**İSTANBUL TEKNİK ÜNİVERSİTESİ ★ BİLİŞİM ENSTİTÜSÜ**

# ENTEGRE BİLGİ SİSTEMİ MODELİ GELİŞTİRİLMESİ: DataOCEAN©

**DOKTORA TEZİ**

**Egnar ÖZDİKİLİLER**

**İletişim Sistemleri Anabilim Dalı**

**Uydu Haberleşmesi ve Uzaktan Algılama Programı**

**ARALIK 2017**

**İSTANBUL TEKNİK ÜNİVERSİTESİ ★ BİLİŞİM ENSTİTÜSÜ**

# ENTEGRE BİLGİ SİSTEMİ MODELİ GELİŞTİRİLMESİ: DataOCEAN©

**DOKTORA TEZİ**

**Egnar ÖZDİKİLİLER**
**(705102006)**

**İletişim Sistemleri Anabilim Dalı**

**Uydu Haberleşmesi ve Uzaktan Algılama Programı**

**Tez Danışmanı: Doç. Dr. Çiğdem GÖKSEL**

**ARALIK 2017**

İTÜ, Bilişim Enstitüsü'nün 705102006 numaralı Doktora Öğrencisi **Egnar ÖZDİKİLİLER**, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı **"ENTEGRE BİLGİ SİSTEMİ MODELİ GELİŞTİRİLMESİ: DataOCEAN©"** başlıklı tezini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

| | | |
|---|---|---|
| **Tez Danışmanı :** | **Doç. Dr. Çiğdem GÖKSEL**<br>İstanbul Teknik Üniversitesi | ..... |
| **Jüri Üyeleri :** | **Prof. Dr. Selçuk PAKER**<br>İstanbul Teknik Üniversitesi | ..... |
| | **Doç. Dr. Filiz BEKTAŞ BALÇIK**<br>İstanbul Teknik Üniversitesi | ..... |
| | **Doç. Dr. Ahmet Korhan TANÇ**<br>Biruni Üniversitesi | ..... |
| | **Doç. Dr. Füsun BALIK ŞANLI**<br>Yıldız Teknik Üniversitesi | ..... |

**Teslim Tarihi :** **16 Kasım 2017**
**Savunma Tarihi :** **22 Aralık 2017**

***Aileme,***

## ÖNSÖZ

Çalışmamızın başladığı günden itibaren bana yol gösteren, teşvik eden, bilgi birikimini paylaşan, yazılarımı sabırla okuyup derleyen danışman hocam Doç. Dr. Çiğdem GÖKSEL'e;
Uydu Haberleşme ve Uzaktan Algılama bölümünü seçme fikrini veren ve tez çalışması süresince bilgi, görüş ve önerilerini paylaşan, beni yönlendiren hocam Prof. Dr. Selçuk PAKER'e,
bir yazılım dehası olan, benimle hayatı paylaşan ve her konuda destekleyen eşim Volkan ÖZDİKİLİLER'e,
tez süresince gösterdiği sabır ve anlayış için kızım Elis ÖZDİKİLİLER'e,
hayatım boyunca desteğini esirgemeyen, ömrünü çocuklarına adayan annem Mayzer VELİEVA'ya,
ilk önce beni dünyanın en mutlu ablası yapan ve defalarca gururlandıran kardeşim Y.Müh. Dr. Cihan MENSEİDOV'a
en içten teşekkürlerimi sunarım.

Paylaştığı bilgi, deneyim ve öğütleriyle beni hayata hazırlayan, aramızdan çok erken ayrılan babam, merhum İbrahim MENSEİDOV'a teşekkür eder, saygı, özlem ve minnetle anıyorum.

Aralık 2017

Egnar ÖZDİKİLİLER
Bilgisayar Yüksek Mühendisi

**İÇİNDEKİLER**

**Sayfa**

**KISALTMALAR**

**API** : Application Programming Interface
**CBO** : Coupling between Objects
**DIT** : Depth of Inheritance Tree
**DO** : DataOCEAN©
**EBS** : Enterprise Service Bus
**ESDI** : European Spatial Data Infrastructure
**EU** : European Union
**GIS** : Geographic Information Systems
**HTTP** : Hypertext Transfer Protocol
**HTTPS** : Secure Hypertext Transfer Protocol
**IETF** : Internet Engineering Task Force
**INSPIRE** : Infrastructure for Spatial Information in Europe
**LCOM** : Lack of Cohesion of Methods
**MVC** : Model-View-Controller
**NOC** : Number of Children
**OLY** : OceanLIBRARY©
**OGC** : Open Geospatial Consortium
**OOA** : Object Oriented Architecture
**PHP** : Hypertext Preprocessor
**REST** : Representational State Transfer
**RFC** : Response For Class
**SaaS** : Software as a Service
**SFA** : Simple Feature Access
**SMTP** : Simple Mail Transfer Protocol
**SOA** : Servise Oriented Architecture
**SOAP** : Simple Object Access Protocol
**SQL** : Structured Query Language
**UDDI** : Universal Description, Discovery, and Integration
**URI** : Uniform Resource Identifier
**URL** : Uniform Resource Locator
**W3C** : The World Wide Web Consortium
**WFS** : Web Feature Service
**WMC** : Weighted Methods for Class
**WMS** : Web Map Service
**WMTS** : Web Map Tile Service
**WS** : Web Service
**WSDL** : Web Services Description Language
**WWW** : World Wide Web
**XML** : eXtensible Markup Language

## ŞEKİL LİSTESİ

**Sayfa**

## ENTEGRE BİLGİ SİSTEMİ MODELİ GELİŞTİRİLMESİ: DataOCEAN©


### ÖZET

Günümüzde veri çeşitliliğinin artması ve veri hacminin çoğalması ile güvenilir, düzenli ve güçlü sistem tasarımı gereksinimi giderek daha önemli hale gelmiştir. Merkezi erişim sağlayan yeni nesil bilişim sistemleri tasarlanarak kullanım yaygınlaştırılmıştır. Bu durum, özellikle kurumlar arası veri paylaşımı ve farklı yapıdaki, sistemler arasındaki entegrasyon çalışmalarını hızlandırmıştır. Dolayısıyla, merkezi erişim amaçlayan sistem tasarım çalışmaları artmış, veri erişimi de web servislerinin yaygın olarak kullanıması ile kolaylaşmıştır.

Bu çalışma kapsamında, dağıtık sistemler için; hızlı, doğru ve güvenilir bilgiye erişimde kullanılabilecek yeni bir entegre bilgi sistemi modeli tasarlanmıştır. Tasarlanan model; birlikte çalışabilirlik ilkelerini koruyan, hibrid yapı temelli, birden fazla sistemi barındıran, entegrasyonu web servisleri aracılığı ile sağlayarak, çok yönlü veri akışına olanak tanımaktadır. Uzun vadede konum ve zaman kavramlarına dayanarak verileri analiz etmeyi, yönetmeyi ve işlemeyi de amaçlamaktadır.

Model, bilişim sistemi çalışmaların kavramsal merkezini (çekirdeğini) temsil etmesi, ayrıca kurumlar arası veri tekrarı durumunda, en doğru verinin tespiti ve güvenilir bilginin alınmasını sağlayacak şekilde kurgulanmıştır. Veri tekrarının, tutarsız analiz sonuçlarına neden olduğu bilinmektedir. Bu nedenle, oluşturulan bu entegre bilgi sisteminde, uluslar arası standartlara uygunluk esas alınmıştır. Birlikte çalışabilirlik ilkesi gereğince, sadece verileri aynı standartta saklamak ve/veya kullanmak değil, merkezi bilgi/veri sistemi tasarımının benimsenerek, verilerin paylaşılması ve ortak kullanılmasının sağlanması çok önemlidir.

"Entegre bilgi sistemi modeli geliştirilmesi: DataOCEAN©" konulu tez çalışmasında tasarlanan model ve sistem prototipi Servis Yönelimli (Servise Oriented Architecture – SOA) yapıyı temel alan ve Nesne Yönelimli (Object Oriented - OOA) mimari yapısına benzerlikler taşımaktadır. Kullanılan iskelet uygulaması (framework), MVC mimari tabanlı (Model-View-Controller) yazılmış, özgün bir çalışmadır. REST (Representational State Transfer) yaklaşımı ile tasarlanmış, RESTful ve SOAP (Simple Object Access Protocol) servis mimarilerine ilişkin web servisleri hazırlanmış, sistemin yönetim panelinin içereceği alanlar yazılmış ve test edilmiştir. Uygulama dili olarak PHP (Hypertext Preprocessor) tercih edilmiş, konumsal veri gösterimi için GoogleMAPs kullanılmıştır.

