) {
        //   console.log(response);
        //   console.log(body);
          console.log(body.split("<ZBACKGROUNDJOBS>").length - 1);
          console.log("Başarılı");
          performance.mark('B');
          performance.measure('A to B', 'A', 'B');
          const measure = performance.getEntriesByName('A to B')[0];
          console.log(measure.duration);
      } );
};

foo();
```

EK 3.7. Node.js ortamında axios kütüphanesi kullanarak REST isteği oluşturup, gönderen ve XML formatındaki cevabı yorumlayan program kodu

```
// REST + XML + axios
var axios = require('axios');

performance.mark('A');

var input = '01012023';
console.log("background-jobs Input: " + input);

const url = "http://tez.aliburak.local:8000/sap/bc/zrest_xml_sm37/"+input;
const uname = 'aliburak' ;
const pass = 'Password1' ;
const auth = "Basic " + new Buffer.from(uname + ":" + pass).toString("base64");

const foo = async () => {
    axios.get(url,{
        headers : {
            "Authorization" : auth
        }
    })
    .then(function (response) {
        // handle success
        // console.log(response);
        // console.log(response.data)
        console.log(response.data.split("<ZBACKGROUNDJOBS>").length - 1);;
        console.log("Başarılı");
        performance.mark('B');
        performance.measure('A to B', 'A', 'B');
        const measure = performance.getEntriesByName('A to B')[0];
        console.log(measure.duration);
    })
    .catch(function (error) {
        // handle error
        console.log(error);
    })
};

foo();
```

EK 4. SAP ABAP Uygulamaları

EK 4.1. Girilen tarihten itibaren artalan işleri için hata kayıtlarını getiren program kodu

```
FUNCTION Z_FM_BACKGROUND_JOBS.
*"----------------------------------------------------------------------
*"*"Local Interface:
*"  IMPORTING
*"     VALUE(INPUT_DATE) TYPE  CHAR10
*"  TABLES
*"      OUTPUT STRUCTURE  ZBACKGROUNDJOBS OPTIONAL
*"----------------------------------------------------------------------

DATA: TAB     TYPE c VALUE cl_abap_char_utilities=>horizontal_tab,
      NEWLINE TYPE c VALUE cl_abap_char_utilities=>NEWLINE.

DATA: BEGIN OF itab,
    JOBNAME   TYPE TBTCO-JOBNAME,
    JOBCOUNT  TYPE TBTCO-JOBCOUNT,
    JOBLOG    TYPE TBTCO-JOBLOG,    "TBTCO tablosunda
    AUTHCKMAN TYPE TBTCO-AUTHCKMAN,"TBTCO tablosunda
    AUTHCKNAM TYPE TBTCP-AUTHCKNAM,
    STATUS    TYPE TBTCO-STATUS,
    SDLSTRTDT TYPE TBTCO-SDLSTRTDT,
    SDLSTRTTM TYPE TBTCO-SDLSTRTTM,
END OF itab,
it_ma LIKE TABLE OF itab WITH HEADER LINE.

DATA: it_line LIKE LINE OF it_ma.

DATA: it_joblogtbl  TYPE STANDARD TABLE OF TBTC5,  "TABLES PARAM
      wa_joblogtbl  LIKE LINE OF it_joblogtbl,
      ld_client      TYPE tbtco-authckman,  "TBTCO tablosunda
      ld_joblog      TYPE tbtco-joblog,     "TBTCO tablosunda
      ld_jobname     TYPE tbtco-jobname,
      ld_jobcount    TYPE tbtco-jobcount,
      text           TYPE SO_TEXT255.

DATA: P_DATUM LIKE SY-DATUM.

CALL FUNCTION 'CONVERSION_EXIT_IDATE_INPUT'
  EXPORTING
    INPUT        =   INPUT_DATE
  IMPORTING
    OUTPUT       =   P_DATUM.

* Güncel tarihten 1 gün önoesini LV_DATE de#i#keninde tutuyoruz
* Girilen tarihi LV_DATE'te tutuyoruz ki kullan#o# istedi#i tarihten itibarenki kay#tlar# çekebilsin
DATA: LV_DATE TYPE DATS,
      LV_TIME TYPE TIMS.
LV_TIME = SY-UZEIT.
LV_DATE = P_DATUM.
*LV_DATE = SY-DATUM - 1.
*LV_DATE = SY-DATUM - 364.

