Avrupa Bilim ve Teknoloji Dergisi
Sayı 52, S. 201-211, Kasım 2023

Araştırma Makalesi

www.ejosat.com ISSN:2148-2683

European Journal of Science and Technology
No. 52, pp. 201-211, November 2023

Research Article

Kurumsal Kaynak Planlama İş Yazılımı Sistemlerinde Uygulama Programlama Arayüzü ve İstemci Farklılıklarının Performansa Etkisi

Ali Burak Zeytinci[1], Rüya Şamlı[2*]

[1] İstanbul Üniversitesi-Cerrahpaşa, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, İstanbul, Türkiye, (ORCID: 0009-0005-0089-7924), zeytinciburak@gmail.com

[2*] İstanbul Üniversitesi-Cerrahpaşa, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, İstanbul, Türkiye, (ORCID: 0000-0002-8723-1228), ruyasamli@iuc.edu.tr

(İlk Geliş Tarihi 15 Mayıs 2023 ve Kabul Tarihi 23 Kasım 2023)

(**DOI:** 10.5281/zenodo.10439864)

ATIF/REFERENCE: Zeytinci, A. B. & Şamlı, R. (2023). Kurumsal Kaynak Planlama İş Yazılımı Sistemlerinde Uygulama Programlama Arayüzü ve İstemci Farklılıklarının Performansa Etkisi. *Avrupa Bilim ve Teknoloji Dergisi*, (52), 201-211.

Öz

Bu çalışma kapsamında ERP (Enterprise Resource Planning - Kurumsal Kaynak Planlama) iş yazılımları içerisinde sıklıkla kullanılan SAP sistemlerinde uygulama programlama arayüzü (API – Application Programming Interface) ve istemci 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 gerçekleştirilmiştir. Her bir çalışmanın altında da C#, Java ve JavaScript (Node.js) programlama dilleri ve platformları ile çeşitli uygulamalar yapılmış ve bu uygulamaların performansları incelenmiştir.

Anahtar Kelimeler: Kurumsal Kaynak Planlama, Uygulama Programlama Arayüzü, Performans Analizi.

Effects of Api and Consumer Differences on Performance in ERP Business Software Systems

Abstract

Within the study, ERP (Enterprise Resource Planning), the effect of application programming interface (API - Application Programming Interface) and client differences on performance in SAP systems, which are frequently used in business software, has been examined. To put it in detail, the effects of using different application programming interfaces on the performance, the performance differences in the interpretation of the data provided from the same web services by different programming languages as it grows, 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) 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.

[*] Sorumlu Yazar: ruyasamli@iuc.edu.tr

1. Giriş

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 AAS (Analitik Ağ Süreci) 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 (Multi-Objective Optimization on the basis of Ratio Analysis - 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, nihaî olarak karar verilen iki yazılım firmasının uygulama yazılımlarında bulanık AHP 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 değerlendirilmiştir. 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 çalışmada 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 değerlendirilmiştir.

2. Materyal ve Metot

2.1. Kurumsal Kaynak Planlama Sistemleri - ERP

ERP sistemleri, yazılımın ana iş süreçlerine entegre yönetimidir. 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.

2.1.1. Kurumsal Kaynak Planlama İş Yazılımı - SAP

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

2.2. Uygulama Programlama Arayüzleri - API

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 duyulmaktadır. Web servisler bu verilere erişir ve bazı internet protokolleri ile veriyi dışarıya açar, önyüz programı veya birçok farklı alandan doğru istekleri yapan ve yetkileri olan herkes bu verilere erişir, işler, kullanır.

2.3. Web Servisler

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 (World Wide Web) 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.

2.3.1. Basit Nesne Erişim Protokolü - SOAP

SOAP, bir paketin farklı uygulamalarının iletişim kurmasını sağlayan bir mesajlaşma protokolüdür. SOAP, web ile ilgili HTTP (Hypertext Transfer Protocol - Hiper Metin Transfer Protokolü) 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 API'ler yazmasına ve aynı zamanda özellikler ve işlevler eklemesine olanak tanır.

SOAP, genellikle XML (Extensible Markup Language - Genişletilebilir İşaretleme Dili) ile web API'leri oluşturmak için kullanılan hafif bir protokoldür. HTTP, SMTP (Simple Mail Transfer Protocol - Basit Posta Aktarım Protokolü) ve TCP (Transmission Control Protocol - İletim Kontrol Protokolü) 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.

2.3.2. Temsili Durum Transferi - 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 API'dir. 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.

2.4. Programlama Dilleri

Bu çalışma C#, Java ve JavaScript (Node.js) programlama dilleri kullanılarak gerçeklenmiş ve bu programlama dillerinin performans analizleri değerlendirilmiştir. 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.