Çalışma süresince oluşturulan web servislerinin entegre edildiği yapı DataOCEAN©-DO (Veri Okyanusu) olarak, oluşturulan web servisleri kütüphanesi ise OceanLIBRARY©-OLY (Okyanus Kütüphanesi) olarak anılacaktır.

DataOCEAN© sistem modelinde veri yapıları web servisleridir. Web servisleri, sistem kütüphanesine entegre edilme aşamasında, nesne tabanlı programlamada

kullanılan hiyererşik oluşumla tanımlanmaktadırlar (içerdiği veri, servis yapısı, v.b. özellikler ile). Veri sorgulaması, dinamik olarak uygulama kapsamında kodlanan web arayüzü aracılığı ile gerçekleşmektedir. Web arayüzünde özgün sorgu yazıldıktan sonra, sorgu işleme aşamasında veri havuzundan kullanılacak web servisleri tespit edilir. Sistemde anlık olarak erişilebilir olan, veri doğruluğu en yüksek servislerden bilgi toplanır, web arayüzünde ilgili formatta sonuç sunulur. Kullanıcı, verinin kaynağı olan kurum ve/veya yapıyı bilmeden, alınan sonucun güncel ve sorgulama anı için en yüksek doğrulukta olduğunu bilmektedir. Dolayısıyla, kullanıcının yüksek yetkinlikte olması beklenmemektedir.

Sunulan modelin temel amacı, yazılım sürecini kısaltmaktır. Sistem, bilgi sistemlerinde bulunan verileri paylaşabilmeye uygun mimari ile tasarlanmış olması nedeniyle birlikte çalışabilirlik konusunda da iyi bir örnek olduğu düşünülmektedir. Tasarlanan DataOCEAN© mimarisinde birbirinden farklı sistemler veri alışverişinde bulunmaktadır. Özgün sorgulama platformu sunan ara yüzü ile sorgulama anında sistemde aktif olan web servisleri arasından doğruluk derecesi en yüksek olan servis/servisler seçilerek, işlemi kısa sürede hızlı ve doğru sonuçlandırmaktadır.

Bu çalışmanın giriş bölümünde, çalışmanın özeti, kullanılan yöntemler ve sistemin yapısı özetlenmiş, veri ve veri çeşitliliği konularında yapılan çalışmalara değinilmiştir. Birinci bölümde çalışmanın temeli olan web servisleri, OOA, SOA (REST ve SOAP), MVC mimarileri incelenmiştir. İkinci bölümde araştırma yöntemi, geliştirilen DO mimarisi ve OLY kütüphanesi açıklanmıştır. Üçüncü bölümde yazılan prototip uygulama ve alınan sonuçlar irdelenmiştir. Sonuç bölümünde sonuçlar özetlenmiş, hedeflenen sonraki çalışmalar ve çalışmanın katkıları belirtilmiştir.

**DEVELOPMENT A MODEL FOR INTEGRATED INFORMATION SYSTEMS: DataOCEAN©**

## SUMMARY

Today, with the increase in the variety and volumes of data, the need for a reliable, systematic and powerful system design has increasingly become more important. Next generation information systems facilitating central access were designed and their use has become widespread. This has particularly accelerated data exchange between institutions and integration work between systems with different structures. Thus, system design works aiming at providing central access have increased, and data access has become easier thanks to the spread of web services.

In the context of this study, a new integrated information system model was designed to be used in providing access to fast, accurate and reliable information for distributed systems. This model protects the principles of interoperability, is based on a hybrid structure, accommodates multiple systems and enables multifaceted data flow by providing integration through web services. It aims to analyze, manage and process data based on the concepts of time and space in the long run.

The model was built to represent the conceptual core of the information system works, and to identify the most accurate data and have the most reliable information in the case of data repetition between institutions. It is known that data repetition results in inconsistent analysis results. Therefore, this integrated information system is based on compliance with international standards. In accordance with the principle of interoperability, not only storing and/or using data according to the same standards, but also enabling the sharing and common use of data by adopting a central information/data system design is highly important.

The model and system prototype designed in the thesis study "Development of an integrated information system model: DataOCEAN©" is based on service oriented architecture (SOA) while it shares similarities with object oriented architecture (OOA). The framework used in the design is an original work based on model-view-controller architecture. It was designed using REST (Representational State Transfer) style. Web services in relation to RESTful and SOAP (Simple Object Access Protocol) service architectures were prepared; areas to be included in the management panel of the system were written and tested. PHP (Hypertext Preprocessor) was chosen as application language while using GoogleMAPs for spatial data representation.

The architecture the web services are integrated in the study is referred to as DataOCEAN© - DO while the web services library is referred to as OceanLIBRARY© - OLY.

In DataOCEAN© system model, data units are web services. These web services are in the process of integration with the system library and defined with the hierarchical formation used in object based programming (with features such as data, service

structure etc.). Data query is dynamically conducted through the web interface coded as part of the application. After the original query is entered in web interface, web services to be used from the data pool during the query processing are identified. Information is collected from the services that can be instantly accessed and have the highest reliability of data, and the result is presented in the relavent format in the web interface. Without knowing the institution and/or structure that is the source of the data, the user know that the result is up-to-date and has the highest accuracy in the query time. The user are not expected to be highly competent.

This model mainly aims at shortening the software process. As it is designed with an architecture that is suitable for sharing the data in the information systems, it is considered setting an example for interoperability. In the designed DataOCEAN© architecture, different systems carry out data exchange. With the interface presenting an original query platform, it selects service/services with the highest accuracy among the web services active in the system during the query, and finishes the operation in a short time with quick and accurate results.

The introduction part summarizes the study, applied methods and system structure, and mention studies on data and data diversity. The first part examines web services, OOA, SOA, REST, SOAP and MVC architectures as the basis of the study. The DataOCEAN© Information System Model was built on SOA architecture. This platform-independent system can be adapted to any web-accessing environment. In the DataOCEAN© structure, the user is not concerned with the resources and structures of the web services. However, the user can assume that the most accurate data available during the usage period is being presented. The web services defined in the system are saved according to the data they include and the query types to which they can respond.

The second part explains the study method, and the DO architecture and the OLY library that were developed. The web services in the library (OLY) are defined according to a hierarchy similar to the structure of the OOP Namespace (Object Oriented Programming Namespace). The interface established on the DataOCEAN© servers facilitates query production by interconnecting with SQL (Structured Query Language). DataOCEAN© also operates as a RESTful web service set up according to the REST architecture. It is a service which processes a SQL sentence submitted to the service, and then it returns the data in the JSON (JavaScript Object Notation) structure.

The model enables the web service library defined within the application to be used in the form of a table by means of SQL. The columns of the table are comprised of request and response parameters. In the service definitions located in the service library (OLY), the parameters are described together with their properties in the form of table columns.

The web service registry is made in compliance with the structure of the object-oriented architecture. In the periods after the first web service registry, the alterations in the service structure do not require any code change in DataOCEAN© operations. Only the usage manner of the service structure is updated.

The third part addresses the prototype application and results. One of the most significant contributions of this study is the logic of the service registry being generated at the software layer. In the DataOCEAN© structure, traditional URL reading and update transactions are required. The interlinked DataOCEAN© servers synchronize continuously with the service libraries. The changes made on one server

are distributed simultaneously to the other servers registered in the system. The DataOCEAN© system is intended to make use of SOA structures more efficiently during integration processes, and to shorten the software process. This system was presented with software, middleware and the implemented working environment.

The conclusion summarizes the results and indicates targeted studies and the contributions of the study. In the suggested model, every kind of system located on the web or with an output to the web can be integrated.

The written application is a candidate integration layer, which could be considered as a compulsory feature of today's information systems. In this case, it is platform - independent, and enables multi directional data flow by increasing the scope of a given system. Any freshly integrated data can be added to the unified data in a simple, fast, and easy way. However, the added data is required to have a structure which fulfills the standards of the web service. The most important innovation introduced by the study is object-oriented definition logic and the algorithm followed while adding web services to the system's library. Thanks to the suggested object-oriented definition, web services would not only be defined easily, but also would be found easily during the transaction. In addition, local changes do not affect the functioning of middleware in the suggested structure. It was observed that the designed DataOCEAN© Architecture has the capacity to conduct transactions swiftly and correctly by selecting the service/services which has the highest correctness level among the web services active within the system at the moment of inquiry. This is possible due to the interface, which provides a distinctive query platform. In this architecture, different systems also interchange data. In the metadata, the section in which the structures are integrated, any updates can be carried out instantaneously system-wide. In the long term, archive data can be generated via this application. A service oriented, time dependent model which was designed to be convenient for fast data sharing with the last user, was also formed using the architecture.