*&---------------------------------------------------------------------*
*&  Background Job Cancelled with JOBLOG                               *
*&---------------------------------------------------------------------*
*STATUS -->
*   P: Tamamland#
*   A: Hata l#
*   S: Onayland# (Çal##acak)
*   R: Planland# (Çal##mayacak)
*   N: Onayland# ?
SELECT O~JOBNAME, O~JOBCOUNT, O~JOBLOG, O~AUTHCKMAN, P~AUTHCKNAM, O~STATUS, O~SDLSTRTDT, O~SDLSTRTTM
  INTO CORRESPONDING FIELDS OF TABLE @it_ma
  FROM (
   TBTCO AS O
   INNER JOIN TBTCP AS P
   ON P~JOBNAME = O~JOBNAME
   AND P~JOBCOUNT = O~JOBCOUNT
   AND O~STATUS = 'A'
   AND O~SDLSTRTDT GE @LV_DATE
*   AND O~SDLSTRTTM GE @LV_TIME
) .
SORT IT_MA BY SDLSTRTDT ASCENDING SDLSTRTTM ASCENDING.

TYPES: BEGIN OF TY_DATA,
   jobname TYPE tbtco-jobname,
   jobcount TYPE tbtcp-jobcount,
END OF TY_DATA.
DATA: IT_DATA TYPE TABLE OF TY_DATA.

SELECT O~JOBNAME
  O~JOBCOUNT
  FROM TBTCO AS O
  INNER JOIN TBTCP AS P
  ON P~JOBNAME = O~JOBNAME AND
  P~JOBCOUNT = O~JOBCOUNT
  INTO TABLE IT_DATA
  UP TO 10 ROWS.
*cl_demo_output=>display( IT_DATA ).

data:
       gd_cutputdate type string,
       gd_cutputtime type string,
       gd_day(2),    "field to store day 'DD'
       gd_month(2),  "field to store month 'MM'
       gd_year(4),   "field to store year 'YYYY'
       gd_hour(2),
       gd_minute(2),
       gd_second(2).
```

```
LOOP AT it_ma into OUTPUT.

  ld_client     =   OUTPUT-authckman.  "TBTCO tablosunda
  ld_joblog     =   OUTPUT-joblog.     "TBTCO tablosunda
  ld_jobname    =   OUTPUT-jobname.
  ld_jobcount   =   OUTPUT-jobcount.

  CALL FUNCTION 'BP_JOBLOG_READ'
   EXPORTING
    client                    = ld_client     " tbtojob-authckman  Job Clients
    jobcount                  = ld_jobcount   " tbtojob-jobcount  Job identification no.
    joblog                    = ld_joblog     " tbtojob-joblog  Name of Job Log in TemSe Database
    jobname                   = ld_jobname    " tbtojob-jobname  Job Name
*    lines                     = ld_lines      " btcint4       No. of Lines
*    direction                 = ld_direction  " btcchar1      Read Direction (B = From Beginning, E = From End)
   TABLES
    joblogtbl                 = it_joblogtbl
   EXCEPTIONS
    CANT_READ_JOBLOG          = 1
    JOBCOUNT_MISSING          = 2
    JOBLOG_DOES_NOT_EXIST     = 3
    JOBLOG_IS_EMPTY           = 4
    JOBLOG_NAME_MISSING       = 5
    JOBNAME_MISSING           = 6
    JOB_DOES_NOT_EXIST        = 7
  .  " BP_JOBLOG_READ

  clear gd_day.
  clear gd_month.
  clear gd_year.
  clear gd_hour.
  clear gd_minute.
  clear gd_second.