2.4.1. ABAP

ABAP, SAP tarafından 1980'lerde geliştirilmeye başlanan bir uygulamaya özel dördüncü nesil programlama dilidir. Başlangıçta, büyük şirketlerin malzeme yönetimi, finans ve muhasebesi için iş uygulamaları oluşturmasını sağlayan bir platform olan SAP R/2'nin rapor(program) diliydi. Günümüzde ise R/3 ve S/4 ile devam etmektedir.

ABAP, Almanca Allgemeiner Berichts-Aufbereitungs-Prozessor (Türkçesi: Jenerik Rapor Hazırlama İşlemcisi) kısaltması olarak ortaya çıkmış, ancak daha sonra İngilizce'ye geçerek Advanced Business Application Programming olarak yeniden adlandırılmıştır.

2.4.2. 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.

2.4.3. Java

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

Java uygulamaları genellikle, temeldeki bilgisayar mimarisinden bağımsız olarak herhangi bir JVM (Java Virtual Machine - Java Sanal Makinesi'nde) 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.

2.4.4. JavaScript (Node.js)

JavaScript, WWW'nin HTML ve CSS (Cascading Style Sheets - Basamaklanmış Stil Katmanları) ile birlikte en temel üç teknolojilerinden biridir. Günümüzde neredeyse tüm web siteleri bir veya birden fazla JavaScript kütüphanesi yardımıyla geliştirilmektedir. Kullanılan tüm popüler web tarayıcılarında JavaScript ile geliştirilmiş bu kodları kullanıcının cihazında çalıştırmak için bir JavaScript motoru bulunmaktadır. Bu sayede cep telefonlardan akıllı saatlere, bilgisayarlardan yeni nesil otomobillere birçok internete bağlanan ve web tarayıcısı olan cihazda JavaScript ile oluşturulmuş web sitelerini görüntülemek mümkündür.

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 kullanılmakta idi, ancak artık sunucuların ve uygulamaların temel bileşeni haline gelmiştir. Bu alanda en popüler JavaScript Runtime ortamı Node.js'dir.

2.5. Programlama Platformları

Bu çalışma kapsamında 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.

2.5.1. Visual Studio Code - VSC

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. Özellikleri arasında hata ayıklama (debugging), sözdizimi vurgulama (syntax highlighting), akıllı kod tamamlama (IntelliSense), snippets, kod refactoring ve gömülü Git desteği yer almaktadır. 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 yerleşik olarak desteklenmektedir. Diğer dillere ise eklentiler ile destek vermektedir.

2.5.2. Visual Studio

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

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.

2.5.3. 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 ticarî bir "Ultimate" sürümü mevcuttur. IntelliJ IDEA kaynak kodunu JetBrains açık kaynak Apache Lisansı 2.0 altında 2009 yılında yayınlanmıştır.

2.6. Veri Formatları

Bu çalışmada SAP tarafına yapılan istekler ve alınan cevaplarda farklı standartlar kullanılmaktadır. Bunlar SOAP için WSDL (Web Services Description Language - Web Servisleri Tanımlama Dili) iken REST için JSON ve XML'dir.

2.6.1. Genişletilebilir İşaretleme Dili - XML

XML verileri depolamak ve farklı bilgisayarlar arasındaki iletişimde verileri iletmek için bir dosya formatıdır. Verileri hem makine hem de insan tarafından okunabilecek bir şekilde kodlanması için çeşitli kuralları bulunmaktadır. WWW Konsorsiyumu'nun 1998 tarihli XML 1.0 Spesifikasyonu ile tanımlanmıştır. XML'in tasarım hedefleri; basitliği, genelliği ve internet genelinde kullanılabilirliği vurgulamaktadır. Farklı insan dilleri için Unicode aracılığıyla desteği olan metinsel bir veri formatıdır.

2.6.2. Web Servisleri Tanımlama Dili - WSDL

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.

2.6.3. JavaScript Nesne Notasyonu - JSON

JSON, hafif (lightweight) bir veri değişim formatıdır. İnsanlar için okuma-yazması, 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 bir şekilde desteklemektedir.

3. Araştırma Sonuçları ve Tartışma

Bu çalışmada sırasıyla SOAP, JSON ve XML uygulamaları gerçekleştirilmiş ve sonuçlar sunulmuştur. Yapılan uygulamalar aşağıdaki şekilde özetlenebilir:

SOAP Uygulamaları:
- IDES Küçük Veri SOAP Çalışması
- LOCAL Küçük Veri SOAP Çalışması
- S4 Küçük Veri SOAP Çalışması
- IDES Büyük Veri SOAP Çalışması
- LOCAL Büyük Veri SOAP Çalışması
- S4 Büyük Veri SOAP Çalışması