Since we preferred diversity in line with the purpose of the study and price policy in the course of thesis work, hardware design and data safety layer was not taken into consideration.

# 1. GİRİŞ

21.yüzyılda Bilişim teknolojisi her bireyin her türden bilgiye ulaşımını kolaylaştırarak imkanlı kılmaktadır. Kurum ve kuruluşlar, aynı veya yüksek benzerlikte veriler ile çalışma konuları doğrultusunda kurulan sistemleri kullanarak hizmet vermektedir. Veriler birbirinden farklı ortamlarda ve yerel sistem yapılarında yer almaktadır. Her kurum, sistem yapısını özgün ve kendi ihtiyaçları doğrultusunda dizayn etmektedir. Bu durum, sistemin işleyişini kolay ve sorunsuz yürütülmesini sağlasa da, diğer sistem yapılarından soyutlanmış olarak çalışan bu özgün sistemde çalışmanın dezavantajları da olmaktadır. Örneğin, kurum içi veriler ile birlikte başka kurumların ürettiği, ancak işlem sırasında ihtiyaç duyulan veriler de tutulmaktadır. Bu durum, atıl veri oluşumuna, uzun vadede ise sistem yoğunluğuna neden olmaktadır. Ayrıca, bu atıl sistemler veri doğruluğu ve güncelliğini de zamanla yitirerek veri tutarsızlığına yol açmaktadır.

Kullanımı kolay ve güncel veri içeren sistemler kurma hedefi, günümüz uzmanlarının en büyük çabaları arasındadır. Güncel veri; herhangi bir konuyla ilişkili, erişilip sunulabilecek en son veri olarak tanımlanmaktadır. Özellikle, rapor ve istatistiksel yapılarda bilgi güncelliği çok önemlidir. Güncel veri ulaşım ve sürekliliği ise, ancak, verileri üreten kurumlar arası birimlerin bir biri ile iletişimde olup veri entegrasyonu sağlayarak gerçekleşebilir.

Bilgi paylaşımı kültürünün oluşması ile birlikte, son yıllarda kurumlar arası veri bütünlüğü sağlamak mümkün olmuştur. Veri, kademeli ve kontrollü olarak ortak kullanıma açılmış, önceden belirlenen ve bilinen yapıda erişimi sağlanmıştır. Sistemlerin gelişimi sürdükçe kurallar belirlenmiş, yapı oturtulmuş, sistemler arası haberleşme standartlar çerçevesinde sağlanmıştır. Veri paylaşımı ve kullanımının ilk dönemlerinde, bilgi kontrolü ve işlem takibi açısından dış sistemden kullanılan verinin bir kopyasını yerel sisteme de aktarma eğilimi söz konusu olmuştur. Zamanla, kontrol altına alınamayan veri artışına, bir de tedbir amacıyla tekrar içeren veri kaydı eklendiğinde, veri kirliliği ve sistem boyutunun büyümesi sorunlarıyla karşı karşıya kalınmıştır. İnternet erişebilirliğinin hızla yaygınlaşması, kullanım ücretlerinin de

düşüşü ve veri iletişim hızının artmasına paralel, kullanıcı sayısı da muazzam biçimde artmıştır. Sayısı giderek artan internet bazlı mecraların çoğalması ile teknolojiye erişimin kolaylaşması ve yaygınlaşması nitelikli veri kavramının anlamını da arttırmıştır. Dolayısıyla, nitelikli veri ve bu veriye erişim süreci yavaşlamış ve veri takibi zorlaşmıştır. Ayrıca, kontrolsüz veri ekleme, inceleme ve veriyi kullanma aşamasındaki düzensizlik atıl emek oluşumuna da neden olmaktadır. Özetle; durdurulamayan veri artışı, büyüyen sistem boyutlarına, veri kirliliğine, veri kalitesinin azalmasına neden olmaktadır. Bu amaçla birlikte çalışabilirlik ilkelerine dayanarak standartlar oluşturulup, çeşitli uygulamalar yazılarak sistemlerin birbiri ile haberleşmesi sağlanmaktadır.

Günümüzde, kurumlar arası ve kurum içi etkili iletişim için, veriye web ortamından erişim sağlanması bir devrim niteliğindedir. Bilgi sistemlerinde birleştirilmiş veri ihtiyacını karşılayabilecek entegre sistemlere gereksinim giderek artmaktadır. Birleştirilmiş sistemler [1-3], veri ambarları [4-6] gibi artık geleneksel kabul edilebilecek yaklaşımlar, son yıllarda büyüyen veri ve bilgi hacmi dolayısıyla ihtiyaçları karşılamakta yetersiz kalmaktadır. Ayrıca, bu sistemlerin, sürekli güncellenme ve bakım ihtiyacı da, onların sürdürülebilirliğini giderek daha zor ve maliyetli hale getirmektedir [7, 8]. Bu kapsamda, kurumlar arası veri paylaşımı gereksinimi; yeni nesil bilgi sistemlerinin tasarımını ve merkezi tasarım zorunluluğunu ortaya çıkarmıştır [9-11].

### 1.1 Tezin Amacı

Yeni nesil "Entegre Bilgi Sistemleri" kavramı özünde web servislerini içermektedir. "Merkezi Tasarım" ise, verinin, web üzerinden, belli standartlar kapsamında bulunduğu, iletilip işlendiği bir yapı belirlemektedir. Web servislerinin en çarpıcı özelliği platform bağımsız olmalarıdır. Bu özellik, kullanım kolaylığı sağlarken kullanım kısıtı da oluşturmamaktadır. Web servisleri ile kurumlar içerisinde veya kurumlar arasında veri kaynaklarının paylaşımı ve kullanılması kolaylıkla sağlanmaktadır.

Bu çalışmada, nitelikli veri oluşumu, paylaşımı, kullanımı ve web servislerinin sunduğu avantajlar ile, merkezi sistem tasarımı için bir öneri getirilmektedir. Hazırlanan merkezi veri paylaşımı sistem modeli ile, web servislerinin haberleşme işlemi nesne yönelimli mimari yapısı temel alınarak veri kaynaklarının paylaşılması

ve birlikte kullanılması sağlanmaktadır. Model, “DataOCEAN” - Veri Okyanusu olarak adlandırılmıştır.

“DataOCEAN” - Veri Okyanusu modelinde, veri kaynakları çok çeşitli platformlarda bulunan, birbirinden farklı zamanda üretilmiş ve farklı karakterde ve doğrulukta oluşturulmuş sistemlerdir. Sorgulama anında erişilebilir olan, ihtiyaç halinde bağlantı kurulan ve kullanımdan sonra iletişimi kesen, web veri erişim servisleridir. Bu servisler, yapı ve ortam değişimlerinden etkilenmeden, kullanılan uygulamaları ve sistemleri entegre etmektedir. DataOCEAN modelinde, web servisleri nesne yönelimli mimariye uygun tanımlanır ve sorgulanırlar. Nesne yönelimli mimari tanımlaması ile yapı ve ortam değişimlerinden etkilenmemeleri sağlanmaktadır. Tanımlama şekli ile lokal sistemde yapılan değişiklikler genel sistemi etkilememektedir. Servis tanımı, OceanLibrary olarak adlandırdığımız sistem kütüphanesinde yapılmaktadır. Sistemde, kesintisiz veri sorgulama işlemi kütüphanede yer alan web servislerin birbirinin yerini dinamik olarak alması ile sağlanmaktadır.

Kesintisiz, sürekli veri iletişiminde DataOCEAN sisteminde bir türden veriye ulaşmak için birden fazla ve farklı konumlarda bulunan servislerin entegre edilmesi ile sağlanmaktadır. Belli türden veri için birden fazla veri mevcut olduğunda, sistem daima en güncel ve doğruluk değeri en yüksek olanı tercih etmektedir. Ancak, bir fiziksel kesinti halinde, sorgulama esnasında kullanıcıya aksettirmeksizin, bir alt doğruluktaki servise yönlenerek veriyi elde eder, sorgulama sonucunu da anlık olarak sunar. Data Ocean modelinde, yapısal değişiminin veya güncelleme ihtiyacının oluşması, hatta sürekli hale gelmesi, sistem bütünlüğünü veya çalışmasını etkilememektedir.

DataOCEAN sistemini tasarlama sürecinde hızlı büyüyen ve oldukça tutarsız olan veri havuzundan nitelikli veriye hızlı erişim amaçlanmıştır. İncelenen çalışmalar ve sistemler sonrasında bu amaca sadece sistem ve veri kalitesini arttırarak ulaşılamayacağı görülmüştür. Bu nedenle, en büyük kazanım web servislerinin sistem kütüphanesine tanımlama sürecinde elde edilmiştir. Yapılan çalışma ile dağıtık sistemler arası merkezi veri erişimi ve entegrasyon sorununa çözüm getirmiş, yazılım sürecini kısaltmış ve kullanım kolaylığı katmıştır.