* split date into 3 fields to store 'DD', 'MM' and 'YYYY'
  gd_day(2)    = OUTPUT-SDLSTRTDT+6(2).
  gd_month(2)  = OUTPUT-SDLSTRTDT+4(2).
  gd_year(4)   = OUTPUT-SDLSTRTDT(4).
* split time into 3 fields to store 'HH', 'MM' and 'SS'
  gd_hour(2)     = OUTPUT-SDLSTRTTM(2).
  gd_minute(2)   = OUTPUT-SDLSTRTTM+2(2).
  gd_second(2)   = OUTPUT-SDLSTRTTM+4(2).

  concatenate gd_day gd_month gd_year     into gd_outputdate separated by '.'.
  concatenate gd_hour gd_minute gd_second into gd_outputtime separated by ':'.

  OUTPUT-DATE = gd_outputdate.
  OUTPUT-TIME = gd_outputtime.

  LOOP AT it_joblogtbl INTO wa_joblogtbl.
    IF wa_joblogtbl-msgtype = 'A'.
      CONCATENATE OUTPUT-TEXT wa_joblogtbl-TEXT '.' newline INTO OUTPUT-TEXT.
    ENDIF.
    IF wa_joblogtbl-msgtype = 'E'.
      CONCATENATE OUTPUT-TEXT wa_joblogtbl-TEXT '.' newline INTO OUTPUT-TEXT.
    ENDIF.
  ENDLOOP.

  APPEND OUTPUT.

ENDLOOP.

ENDFUNCTION.
```

EK 4.2. Girilen tarihten itibaren artalan işleri için hata kayıtlarını getiren ve JSON formatına dönüştüren program kodu

```
method IF_HTTP_EXTENSION~HANDLE_REQUEST.

DATA: lv_path_info      TYPE string,
      lv_request_method TYPE string.

DATA: it_inputparams TYPE tihttpnvp,
      ls_inputparams TYPE LINE OF tihttpnvp.

FIELD-SYMBOLS <fs_param> TYPE LINE OF tihttpnvp.

DATA: TAB     TYPE c VALUE cl_abap_char_utilities=>horizontal_tab,
      NEWLINE TYPE c VALUE cl_abap_char_utilities=>NEWLINE.

DATA: OUTPUT TYPE TABLE OF ZBACKGROUNDJOBS.
DATA: WA_OUTPUT TYPE ZBACKGROUNDJOBS.

DATA: BEGIN OF itab,
    JOBNAME   TYPE TBTCO-JOBNAME,
    JOBCOUNT  TYPE TBTCO-JOBCOUNT,
    JOBLOG    TYPE TBTCO-JOBLOG,   "TBTCO tablosunda
    AUTHCKMAN TYPE TBTCO-AUTHCKMAN,"TBTCO tablosunda
    AUTHCKNAM TYPE TBTCP-AUTHCKNAM,
    STATUS    TYPE TBTCO-STATUS,
    SDLSTRTDT TYPE TBTCO-SDLSTRTDT,
    SDLSTRTTM TYPE TBTCO-SDLSTRTTM,
END OF itab,
it_ma LIKE TABLE OF itab,
wa_ma LIKE LINE OF it_ma.

DATA: it_line LIKE LINE OF it_ma.

DATA: it_joblogtbl  TYPE STANDARD TABLE OF TBTC5,  "TABLES PARAM
      wa_joblogtbl  LIKE LINE OF it_joblogtbl,
      ld_client     TYPE tbtco-authckman,   "TBTCO tablosunda
      ld_joblog     TYPE tbtco-joblog,      "TBTCO tablosunda
      ld_jobname    TYPE tbtco-jobname,
      ld_jobcount   TYPE tbtco-jobcount,
      text          TYPE SO_TEXT255.

DATA: P_DATUM LIKE SY-DATUM,
      input_date TYPE STRING.

"Get request info
lv_path_info = server->request->get_header_field( name = '~path_info' ).
lv_request_method = server->request->get_header_field( name = '~request_method' ).

"Only GET is allowed
IF lv_request_method NE 'GET'.
  CALL METHOD server->response->set_header_field( name = 'Allow' value = 'GET' ).
  CALL METHOD server->response->set_status( code = '405' reason = 'Method not allowed' ).
  EXIT.
ENDIF.

"Get GET parameters
CALL METHOD server->request->get_form_fields
  CHANGING
    fields = it_inputparams.