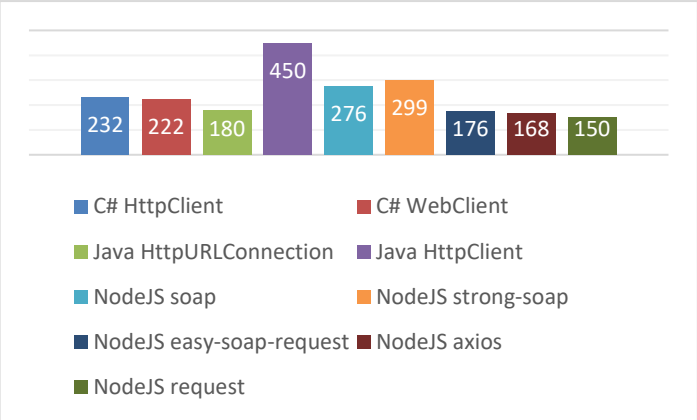
JSON Uygulamaları:
- IDES Küçük Veri JSON Çalışması
- LOCAL Küçük Veri JSON Çalışması
- S4 Küçük Veri JSON Çalışması
- IDES Büyük Veri JSON Çalışması
- LOCAL Büyük Veri JSON Çalışması
- S4 Büyük Veri JSON Çalışması

XML Uygulamaları:
- IDES Küçük Veri XML Çalışması
- LOCAL Küçük Veri XML Çalışması
- S4 Küçük Veri XML Çalışması
- IDES Büyük Veri XML Çalışması
- LOCAL Büyük Veri XML Çalışması
- S4 Büyük Veri XML Çalışması

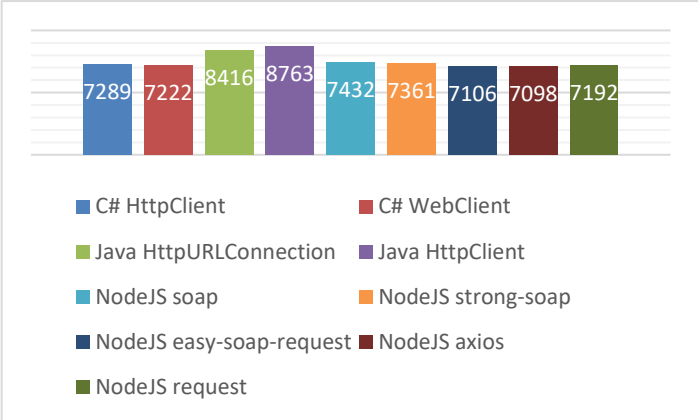
3.1. SOAP Uygulamaları

Bu çalışmada ilk olarak WSDL formatındaki küçük ve büyük verilerle SOAP API uygulaması gerçekleştirilmiştir. Test çeşitliliği ve sonuçların doğru yorumlanabilmesi için üç farklı sunucuyla çalışılmıştır. Sunuculardaki küçük veri büyüklükleri sırasıyla IDES ile LOCAL arasında yaklaşık 100 kat, LOCAL ile S4 arasında yaklaşık 5 kat büyüklük farkı olacak şekildedir. Büyük verilerde ise üç sunucuda da aynı verilerle çalışılmıştır ve bunların küçük veriye büyüklük oranları 100-10.000 kat fazladır. Üç farklı programlama dili kullanılıp bunlar sırasıyla C#, Java ve JavaScript (Node.js) dilleridir. Bu programlama dilleri 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 çalışma kapsamında yorumlanmıştır. Bu çalışma kapsamında yorumlanan veriler; programların toplam çalışma süreleri değil, belirtilen bu üç ana başlıkta geçirilen sürelerdir.

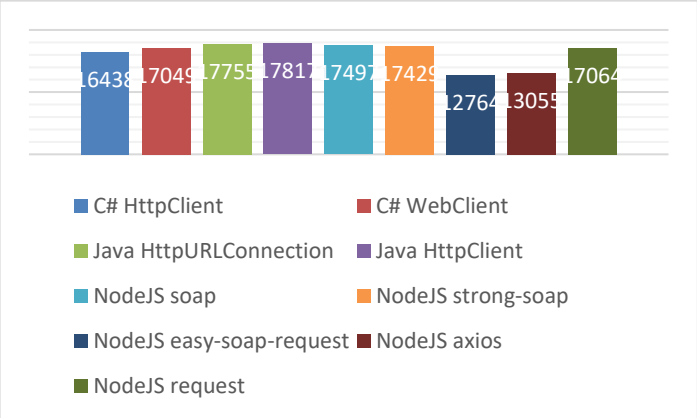
Kullanılan farklı diller ve bu dillerde kullanılan farklı kütüphaneler veya sınıflar SOAP istek mektubunu ve HTTP 1.1 isteğini farklı yöntemlerle oluşturup, gönderip, aynı cevabı almıştır. Bu cevap; SAP tarafında veritabanının farklı alanlarından SQL sorguları ve ABAP fonksiyonlarıyla çekilmiş, paketlenmiş ve gönderilmiştir. Sırasıyla Şekil 1'de IDES Küçük Veri SOAP Çalışması'nın, Şekil 2'de LOCAL Küçük Veri SOAP Çalışması'nın, Şekil 3'te S4 Küçük Veri SOAP Çalışması'nın, Şekil 4'te IDES Büyük Veri SOAP Çalışması'nın, Şekil 5'te LOCAL Büyük Veri SOAP Çalışması'nın, Şekil 6'da S4 Büyük Veri SOAP Çalışması'nın sonuçları sunulmuştur.