## 1.2 Hipotez ve Yöntem

"Entegre Bilgi Sistemi Modeli: DataOCEAN" başlıklı tez çalışması ile önerilen sistemde özellikle hedeflenen dinamik, anlık sorgu oluşturma, ilgili veriye ulaşma ve kesintisiz süreci devam ettirme hedeflenmiş ve uygulanmıştır. DataOCEAN özellikleri;

- Entegrasyon yazılım katmanında sağlanmaktadır.
- Merkezi veri erişimi sunar. Tüm servislere erişim merkezidir.
- Veri ve veri kaynaklarının türü ve yapısı özgündür.
- Platform bağımsızdır. Her sisteme kolayca entegre edilebilmektedir. Bu amaçla bir arabirim yazılmıştır.
- Yazılım kolaylığı sunmaktadır.
- Geniş kullanıcı kitlesi tarafından kullanılabilir. Kullanıcı katmanında yazılım bilgisi gerektirmemektedir.
- Kullanım kolaylığı sağlamaktadır.

Sistemin gerek şartı, modelde yer alan yapıların tamamının internet çıkışı olmasıdır. Platform bağımsız olduğundan bu anlamda bir kısıtlama getirilmemektedir. İkinci önemli şart ise; her veri için hazırlanan servisin web servisi standartlarını sağlayan yapıda olmasıdır.

**Çalışmanın en önemli yeniliği web servislerini sistem kütüphanesine eklerken izlenen nesne tabanlı tanımlama mantığı ve algoritmasıdır.** Önerilen nesne tabanlı tanımlama sayesinde web servisleri yalnız kolay tanımlanmakla kalmaz, işlem sırasında kolay bulunur ve local değişiklikler önerilen yapıdaki ara katman işleyişini etkilememektedir. Entegrasyon arabiriminde web servisi şeklinde sunulabilen her türden verinin birbiri ile entegre edilme olanağı vardır. Yazılan arabirim, sistem kapsamı arttırılabilen, çok yönlü veri akışına olanak sağlayan, günümüz ihtiyaçlarını karşılayabilcek bir entegrasyon katmanıdır. Dağıtık bulunan bilgi sistemleri veya sisteme yeni entegre olan her türden veri arasındaki haberleşme işlemi, basit, hızlı ve kolay bir şekilde birleştirilerek veriye eklenebilmektedir. Burada, veri sorgulama, sistemler birbirinin yapısını bilmek zorunda olmaksızın, yazılımsal bir ara katman aracılığı ile tek bir veri alanından veri sorgularmış gibi gerçeklemektedir. Veri ve veri kaynaklarının türü ve yapısı özgündür. Sistemler arası entegrasyon yazılım katmanında, yazılım arabirimi eklenerek sağlanmıştır.

Yazılım arabirimi statik ve dinamik metrikler aracılığı ile incelenmiş, sonuç raporlar Ekler bölümünde sunulmuştur.

### 1.3 Sistem Özeti

Hazırlanan entegre bilgi sistemi modeli; web servisleri ile nesne tabanlı programlamayı anımsatan bir tanımlama yapısı ile sistem kütüphanelerine bir kereye mahsus tanımlanır (içerdiği veri, servis yapısı, v.b özellikler). Sonrasında, tek bir arayüzden veri havuzunda bulunan web veri servisleri üzerinde veritabanında bulunan farklı tablolardan veri sorgulaması anımsatan biçimde sorgulama yapılır.

Yazılan uygulamanın içerdiği temel mantık aşağıdaki adımlar ile kısaca açıklanabilir;

- Veritabanın SQL sorgu alanı yapısına benzeyen, web arayüzünde bulunan alana, özgün sorgu yazılır.
- Sorgu, Data Ocean yapısına yönlendirildikten sonra işlenir. Çözümleme işlemi başlar. SQL sorgusunun anlamlaştırılma işlemi sorgu çözümleyicisi ile sağlanmaktadır.
- Çözümlenen sorgu, aşamalı olarak önce veri edinebileceği web servislerini, sonrasında ise talep edilen veri yapılarını tespit eder.
- İlgili servis aktif durumda ise sorgulama sonucu alınır ve kullanıcıya web arayüzü aracılığı ile aktarılır.

Sistemin yapısı her türden web servisi entegre edebilme yeteneğine sahiptir. İnternet çıkışı sunabilen her platformda kesintisiz olarak hizmet verebilir. Eklenen demografik veri servisleri ile birlikte, günümüzde vazgeçilmez olan konumsal veri servisleri de uygulama kapsamına dahildir. İsteğe göre DataOCEAN yapısı başka bir uygulamaya gömülerek (embed) de kullanılabilir. Verilerin; talep üzerine, sorgulama anında, dinamik olarak başka bir alanda birleştirilmesi, sonuç üretmesi ve görüntülenmesi hedeflenmiş ve gerçekleştirilmiştir.

Model, servis yönelimli mimariye uygun olarak tasarlanan, web servisi temelli bir veri entegrasyon modeli önerisidir. Bu, dağıtık sistemler arasındaki iletişimi sağlayan; sorgulama anında erişilebilir olan, ihtiyaç halinde bağlantı kurulan, kullanımdan sonra iletişimi kesen, web veri erişim servisleridir. Yapısı, boyutu, platformu birbirinden farklı olan sistemlerin bulundurduğu verileri bir arada çalıştırıp, analizler yapılıp, sonuç raporlar elde etme amacıyla yazılmış, entegre sistemlerde etkin kullanmayı

hedeflemiş, tasarlanmış, uygulaması yapılmıştır. Bu doğrultuda DataOCEAN örnek uygulaması kurgulanmış ve yazılmıştır. Bu uygulamada Web servislerinde kullanılan verilerin tamamı gerçeğe uygun üretilmiştir. İstanbul ilçelerinde planlanan kentsel dönüşüm çalışmaları ve bu kapsamda yeni yapılara olabilecek talep ve potansiyel alıcı kitlesi belirlemek amacıyla çeşitli kurumlardan bilgiler entegre edilerek bir tahmin sonuç üretilmesi amaçlanmıştır. Bu kurguda yeni yapılara yakın ikamet eden kişiler, ilgili kişilerin bankadaki bilgileri, önceden belirlenmiş hesap bakiyesi tutarı, olanların kredi notu, tanıtımın yoğunlaşacağı hedef kitlenin belirlenmesi gerçeklenmiştir.

Bu modelin uygulamada büyük bir gereksinimi karşılayacağı düşünülmektedir.

# 2. ENTEGRASYONDA KAPSAM

## 2.1 Amaç

Yazılım mimarileri, 1990 yılından itibaren Perry ve Wolf'un [12] da öngördüğü gibi, yazılım mühendisleri için bir odak noktası olmuştur. Yazılım mimarilerinin, modern sistemlerin karmaşıklığı, çok hızlı veri artışı, uygulama çeşitliliği, bilişim sistemlerin uygun bölümleme ve düzenlemeler ile ayrı ayrı sistemler olarak değil de, modüler olarak birbiri ile iletişime geçmeleri amaçlanmıştır. İyi bir ürün veya ürünler zinciri kuvvetli bir altyapı mimarisi ile ortaya çıkabilir. Sürekli gelişen ve ihtiyaçları hızla artan bilişim sistemleri birbiri ile performans kaybı olmaksızın kolaylıkla entegre olmasının sağlanabilmesi bu çalışmanın motivasyonu olmuştur.

Son yıllarda, geleneksel uygulama mimarilerinin izlenmesiyle, sistemlerin büyüme hızına yetişmek veya mevcut yapıları belli bir tasarım mimarisine uyarlamak için zaman problemi ortaya çıkmaktadır. Problemin çözümü için, üst bir yapı ya da tüm sistemleri kapsayacak bir katman hazırlanmasının, uygun olacağı düşünülmüştür. Öncelikli amaç olarak, verinin, performans kaybına uğramadan ve iş gücü artışı gerektirmeden, birbiri ile entegre edilebilmesi hedeflenmiştir.

Servis yönelimli ve nesne tabanlı yazılım mimarileri, güvenli veri akışı sağlayan web servisi mantığı ile birleştirildiğinde, ideal entegre yapının ortaya konulabileceği öngörülmüştür. Bu kapsamda, web servislerinin nesne tabanlı tanımlama yapısı ile sisteme tanıtılan, her türden sistemi birbiri ile haberleştirebilen bir ara katman mimarisi tasarlanmış ve uygulaması yapılmıştır. Bu tasarımda, Servis Yönelimli Mimari (SOA) benimsenmiş, kullanılan web servislerin yapısı MVC mimarisi ve REST ve SOAP yaklaşımı temel alınarak kodlanmıştır. Web servislerin yazılım katmanı PHP ile yazılmış ve veritabanı olarak SQLite - MemoDB kullanılmıştır.