UNASSIGN <fs_param>.
LOOP AT it_inputparams ASSIGNING <fs_param>.
  TRANSLATE <fs_param>-name TO UPPER CASE.
ENDLOOP.

CLEAR ls_inputparams.
READ TABLE it_inputparams INTO ls_inputparams WITH KEY name = 'DATE'.
input_date = ls_inputparams-value.
CLEAR ls_inputparams.

"Empty parameter
IF input_date IS INITIAL.
  CALL METHOD server->response->set_status( code = '404' reason = 'Empty parameters are not allowed' ).
  CALL METHOD server->response->set_cdata( data = 'There are one or more missing parameters' ).
  EXIT.
ENDIF.


CALL FUNCTION 'CONVERSION_EXIT_IDATE_INPUT'
  EXPORTING
    INPUT       =   input_date
  IMPORTING
    OUTPUT      =   P_DATUM.

* Güncel tarihten 1 gün öncesini LV_DATE de#i#keninde tutuyoruz
* Girilen tarihi LV_DATE'te tutuyoruz ki kullan#o# istedi#i tarihten itibarenki kay#tlar# çekebilsin
DATA: LV_DATE TYPE DATS,
      LV_TIME TYPE TIMS.
LV_TIME = SY-UZEIT.
LV_DATE = P_DATUM.
*LV_DATE = SY-DATUM - 1.
*LV_DATE = SY-DATUM - 364.
```

```
91
92  *&---------------------------------------------------------*
93  *&  Background Job Cancelled with JOBLOG                  *
94  *&---------------------------------------------------------*
95  *STATUS ->
96  *  F: Tamamland#
97  *  A: Hatal#
98  *  S: Onayland# (Çal##acak)
99  *  P: Planland# (Çal##mayacak)
100 *  R: Onayland# ?
101 SELECT O~JOBNAME, O~JOBCOUNT, O~JOBLOG, O~AUTHCKMAN, P~AUTHCKNAM, O~STATUS, O~SDLSTRTDT, O~SDLSTRTTM
102   INTO CORRESPONDING FIELDS OF TABLE @it_ma
103   FROM (
104    TBTCO AS O
105    INNER JOIN TBTCP AS P
106    ON P~JOBNAME = O~JOBNAME
107    AND P~JOBCOUNT = O~JOBCOUNT
108    AND O~STATUS = 'A'
109    AND O~SDLSTRTDT GE @LV_DATE
110 *   AND O~SDLSTRTTM GE @LV_TIME
111 ) .
112
113 SORT IT_MA BY SDLSTRTDT ASCENDING SDLSTRTTM ASCENDING.
114
115 TYPES: BEGIN OF TY_DATA,
116   jobname TYPE tbtco-jobname,
117   jobcount TYPE tbtcp-jobcount,
118   END OF TY_DATA.
119   DATA: IT_DATA TYPE TABLE OF TY_DATA.
120
121 SELECT O~JOBNAME
122   O~JOBCOUNT
123   FROM TBTCO AS O
124   INNER JOIN TBTCP AS P
125   ON P~JOBNAME = O~JOBNAME AND
126   P~JOBCOUNT = O~JOBCOUNT
127   INTO TABLE IT_DATA
128   UP TO 10 ROWS.
129 *cl_demo_output=>display( IT_DATA ).
130
131 data:
132       gd_outputdate type string,
133       gd_outputtime type string,
134       gd_day(2),   "field to store day 'DD'
135       gd_month(2), "field to store month 'MM'
136       gd_year(4),  "field to store year 'YYYY'
137       gd_hour(2),
138       gd_minute(2),
139       gd_second(2).
140
141 LOOP AT it_ma INTO wa_output.
142
143   ld_client     =   WA_OUTPUT-authckman.  "TBTCO tablosunda
144   ld_joblog     =   WA_OUTPUT-joblog.     "TBTCO tablosunda
145   ld_jobname    =   WA_OUTPUT-jobname.
146   ld_jobcount   =   WA_OUTPUT-jobcount.
147
148   CALL FUNCTION 'BP_JOBLOG_READ'
149    EXPORTING
150     client                  = ld_client     " tbtcjob-authckman  Job Clients
151     jobcount                = ld_jobcount   " tbtcjob-jobcount  Job identification no.
152     joblog                  = ld_joblog     " tbtcjob-joblog  Name of Job Log in TemSe Database
153     jobname                 = ld_jobname    " tbtcjob-jobname  Job Name
154 *    lines                   = ld_lines      " btcint4       No. of Lines
155 *    direction               = ld_direction  " btcchar1      Read Direction (B = From Beginning, E = From End)
156    TABLES
157     joblogtbl               = it_joblogtbl
158    EXCEPTIONS
159     CANT_READ_JOBLOG        = 1
160     JOBCOUNT_MISSING        = 2
161     JOBLOG_DOES_NOT_EXIST   = 3
162     JOBLOG_IS_EMPTY         = 4
163     JOBLOG_NAME_MISSING     = 5
164     JOBNAME_MISSING         = 6
165     JOB_DOES_NOT_EXIST      = 7
166   .  "  BP_JOBLOG_READ
```