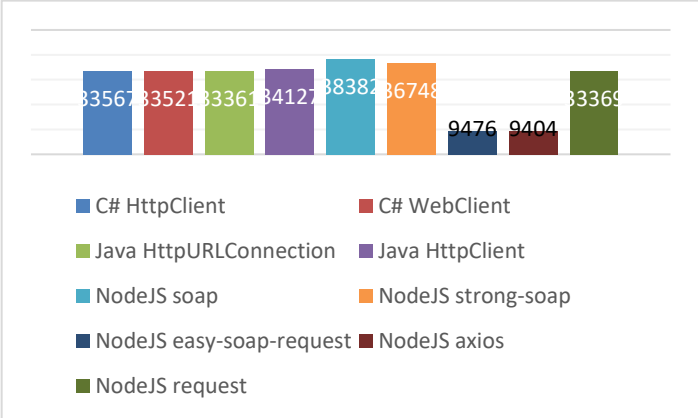
Şekil 1. IDES Küçük Veri SOAP Çalışması
(Figure 1. IDES Small Data SOAP Study)



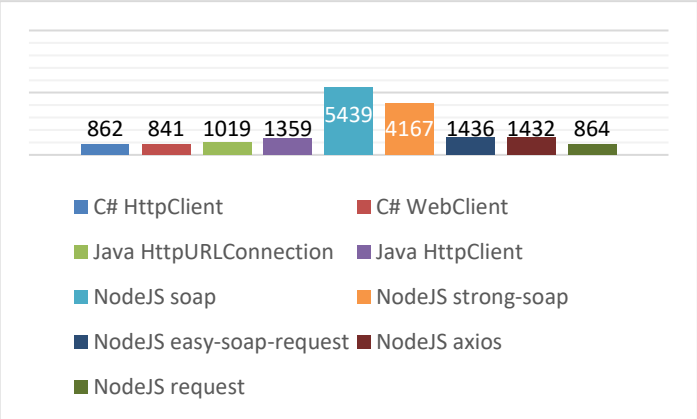
Şekil 2. LOCAL Küçük Veri SOAP Çalışması
(Figure 2. LOCAL Small Data SOAP Study)



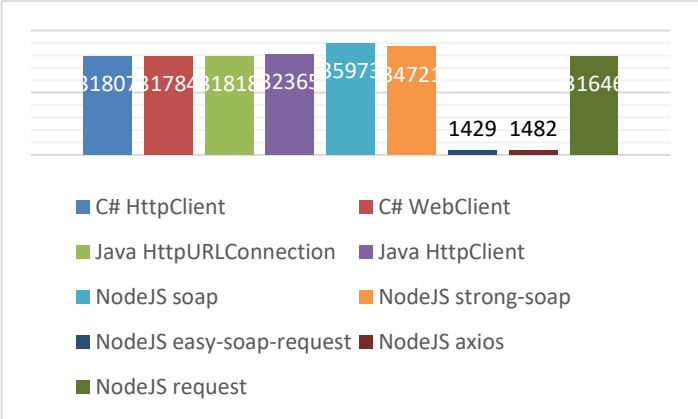
Şekil 3. S4 Küçük Veri SOAP Çalışması
(Figure 3. S4 Small Data SOAP Study)



Şekil 4. IDES Büyük Veri SOAP Çalışması
(Figure 4. IDES Large Data SOAP Study)



Şekil 5. LOCAL Büyük Veri SOAP Çalışması
(Figure 5. LOCAL Large Data SOAP Study)



Şekil 6. S4 Büyük Veri SOAP Çalışması
(Figure 6. S4 Large Data SOAP Study)

SAP sistemlerinden verileri dışarıya SOAP API ile WSDL formatında açmak istendiğinde çoğu senaryoda en iyi tercih Node.js ortamı olacaktır. Veri boyutu küçüldükçe C# programlama dilinde WebClient sınıfı kabul edilebilir sonuçlar vermektedir. Java programlama dilinde HttpClient sınıfı ise çoğu çalışmada olduğu gibi bu çalışmalarda da en yavaş ya da 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 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 çalışma kapsamında gerçeklenen senaryoların neredeyse tamamında request ile yakın sonuçlar elde eden diğer bir Node.js kütüphanesi olmuştur. Düzenli güncellemeler almaya devam eden ve mimarî 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.