Bu bölümde, ilgili mimari yapıları, web servisleri, veri tabanı ve entegrasyon bileşenleri hakkında bilgi aktarılacaktır.

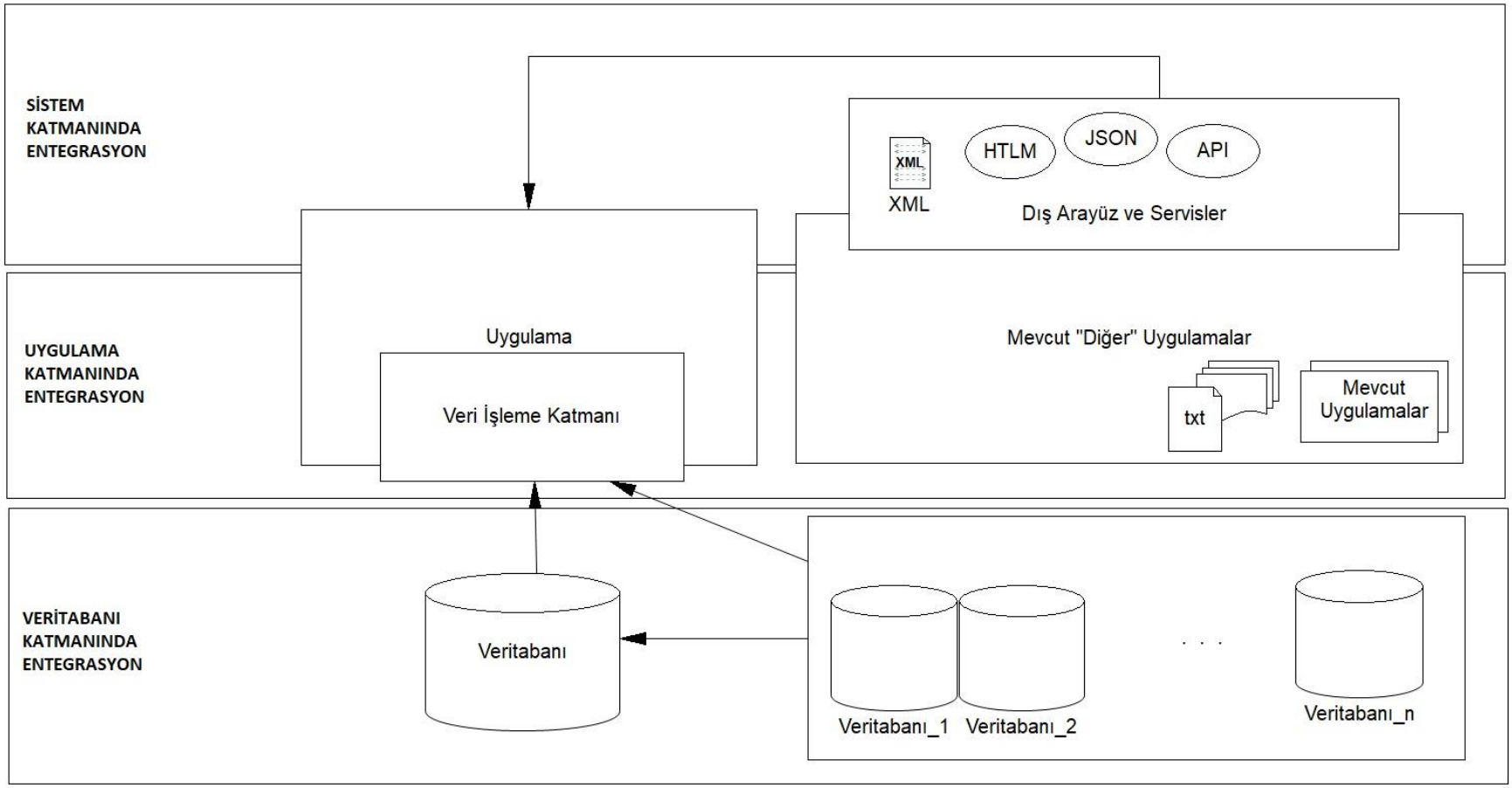
## 2.2 Veri ve Veri Entegrasyonu

Entegrasyon; intégration, Fransızca'dan dilimize geçmiş bir kavram olup, birleşme, bütünleşme, uyum sağlama anlamı taşımaktadır [13, 14]. Entegre olmak, belli işlevler, kümeler, varlıklar arasında bağ kurmak şeklinde de tanımlanabilir. Entegrasyon, insanlığın varoluşundan itibaren kullanılan, günümüz biliminin de her alanında yer edinen bir kavramdır. Toplum bilimleri, Tıp dünyası ve Finans sektöründe de sıkça söz edilen entegrasyon kavramı, dikey ve yatay olarak tanımlanan iki şekliyle birçok bilim alanında da incelenmektedir [15].

Bilişim dünyasında "entegrasyon" ise, sistemler arasında birleşimleri, uyum süreçleri ve geçişleri tanımlamaktadır. Entegre çalışan bir örnek bilgi sistemi modeli Şekil 2.1.de gösterilmektedir.

Bilişim sistemlerindeki entegrasyonun temeli veridir. Veriyi kendinde barındıran sistemler ise entegrasyon yönlendirmesini yapan sistemlerdir (entegrasyon türünü belirleyen). Veri barındırma kavramı, *kaliteli* veri barındırmayı kapsamaktadır. Veri, her sistemin en temel, en tabandaki katmanında bulunur. Başta sistem ve uygulama tasarımları olmak üzere, diğer bitün sistem "*inşaa*" işlemleri onun üzerinde yapılmaktadır. Amaç veri kullanımında nihai veri gösterimini sağlamaktır. Sistem inşaa süreci ile düşünecek olursak, sistem oluşumu için üç temel bileşene ihityaç duyarız; fiziksel olarak verinin barındığı ortam, veritabanın kendisi ve veriyi görselleştirdiğimiz uygulama arayüzüdür. Dolayısıyla, entegrasyon süreci de bu üç temel bileşenle ilglidir; entegrasyon, sistem katmanında, uygulama katmanında ve veritabanı katmanında olmak üzere üç katmanda gerçekleştirilebilirdir [16, 17].

Şekil 2.1. de gösterilen alt katmanda çok sayıda veritabanı yer almaktadır. Her bir veri tabanı, ayrı birer bilgi sisteminin veri katmanları olarak değerlendirilebilir. Bu durum, sembolik olarak n-türden veri barındıran, m-türden veritabanı sistemi için kurgulanmıştır. Burada yer alan ve ayrı ayrı yapılarda tanımlanan veri ve veri yapıları genellikle her sistemin kurulma aşamasında oluşturulmaktadır. Amaca uygun olarak veri edinilir ve uygun bir veri tasarım modeli ile barındırılacakları yapıda kaydedilir ve işlenirler.




**Şekil 2.1 :** Entegrasyon Katmanları.

Her kurum ihtiyaç duyduğu verileri kendi bünyesinde barındırır, lokal sistem ihtiyaçları için kullanır ve güncelliğini sürdürür. Giderek çoğalan sistem sayısı, veri tekrarı sonucu oluşmuş atıl veri boyutunu artırmaktadır. Bu nedenle, farklı sistemlerdeki verileri paylaşıma açma ve birleştirme çabası da artmaktadır. Birbirinden bağımsız, farklı amaçlarla kullanılan iki sistem ele alındığında, ayrışımlarıda, birleşimleride veri üzerinden sağlanmaktadır. Ayrıştırmayı birleşmenin üssü gibi değerlendirdiğimizde ise, entegrasyon, her iki özelliği de içermektedir. Dolayısıyla, birbirini bütünler iki kavram gibi değerlendirilebilir.

Konuyla ilgili birçok çalışma, veride en temel ve kalıcı birleşmenin veriyle/veride yapılması gerektiğini göstermektedir. Örneğin Oracle veritabanı, "Exadata" sunucusu uygulamasıyla büyük veri barındırmada bir çığır açmıştır. Ancak dağınık sistem, veri çeşitliliği, bu kapsamda "gizli veri" kavramını da, bu çözümleri de yeri geldiğinde yetersiz kılabilmektedir. Özetle, veritabanı sistemleri her geçen gün büyümekte ve gelişmektedir. Fakat, gelinen veri hacim yapısında verinin yeniden düzenlenmesi, ortak bir veritabanında barındırılması olanaksız hale gelmiştir ve günümüzün veri değişim/gelişim ve büyüme hızında yetersiz, maliyetli ve yavaş kalmaktadır.