```
    clear gd_day.
    clear gd_month.
    clear gd_year.
    clear gd_hour.
    clear gd_minute.
    clear gd_second.

  * split date into 3 fields to store 'DD', 'MM' and 'YYYY'
    gd_day(2)    = WA_OUTPUT-SDLSTRTDT+6(2).
    gd_month(2)  = WA_OUTPUT-SDLSTRTDT+4(2).
    gd_year(4)   = WA_OUTPUT-SDLSTRTDT(4).
  * split time into 3 fields to store 'HH', 'MM' and 'SS'
    gd_hour(2)     = WA_OUTPUT-SDLSTRTTM(2).
    gd_minute(2)   = WA_OUTPUT-SDLSTRTTM+2(2).
    gd_second(2)   = WA_OUTPUT-SDLSTRTTM+4(2).

    concatenate gd_day gd_month gd_year     into gd_outputdate separated by '.'.
    concatenate gd_hour gd_minute gd_second into gd_outputtime separated by ':'.

    WA_OUTPUT-DATE = gd_outputdate.
    WA_OUTPUT-TIME = gd_outputtime.

    LOOP AT it_joblogtbl INTO wa_joblogtbl.
      IF wa_joblogtbl-msgtype = 'A'.
        CONCATENATE WA_OUTPUT-TEXT wa_joblogtbl-TEXT '.' newline INTO WA_OUTPUT-TEXT.
      ENDIF.
      IF wa_joblogtbl-msgtype = 'E'.
        CONCATENATE WA_OUTPUT-TEXT wa_joblogtbl-TEXT '.' newline INTO WA_OUTPUT-TEXT.
      ENDIF.
    ENDLOOP.

    APPEND WA_OUTPUT TO OUTPUT.

  ENDLOOP.

  DATA writer TYPE REF TO cl_sxml_string_writer.
  DATA json TYPE xstring.
  "ABAP to JSON
  writer = cl_sxml_string_writer=>create( type = if_sxml=>co_xt_json ).
  CALL TRANSFORMATION id SOURCE item = output
                         RESULT XML writer.
  json = writer->get_output( ).
  CALL METHOD CL_BCS_CONVERT=>XSTRING_TO_STRING
    EXPORTING
      IV_XSTR   = json
      IV_CP     = 1100
    RECEIVING
      RV_STRING = DATA(lv_string)
      .

  "Set JSON Content-Type
  CALL METHOD server->response->set_header_field( name = 'Content-Type' value = 'application/json; charset=UTF-8' ).
  "Set Response Data
  CALL METHOD server->response->set_cdata( data = lv_string ).

  endmethod.
```

EK 4.3. Girilen tarihten itibaren artalan işleri için hata kayıtlarını getiren ve XML formatına dönüştüren program kodu

```
method IF_HTTP_EXTENSION~HANDLE_REQUEST.

data:lv_path type string,
     lv_data type string,
     lt_data type table of string,
     lt_list type table of BAPIORDERS,
     ls_list type BAPIORDERS,
     XML_OUT type string.

DATA: TAB     TYPE c VALUE cl_abap_char_utilities=>horizontal_tab,
      NEWLINE TYPE c VALUE cl_abap_char_utilities=>NEWLINE.