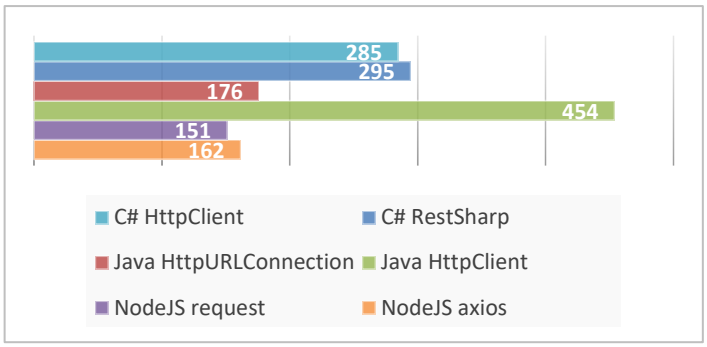
SOAP çalışmaları için Node.js ortamında toplam beş olmak üzere üç alternatif kütüphane daha kullanılmıştır. İlk ikisi REST çalışmalarında da kullanılan axios ve request kütüphaneleri olmak üzere kalan üçü sırasıyla soap, strong-soap ve easy-soap-request kütüphaneleridir. Bunlardan soap, Node.js ortamındaki 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 çalışma 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 çalışma 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.

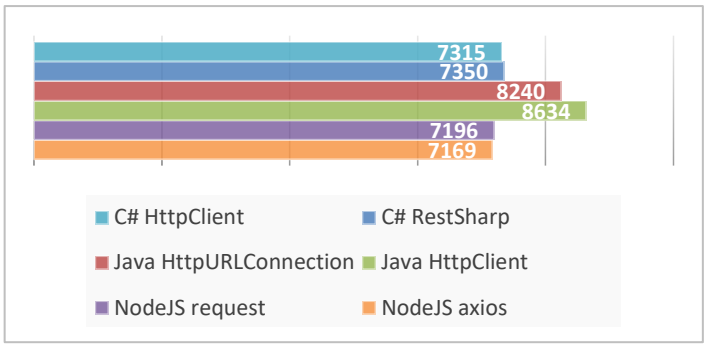
3.2. JSON Uygulamaları

Bu çalışmada ikinci olarak JSON formatındaki küçük ve büyük verilerle REST API uygulaması gerçekleştirilmiştir. Test çeşitliliği ve sonuçların doğru yorumlanabilmesi için üç farklı sunucuyla çalışılmıştır. Sunuculardaki küçük veri büyüklükleri sırasıyla IDES ile LOCAL arasında yaklaşık 100 kat, LOCAL ile S4 arasında yaklaşık 5 kat büyüklük farkı olacak şekildedir. Büyük verilerde ise üç sunucuda da aynı verilerle çalışılmış ve bunların küçük veriye büyüklük oranları 100-10.000 kat fazladır. 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 çalışma kapsamında yorumlanmıştır. Bu çalışma kapsamında yorumlanan veriler; programların toplam çalışma süreleri değil, belirtilen bu üç ana başlıkta geçirilen sürelerdir.

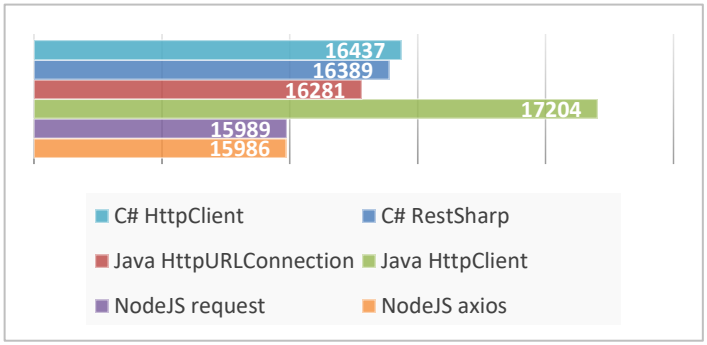
Kullanılan farklı diller ve bu dillerde kullanılan farklı kütüphaneler veya sınıflar REST isteklerini farklı yöntemlerle oluşturup, gönderip, aynı cevabı almıştır. Bu cevap; SAP tarafında veritabanının farklı alanlarından SQL sorguları ve ABAP fonksiyonlarıyla çekilmiş, paketlenmiş ve gönderilmiştir. Sırasıyla Şekil 7'de IDES Küçük Veri JSON Çalışması'nın, Şekil 8'de LOCAL Küçük Veri JSON Çalışması'nın, Şekil 9'da S4 Küçük Veri JSON Çalışması'nın, Şekil 10'da IDES Büyük Veri JSON Çalışması'nın, Şekil 11'de LOCAL Büyük Veri JSON Çalışması'nın, Şekil 12'de S4 Büyük Veri JSON Çalışması'nın sonuçları sunulmuştur.