Bennet ve Ghezi'ye göre, yeniden veri tasarımı ve sürekli bir üst platforma değişim çabası her zaman uygulanılabilir bir işlem değildir [7, 18, 19]. Sistem katmanında yapılan değişiklikler, veri ve veri türlerini iyileştirmek ve yeniden yapılandırmak adına maliyetli ve zaman açısında uzun süren işlemlerdir [20, 21] Sistem ve veritabanında entegrasyon gerçekleme işlemleri yüksek maliyetli, oyalayıcı ve zordur [8]. Yeniden veritabanı tasarımı yerine veri sistemlerinde standartların oluşturulmasını dolayısıyla da ortak kullanımın mümkün kılınması hedeflenmiştir.

Birleşik/Entegre sistemler genellikle bazı özel durumlara cevap vermek durumundadırlar. Örneğin, önceleri veri ve veritabanı dendiğinde demografik veri içerikli sistemler akla gelmekteyken, şimdi, konumsal verinin de bilişim sistemlerinde tanımlanıp yer almasıyla, entegrasyonda kullanılabilir veri kapsamı genişlemiştir. Hatta, veri önem merkezi *konum*'a doğru taşınmıştır.

Konumsal veri depolama çalışmaları 1950'li yıllarda başlamış, sonraki 50 yıl gelişimini sürdürerek, 2000'li yılların başında kullanılmaya başlanmıştır [22-24]. Veri madenciliğinde mekansal-konumsal analizler ile daha nitelikli veri işleme, analiz ve

sentez çalışmaları da gerçekleştirilmiştir [25]. Konuma bağlı veri kullanımı ile Coğrafi Bilgi Sistemleri-CBS(GIS) kullanılmaya başlanmıştır [26]. Teknolojinin gelişim hızı ile coğrafi veri kullanımı akademik, kurumsal ve pratik hayatta yaygınlaşmıştır. Özellikle, mobil uygulamalarla birlikte her yaşta ve her gelişmişlik düzeyinde insanın yaşamı kolaylaştıran veriye ulaşımı sağlanmıştır. Bu doğrultuda birleştirilebilir sistem türleri çoğalmış, birlikte çalışabilirlik alanı genişlemiştir.

Birlikte çalışabilirlik ilkeleri günümüz sistemlerinde önemli bir yere sahiptir. Konumsal veri kullanımının yaygın hale gelmesiyle dünya birbiri ile daha kolay iletişebilir ve haberleşebilir olmuştur. Mekansal verinin, geleneksel ilişkisel veritabanlarına aktarımı konusunun gündeme gelmesiyle veritabanı yapıları da gelişim göstermiştir. Bu bağlamda, özel standartlar ortaya konulmuş, tasarlanmış ve birlikte çalışabilirlik ilkeleri belirtilmiştir [27, 28].

Veriye zaman boyutunun da eklenmesi ile veri yapıları her türden veriyi barındırabilmekte ve birlikte çalıştırabilmektedir. Sistemler arası veritabanı odaklı entegrasyon zaman ve maliyet açısından kayıplara neden olmaktadır.

**Uygulama katmanında entegrasyon,** veritabanı katmanında entegrasyona göre gerçekleştirilmesinin kısa zaman alması bakımından, avantajlı görünmektedir. Örneğin, Pennie Araştırma Grubu tarafından (Pennine Research Group) *Hizmet Olarak Servis* - SaaS (Software as a Service) kavramı önerilmiştir [29]. Bu yaklaşım, yazılımı bir hizmet olarak kullanmayı önermektedir. Bir başka görüş ise; Enterprise Service Bus-İşletme Servis Sürücüsü – ESB ile ortaya konmuştur[30]. Anlaşılmıştır ki, sadece ilgili uygulama bazında yapılan yeniliklerin (veritabanı yapıları ilk oluşturulmuş haliyle kalmaları, sistemin tamamının uyum sağlayamaması) sistemler arası entegrasyona direkt katkıda bulunamamaktadır. Dolayısıyla, en kapsamlı ve kolay uygulanabilir olan **Uygulama katmanındaki entegrasyon**dur. Yazılımların da, yazılım bileşenlerin de, aynı veriler gibi standartlaşması ve paylaşım platformlarının oluşması entegrasyon çalışmalarını kolaylaştırmıştır. Özellikle, web servislerinin ve ortak yazılım kütüphanelerinin kullanılabilirliği, veri paylaşımı ve veri kullanımı konusunda hız kazanmıştır.

## 2.3 SOA Mimarisi

Servis Yönelimli Mimari(SOA) yapısı; veri miktarının artışı, servis odaklı teknolojilerin gelişimi, standartlar, entegrasyon çalışmaları, işlem yönetimi ve güvenlik ilkeleri gibi birçok dağıtılmış kurumsal bilgi sistemleri gibi potansiyel problemli konuların çözümü için kolaylıklar sağlamıştır [31-34]

SOA'da *servisler*, çoklu platform ve protokollere izin verilerek, standart bir tanımlama dilinde açıklanırlar, yayınlanmış bir arayüze sahip olurlar, ortak veya benzer bir iş görevini veya işlemini müşterek olarak desteklemek üzere faaliyetlerinin yürütülmesini isteyecek biçimde birbirleriyle iletişim kurarlar [26]. Çok sayıda erişim aygıtı ve eski sistem kullanımı, web ile erişim engellerinin ortadan kaldırılması, uygulamaların sorunsuz bir şekilde entegre olmasını ve çalışmasını sağlamaktadırlar. Bu şekilde, bir SOA, işletmelerin temel yapı taşları olarak, devam eden ve değişen iş ihtiyaçlarını kolaylaştıracak şekilde bir araya getirilebilen ve yeniden kullanılabilen hizmetleri tanımlayan, işletme kullanıcılarının ihtiyaç duyacağı esnekliği ve çevikliği sunabilir. Klasik yazılım mimarilerinin aksine, işletmelerin kurumsal uygulamalar ve hizmetleri verimli ve düşük maliyetli bir şekilde geliştirmelerini, bağlamalarını ve sürdürmelerini sağlayacak bir tasarım stili, teknoloji ve süreç çerçevesi oluşturmaya odaklanmıştır [35-37]. SOA ile, modüler programlama, kodların yeniden kullanımı ve nesne yönelimli yazılım geliştirme teknikleri gibi geçmişten gelen alışkanlıklar geri plana atılmıştır. *Tasarım felsefesi olarak SOA*, örneğin Web servisleri veya J2EE kurumsal bileşenleri gibi, *herhangi bir spesifik teknolojiden bağımsızdır*.

SOA'daki servisler aşağıdaki özellikleri taşır [33];

- *SOA'da her fonksiyon servis olarak tanımlanır* [38]. SOA'nın gevşek bağlamlar (loose coupling) yani, servis arayüzlerinin dâhili uygulamalardan sorunsuz ayrılması prensibi, kurumlar arası uygulamalar için vazgeçilmez kılmaktadır [38].

- *Tüm servisler otonomdur* [30]. Web servisleri, SOA servis arayüzlerini belirgin bir şekilde tanımlayarak uygulama karmaşıklığını azaltmaktadır. Ayrıca, Web hizmetleri, tam zamanlı (just-in-time) entegrasyonu ve eski uygulamaların birlikte çalışabilirliğini sağlamaktadır. Web hizmetleri, maksimum servis paylaşımı, yeniden kullanım ve birlikte çalışabilirlik gibi SOA vaatlerini gerçekleştirmek için tercih edilen uygulama teknolojisi haline gelmiş gibi görünmektedirler[39].

## 2.4 Web Servisleri

SOA ve Web servisleri çözümleri iki temel bileşeni destekler. İlki: hizmet talep yoluyla iletişim kuran bir hizmet talep edici - istemci (client)dir. İkincisi de aynı yolla hizmet sunan hizmet sağlayıcı- sunucu(server)dur. Bu bileşenler ise, sistem katılımcının SOA'daki türünü yansıtır [34, 40].

Web servisleri sistemler arası veri akışını sağlayan sunucu uygulamalarıdır. HTTP protokolünü kullanarak ortak veri yapısında bilgi aktarırlar. Çalışma esnasında herhangi bir durum bilgisi barındırmazlar. Bilgi alınacak sistemlerin özelliklerinden, yapısından, gerçeklemesinden bilgi sahibi değildirler. Sistemler arası herhangi bir altyapı ve platform uyumu gerektirmezler. Web Servisleri aracılığı ile internette yer alan herhangi bir yapıdaki veri güvenli bir şekilde çok büyük kullanıcı kitlesi ile paylaşılabilir [41, 42].