DATA: OUTPUT TYPE TABLE OF ZBACKGROUNDJOBS.
DATA: WA_OUTPUT TYPE ZBACKGROUNDJOBS.

DATA: BEGIN OF itab,
    JOBNAME   TYPE TBTCO-JOBNAME,
    JOBCOUNT  TYPE TBTCO-JOBCOUNT,
    JOBLOG    TYPE TBTCO-JOBLOG,    "TBTCO tablosunda
    AUTHCKMAN TYPE TBTCO-AUTHCKMAN,"TBTCO tablosunda
    AUTHCKNAM TYPE TBTCP-AUTHCKNAM,
    STATUS    TYPE TBTCO-STATUS,
    SDLSTRTDT TYPE TBTCO-SDLSTRTDT,
    SDLSTRTTM TYPE TBTCO-SDLSTRTTM,
END OF itab,
it_ma LIKE TABLE OF itab,
wa_ma LIKE LINE OF it_ma.

DATA: it_line LIKE LINE OF it_ma.

DATA: it_joblogtbl  TYPE STANDARD TABLE OF TBTC5,  "TABLES PARAM
      wa_joblogtbl  LIKE LINE OF it_joblogtbl,
      ld_client     TYPE tbtco-authckman,  "TBTCO tablosunda
      ld_joblog     TYPE tbtco-joblog,     "TBTCO tablosunda
      ld_jobname    TYPE tbtco-jobname,
      ld_jobcount   TYPE tbtco-jobcount,
      text          TYPE SO_TEXT255.

DATA: P_DATUM LIKE SY-DATUM.

* get the request attributes

  lv_path = server->REQUEST->get_header_field( name = '~PATH_INFO' ).

  SHIFT lv_path LEFT BY 1 PLACES.


CALL FUNCTION 'CONVERSION_EXIT_IDATE_INPUT'
  EXPORTING
    INPUT        =   lv_path
  IMPORTING
    OUTPUT       =   P_DATUM.

* Güncel tarihten 1 gün öncesini LV_DATE de#i#keninde tutuyoruz
* Girilen tarihi LV_DATE'te tutuyoruz ki kullan#c# istedi#i tarihten itibarenki kay#tlar# çekebilsin
DATA: LV_DATE TYPE DATS,
      LV_TIME TYPE TIMS.
LV_TIME = SY-UZEIT.
LV_DATE = P_DATUM.
*LV_DATE = SY-DATUM - 1.
*LV_DATE = SY-DATUM - 364.

*&---------------------------------------------------*
*&  Background Job Cancelled with JOBLOG             *
*&---------------------------------------------------*
*STATUS ->
*  F: Tamamland#
*  A: Hatal#
*  S: Onayland# (Çal##acak)
*  P: Planland# (Çal##mayacak)
*  R: Onayland# ?
SELECT O~JOBNAME, O~JOBCOUNT, O~JOBLOG, O~AUTHCKMAN, P~AUTHCKNAM, O~STATUS, O~SDLSTRTDT, O~SDLSTRTTM
  INTO CORRESPONDING FIELDS OF TABLE @it_ma
  FROM (
   TBTCO AS O
   INNER JOIN TBTCP AS P
   ON P~JOBNAME = O~JOBNAME
   AND P~JOBCOUNT = O~JOBCOUNT
   AND O~STATUS = 'A'
   AND O~SDLSTRTDT GE @LV_DATE
*   AND O~SDLSTRTTM GE @LV_TIME
) .

SORT IT_MA BY SDLSTRTDT ASCENDING SDLSTRTTM ASCENDING.

TYPES: BEGIN OF TY_DATA,
  jobname TYPE tbtco-jobname,
  jobcount TYPE tbtcp-jobcount,
  END OF TY_DATA.
  DATA: IT_DATA TYPE TABLE OF TY_DATA.