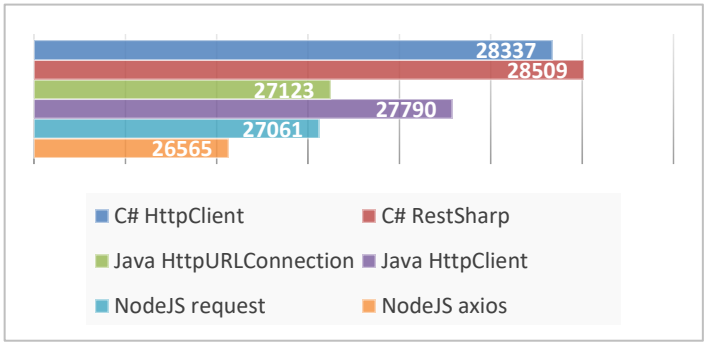
Şekil 7. IDES Küçük Veri JSON Çalışması
(Figure 7. IDES Small Data JSON Study)



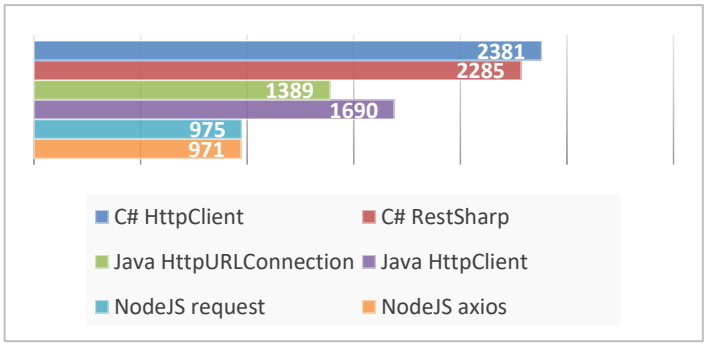
Şekil 8. LOCAL Küçük Veri JSON Çalışması
(Figure 8. LOCAL Small Data JSON Study)



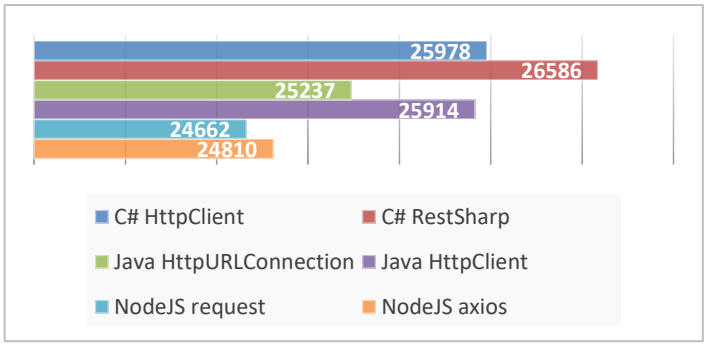
Şekil 9. S4 Küçük Veri JSON Çalışması
(Figure 9. S4 Small Data JSON Study)



Şekil 10. IDES Büyük Veri JSON Çalışması
(Figure 10. IDES Large Data JSON Study)



Şekil 11. LOCAL Büyük Veri JSON Çalışması
(Figure 11. LOCAL Large Data SOAP Study)



Şekil 12. S4 Büyük Veri JSON Çalışması
(Figure 12. S4 Large Data SOAP Study)

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 çalışmalardır. 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 çalışma 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.

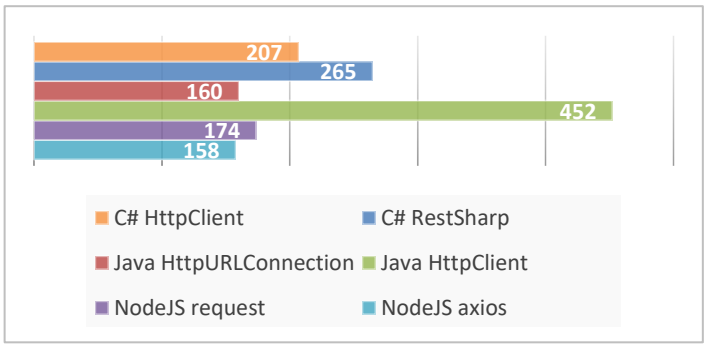
JSON formatındaki verilerin yorumlanması için C# programlama dilinde Json.NET çatısı, Java programlama dilinde JSON-Java kütüphanesi kullanılmıştır. Node.js ortamında ek bir kullanıma gerek yoktur. Bu durum da performansa oldukça etki etmektedir.

3.3. XML Uygulamaları

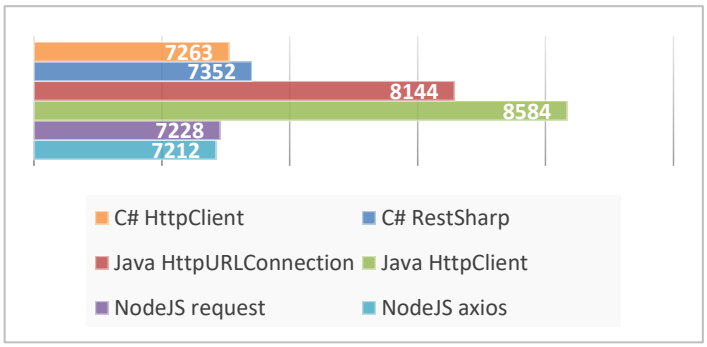
Bu çalışmada son olarak XML formatındaki küçük ve büyük verilerle REST API uygulaması gerçekleştirilmiştir. Test çeşitliliği ve sonuçların doğru yorumlanabilmesi için üç farklı sunucuyla çalışılmıştır. Sunuculardaki küçük veri büyüklükleri sırasıyla IDES ile LOCAL arasında yaklaşık 100 kat, LOCAL ile S4 arasında yaklaşık 5 kat büyüklük farkı olacak şeklindedir. Büyük verilerde ise üç