Web servisleri için diğer bir tanımlama da W3C [41] tarafından şöyle verilmektedir: Bir Web Servisi, bir ağ üzerinden makineden-makineye (sistemden-sisteme) birlikte çalışabilir bir etkileşimi desteklemek için tasarlanmış bir yazılım sistemidir [42].

Web Servislerinin temel altyapısı ve gelişmiş veri paylaşım standartları beş ana unsuru içerir;

- XML: web servislerinin veriyi sunmakta kullandığı standarttır [41]. XML (eXternal Markup Language), web servislerinin veriyi sunmakta kullandığı, W3C (World Wide Web Consortium) [41] tarafından tanımlanmış bir standarttır. XML kolay okunabilecek dokümanlar oluşturmaya, veri saklamanın yanısıra farklı sistemler arasında veri alışverişi yapmaya yarayan bir formattır.

- SOAP: ağ destekli uygulamalar arasında veri alışverişini kolaylaştıran ve yaygın kabul gören bir standart mesajlaşma protokolüdür [43, 44].
- WSDL: makine tarafından işlenebilen ve bir hizmet arayüzünü tanımlamak için kullanılan bir formattır [45]. WSDL (Web Service Description Language) web servisini tanımlayan XML yapısında metadatadır.
- UDDI: hizmetleri tanımlama, ortaya çıkarma ve yayınlama için oluşturulan, platformdan bağımsız bir çerçevedir [46, 47]. UDDI (Universal Description, Discovery and Integration) ise web servislerinin bilgilerini barındıran ve yayınlayan sunuculardır [46].
- REST: dağıtık sistemler için kullanılan, hızlı çalışan ve kolay uygulanan bir yazılım mimarisi biçimidir [48].

Her bir bileşenin işlevini, bir Web servisinin oluşturulmasından tüketimine kadarki yaşam döngüsü boyunca belirli bir sıralama üzerinden ele alınacak olursa [49-52] aşağıdaki gibi açıklanabilir;

- Sunucu, çeşitli hizmetler oluşturmak için bir SOAP araç seti kullanmaktadır.
- Hizmetler UDDI ortak kayıt defterinde yayınlanmaktadır.
- Bir istemci bir arama ölçütü belirtir ve istenen bir hizmeti UDDI kayıt defterinde arar.
- İstemci, hizmet açıklamasını (bir WSDL belgesi) kayıt defterinde belirtilen URL'den alır.
- İstemci, sunucuya bağlanmak için WSDL'deki bağlantı bilgilerini kullanır.
- İstemci, uzaktaki makinelerde bulunan servisi API'leri üzerinden çağırmak için bir SOAP isteği oluşturur.
- Uygulama aracılığı ile, istemciler ve sunucular arasında bilgi alışverişinde bulunmak için serileştirme ve seriden paralel hale getirme yöntemiyle SOAP protokollerine çevrilir [53].

Web servisleri HTTP tabanlı oldukları için platform bağımsız ve temel bilgiye sahip her yazılımcı tarafından hızlı ve kolay geliştirilebilir yapıdadırlar. Platform bağımsız çalışabilme özelliği W3C tarafından belirlenen standartlar doğrultusunda tasarlanmışlardır. Bu çalışması kapsamında adı geçen servis türlerinden REST ve SOAP servislerinin çalışma mantığı incelenecek ve örnekler oluşturulacaktır.

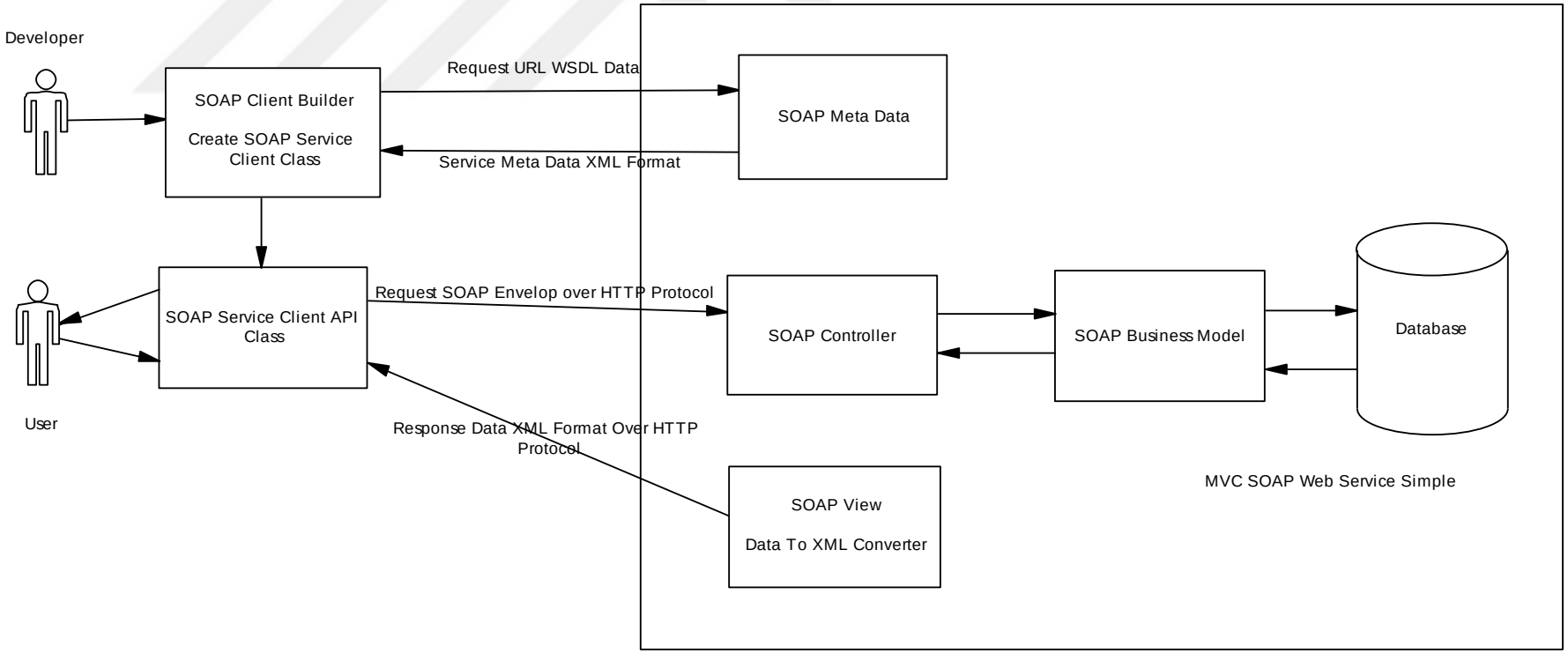
## 2.5 SOAP Mimarisi

SOAP (Simple Object Access Protocol) Basit Nesne Erişim Protokolü veri aktarım protokolüdür. HTTP tabanlı ve platform bağımsızdır. Hizmet talepleri, biçimlendirilmiş mesajlardır [44]. SOAP mesajları, bir bağlantı tanımlandığı sürece, herhangi bir protokol (HTTP, HTTPS, SMTP) kullanılarak iletilebilirler. Bir SOAP isteği, SOAP iletisini kabul eden, XML ileti gövdesini ayıklayan, XML iletisini yerel bir protokole dönüştüren ve isteği gerçek iş sürecinde temsil eden bir çalışma zamanı hizmeti - SOAP dinleyicisi, tarafından alınır. İstek yapılan Web hizmetleri bir veya daha fazla Web servisi bileşeni kullanılarak uygulanmaktadır [49]. Bir hizmet kapsayıcısı, soyut hizmet uç noktasının fiziksel bir belirtisidir ve servis arayüzünün uygulanmasını sağlar [50, 51]. Buna ek olarak, servis kapsayıcıları, başlatma, kapatma ve kaynak temizleme gibi yaşam döngüsü yönetimi için olanaklar sağlarlar. Bir hizmet kapsayıcısı, aynı dağıtılmış sürecin bir parçası olmasa bile birden çok hizmeti barındırabilir. Dizilerin toplanması (thread pooling), bir hizmetin birden çok formunun tek bir kapsayıcı içinde birden çok dinleyiciye bağlanmasına izin verir [52]. Son olarak, sağlayıcının istemciye geri gönderdiği yanıt, bir XML iletisi taşıyan bir SOAP zarfının şeklini alır. SOAP da platform ve sunucu ayrımı gözetmeyen bir standarttır. İstemci ve sunucunun farklı kuruluşlarda veya işletmelerde bulunması internet üzerinde ikisi arasında "gevşek" bir ilişki kurulmasına izin verir. Bu noktada belirtilmelidir ki, SOA, SOAP kullanımını gerektirmez. SOA hizmetleri servis istemcisi tarafından görülebilirken, temel Web bileşenleri ise şeffaftır. İstenilen hizmet kalitesi sunulurken gerekli fonksiyonellik desteklendiği sürece, istemci ve sunucu için, bileşenlerin tasarımı, hizmete sunulmaları ve yönetimi, SOA'daki hizmetlerin sunulmalarını sağlayan kilit mimari ve tasarım kararlarını yansıtmaktadır. Bir hizmet sağlayıcının doğrudan bir hizmet sağlayıcısı ile etkileşime girmesini gerektiren süreç, hizmet talep eden kişileri, farklı servis sağlayıcılar arasında hizmetleri araştırma, keşfetme, devretme ve ayırma gibi olası karmaşıklıklara maruz bırakabilir [30].