SELECT O~JOBNAME
  O~JOBCOUNT
  FROM TBTCO AS O
  INNER JOIN TBTCP AS P
  ON P~JOBNAME = O~JOBNAME AND
  P~JOBCOUNT = O~JOBCOUNT
  INTO TABLE IT_DATA
  UP TO 10 ROWS.
*cl_demo_output=>display( IT_DATA ).
```

```
101
102  data:
103        gd_outputdate type string,
104        gd_outputtime type string,
105        gd_day(2),   "field to store day 'DD'
106        gd_month(2), "field to store month 'MM'
107        gd_year(4),  "field to store year 'YYYY'
108        gd_hour(2),
109        gd_minute(2),
110        gd_second(2).
111
112  LOOP AT it_ma INTO wa_output.
113
114    ld_client     =   WA_OUTPUT-authclman.  "TBTCO tablosunda
115    ld_joblog     =   WA_OUTPUT-joblog.     "TBTCO tablosunda
116    ld_jobname    =   WA_OUTPUT-jobname.
117    ld_jobcount   =   WA_OUTPUT-jobcount.
118
119    CALL FUNCTION 'BP_JOBLOG_READ'
120     EXPORTING
121      client                    = ld_client     " tbtcjob-authclman  Job Clients
122      jobcount                  = ld_jobcount   " tbtcjob-jobcount  Job identification no.
123      joblog                    = ld_joblog     " tbtcjob-joblog  Name of Job Log in TemSe Database
124      jobname                   = ld_jobname    " tbtcjob-jobname  Job Name
125  *    lines                      = ld_lines      " btcint4       No. of Lines
126  *    direction                  = ld_direction  " btcchar1      Read Direction (B = From Beginning, E = From End)
127     TABLES
128      joblogtbl                 = it_joblogtbl
129     EXCEPTIONS
130      CANT_READ_JOBLOG          = 1
131      JOBCOUNT_MISSING          = 2
132      JOBLOG_DOES_NOT_EXIST     = 3
133      JOBLOG_IS_EMPTY           = 4
134      JOBLOG_NAME_MISSING       = 5
135      JOBNAME_MISSING           = 6
136      JOB_DOES_NOT_EXIST        = 7
137    .  "  BP_JOBLOG_READ
138
139    clear gd_day.
140    clear gd_month.
141    clear gd_year.
142    clear gd_hour.
143    clear gd_minute.
144    clear gd_second.
145
146  * split date into 3 fields to store 'DD', 'MM' and 'YYYY'
147    gd_day(2)    = WA_OUTPUT-SDLSTRTDT+6(2).
148    gd_month(2)  = WA_OUTPUT-SDLSTRTDT+4(2).
149    gd_year(4)   = WA_OUTPUT-SDLSTRTDT(4).
150  * split time into 3 fields to store 'HH', 'MM' and 'SS'
151    gd_hour(2)     = WA_OUTPUT-SDLSTRTTM(2).
152    gd_minute(2)   = WA_OUTPUT-SDLSTRTTM+2(2).
153    gd_second(2)   = WA_OUTPUT-SDLSTRTTM+4(2).
154
155    concatenate gd_day gd_month gd_year     into gd_outputdate separated by '.'.
156    concatenate gd_hour gd_minute gd_second into gd_outputtime separated by ':'.
157
158    WA_OUTPUT-DATE = gd_outputdate.
159    WA_OUTPUT-TIME = gd_outputtime.
160
161    LOOP AT it_joblogtbl INTO wa_joblogtbl.
162      IF wa_joblogtbl-msgtype = 'A'.
163        CONCATENATE WA_OUTPUT-TEXT wa_joblogtbl-TEXT '.' newline INTO WA_OUTPUT-TEXT.
164      ENDIF.
165      IF wa_joblogtbl-msgtype = 'E'.
166        CONCATENATE WA_OUTPUT-TEXT wa_joblogtbl-TEXT '.' newline INTO WA_OUTPUT-TEXT.
167      ENDIF.
168    ENDLOOP.
169
170    APPEND WA_OUTPUT TO OUTPUT.
171
172  ENDLOOP.
173
174  * Convert the data to  XML format
175
176    CALL TRANSFORMATION ID
177
178                  SOURCE TAB = output
179
180              RESULT XML XML_OUT
181
182                  OPTIONS XML_HEADER = 'WITHOUT_ENCODING'.
183
184  * Set the converted data to Browser
185
186    server->response->set_cdata( data = XML_OUT ).
187
188  endmethod.
```