sunucuda da aynı verilerle çalışılmış ve bunların küçük veriye büyüklük oranları 100-10.000 kat fazladır. 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 çalışma kapsamında yorumlanmıştır. Bu çalışma kapsamında yorumlanan veriler; programların toplam çalışma süreleri değil, belirtilen bu üç ana başlıkta geçirilen sürelerdir.

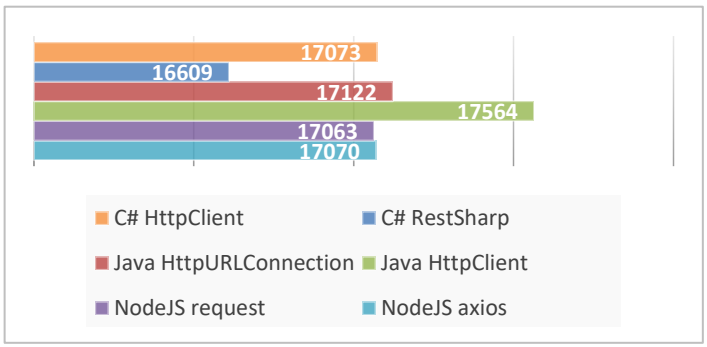
Kullanılan farklı diller ve bu dillerde kullanılan farklı kütüphaneler veya sınıflar REST isteklerini farklı yöntemlerle oluşturup, gönderip, aynı cevabı almıştır. Bu cevap; SAP tarafında veritabanının farklı alanlarından SQL sorguları ve ABAP fonksiyonlarıyla çekilmiş, paketlenmiş ve gönderilmiştir. Sırasıyla Şekil 13'de IDES Küçük Veri MXL Çalışması'nın, Şekil 14'te LOCAL Küçük Veri XML Çalışması'nın, Şekil 15'te S4 Küçük Veri XML Çalışması'nın, Şekil 16'da IDES Büyük Veri XML Çalışması'nın, Şekil 17'de LOCAL Büyük Veri XML Çalışması'nın, Şekil 18'de S4 Büyük Veri XML Çalışması'nın sonuçları sunulmuştur.



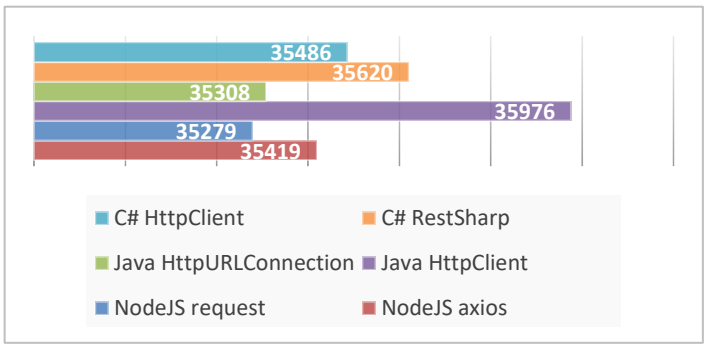
Şekil 13. IDES Küçük Veri XML Çalışması
(Figure 13. IDES Small Data XML Study)



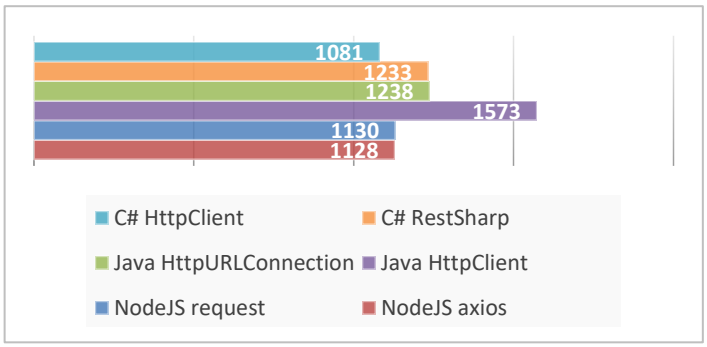
Şekil 14. LOCAL Küçük Veri XML Çalışması
(Figure 14. LOCAL Small Data XML Study)



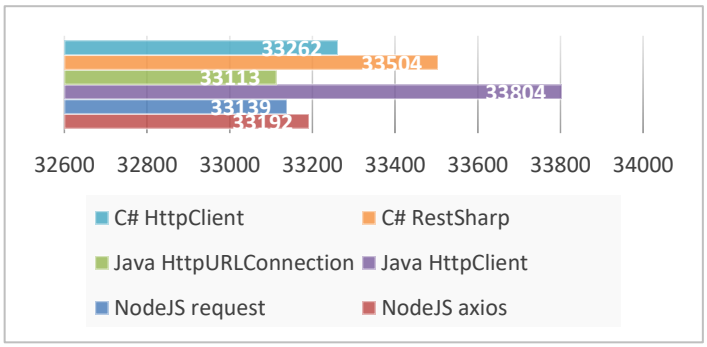
Şekil 15. S4 Küçük Veri XML Çalışması
(Figure 15. S4 Small Data XML Study)



Şekil 16. IDES Büyük Veri XML Çalışması
(Figure 16. IDES Large Data XML Study)



Şekil 17. LOCAL Büyük Veri XML Çalışması
(Figure 17. LOCAL Large Data XML Study)



Şekil 18. S4 Büyük Veri XML Çalışması
(Figure 18. S4 Large Data XML Study)

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.

Bütün çalışmalar boyunca 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 kullanılabileceği söylenebilir. Tablo 1'de gerçeklenen çalışmalar ve özet sonuçları verilmiştir.

Tablo 1. Gerçeklenen Çalışmalar ve Özet Sonuçları (sn.) (Table 1. Performed Studies and Summary Results) (sec.)

	IDES			**LOCAL**			**S4**		
	En Hızlı Çalışma	**Ortalama**	**Çeyrekler Ortalaması**	**En Hızlı Çalışma**	**Ortalama**	**Çeyrekler Ortalaması**	**En Hızlı Çalışma**	**Ortalama**	**Çeyrekler Ortalaması**
Küçük Veri SOAP	150	239	226	7098	7542	7312	12764	16319	16968
Küçük Veri JSON	151	254	229	7169	7651	7622	15986	16381	16243
Küçük Veri XML	158	236	207	7212	7630	7591	16609	17084	17087
Büyük Veri SOAP	9404	29106	33744	841	1935	1150	1429	25892	32006
Büyük Veri JSON	26565	27564	27638	971	1615	1607	24662	25531	25439
Büyük Veri XML	35279	35515	35461	1081	1230	1182	33113	33336	33298

4. Sonuç

Bu çalışmada, bir ERP iş yazılımı olan SAP sistemlerinden dışarıya açılması istenen verilerin boyutlarına göre seçilecek istemci (client), API ve veri formatları taraflı yaşanabilecek performans farkları ortaya konmuştur.

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. SAP tarafında SOAP mesajlar doğrudan çıkış yapabilirken, REST mesajlar için SAP içerisinde ek olarak her satır veriye XML veya JSON yazım formatına dönüştürme aşaması uygulanmaktadır. İstemci tarafında ise her satır veri bir sayaç fonksiyonu ile sayılmakta böylelikle en az bir kez yorumlanmaktadır. Fakat JavaScript, JSON cevapları doğrudan veya bir 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 iki kez yorumlanması anlamına gelmektedir. Gerçek hayat kullanımlarında bu senaryolar gerçekleşeceği için bu çalışmada da bu dönüşümlere de yer verilmiş, sonuçlar bu şekilde toplanmıştır.

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 çalışma 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.

SAP sistemlerinden dışarıya veri açılan performans odaklı çalışmalarda API ve veri formatı seçiminde bu çalışma kapsamında yapılan çalışmalar en performanslı sonuçlara göre sıralandığında SOAP API seçiminin özellikle veri boyutu büyüdükçe net bir şekilde tek tercih olduğu görülmektedir. 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 çözülmektedir. XML formatındaki veriler, JSON formatındaki aynı verilerden daha fazla etiket ve elementler ile yaklaşık yüzde elli daha büyük dosya boyutuna sahip

olabilmektedir. İstemci tarafından alınan bu verilerin yorumlanmasında ise JSON obje ve dizilerine dönüştürme işlemleri bu farkı kapatmakta hatta JSON çalışmalarını geriye düşürmektedir. Aynı işlemler tam tersinden hem XML hem JSON için SAP içerisinde de gerçekleştirildiği için performans farkı SOAP çalışmalarına kıyasla giderek artmaktadır.

Genel değerlendirme olarak performans odaklı çalışmalarda SAP sistemlerinden dışarıya SOAP API ile WSDL formatında veya RESTful API ile XML veya JSON formatında veriler açılmak istendiğinde 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.

Kaynakça

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

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

Başar, R. ve Arslan, H.M. (2017). 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.

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

Boztaş, M. ve Özmızrak, M. (2012). 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.

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

Dikmen, F.C. ve Kavakcı, I. (2022). 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.

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

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

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

Özkan Özen, Y.D. ve Koçak, A. (2017). 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.

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

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

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

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

Vatansever, K. ve Uluköy, M. (2013). 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.

Yeşildağ, B. (2010). 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, Muğla.

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