SOAP genel çalışma mimarisi Şekil 2.3.'deki şemada gösterilmiştir.

İstemci protokolü servislerin kayıtlı olduğu yapıdan ilgili web servisini bulur, hazırlamış olduğu SOAP mesajını web veya uygulama sunucusunda çalışan dinleyiciye gönderir. SOAP sunucusu gelen SOAP mesajını ayrıştırır, anlamlaştırır ve gerekli parametreleri göndererek nesnenin ilgili yöntemini çağırır. Elde edilen sonuçlar SOAP sunucusuna gönderilir. Sunucu ise gelen sonucu SOAP formatında biçimlendirerek istemciye gönderir.

SOAP istemci tarafından oluşturan istek program üzerinden SOAP servisinden gelen WSDL verisinin içerisinde bulunan servisin istek ve sonuç verilerini kullanarak SOAP servis istemci objesine dönüştürür. İstemci objesi, istek ve sonuç verilerini gönderirken veya alırken SOAP mesajının içerisine yerleştirir ve okur. Kullanıcı SOAP istemci objesini kullanarak istek verilerine göre web servisten hangi tipte veri istediğini talebini yapar. SOAP servisinin ana kontroleri istekte bulunulan verilerin neler olduğunu bulup, uygulanacağı kısma aktarır. Yapılan isteğe göre veritabanından gelen veriler filtrelenerek, kontroller sonrası ilgli alana geri döner. Kontrol katmanı gelen veriyi anlamlandırmak için veriyi XML yapısına çeviren katmana aktarır. Veri XML'e çevirildikten sonra istemciye geri döndürülür. İstemci objesi de gelen veriyi kullanıcıya iletir. Dolayısıyla, SOAP yapıları her türden veri alışveri yapabilme kabiliyetine sahiptir.




**Şekil 2.2 :** SOAP Mimari Yapısı.

## 2.6 REST Mimarisi

XML, SOAP ve WSDL gibi açık standartlara dayalı olan Web servisleri, "gevşek" bağlantı, kesintisiz birlikte çalışabilirlik ve iyi ölçeklendirilebilirlik avantajları sağlamaktadırlar. Ayrıca, web servisi protokolleri aynı zamanda fonksiyonel ve fonksiyonel olmayan gereksinimleri de karşılamaktadır. Son yıllarda, Apache HTTP Server projesinin de kurucularından olan Roy Fielding'in 2000 yılında tamamladığı doktora tezinde [48] REST modelini ortaya koymasından sonra; RESTful Web Servisleri, basitliği, ölçeklenebilirliği ve performansı nedeniyle geniş bir ilgi alanı kazanmıştır.

REST (Representational State Transfer), HTTP protokolü ile çalışan, dağıtık sistemler için kullanılan, hızlı çalışan ve kolay uygulanan bir yazılım mimarisi biçimidir. WWW (World Wide Web)'nin de temel mimarisini oluşturmaktadır.

REST terimi, Henry (2011)'ye göre REST tek başına bir mimari değil, sistem tasarımına uygulanırken çalışmanın gerektirdiği kural ve kısıtlamaları içeren bir yazılım uygulamasıdır. REST mimarisi Şekil 2.4.'de sunulmuştur.

Fielding [48] doktora çalışmasında REST'in kural adı altında bildirilen tasarım ilkelerini aşağıdaki gibi tanımlamıştır:

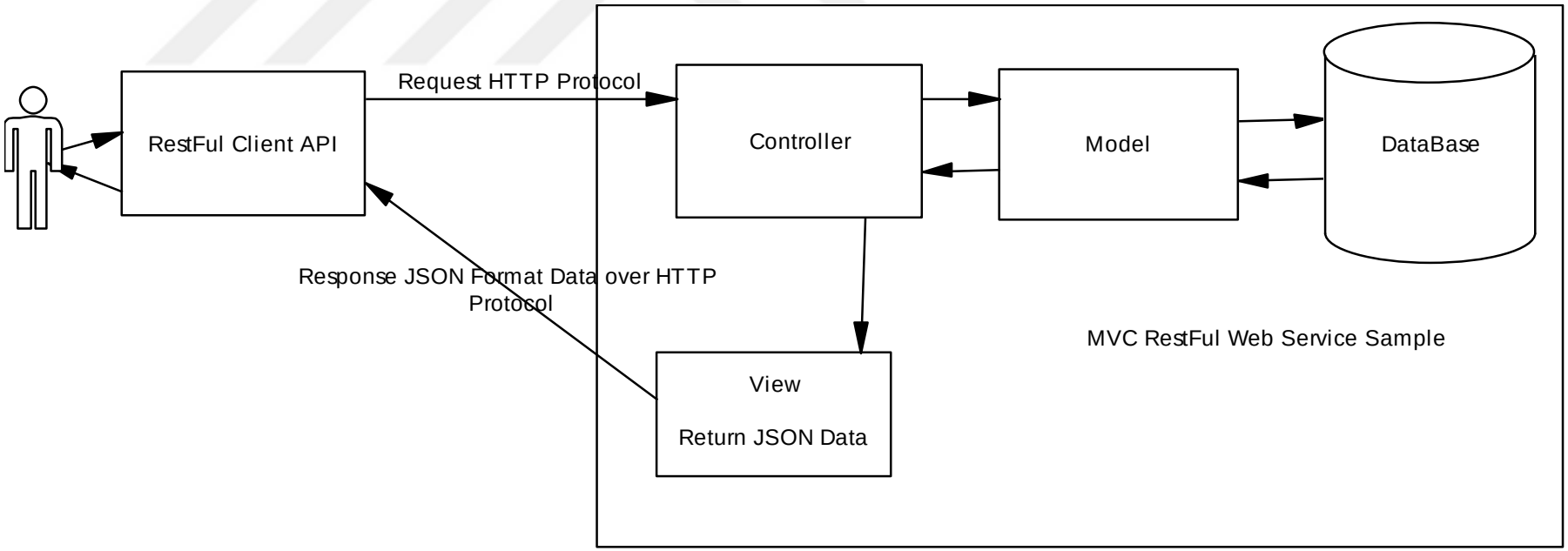
- *İstemci-Sunucu (Client-Server)* mimarisinde, sunucu ve istemci birbirlerinden ayrı ayrı kendi görevini üstlenerek sistemde yer alırlar. İstemcinin veri kaynağı hakkında bilgi sahibi olmaması, sunucudan doğru istekler geldiğinde yanıt vermesi gerekmektedir.
- *Durum Bilgisi (Stateless),* her türlü durum bilgisi istemci tarafında tutulur. Sunucudaki bilgiyi istekler dahilinde istemci taşır.
- *Önbellekleme (Caching)* daima ağ performansını arttırır. Burada da, istemci veriyi bellekleyebilme ve sunucuya istediği anda tekrar sunabilme özelliğine sahiptir.
- *Tek biçimlilik (Uniform Interface),* istemci ve sunucuda tek biçimli olan arayüz iletişimini kolaylaştırmaktadır.

- *Katmanlı sistem (Layered System),* istemcinin, istediği sunucuya bağlanma özgürlüğünü sağlar. Arada başka sunucuların yer alabilirliğini destekler ve fakat her katmanın tek bir katmanı bilmesini gerektirir.
- *Talebe göre kod (Code-On-Demand)* kuralı opsiyoneldir. Sunucunun gerektiği durumlarda istemci tarafında yazılım kodları çalıştırabilirliğini vurgular.

REST, platformdan bağımsız ve sistem kapsamını yükselterek, çalışmayı ve sunucu üzerindeki durum bilgisi barındırma özelliğini kaldırarak, önbellekleme ağ performansını arttırarak, kullanıcıya daha hızlı yanıt vermeyi amaçlamaktadır. Tek biçimliliği sağlayan ortak arayüz ise, iletişim yöntemini basitleştirip, her parçanın birbirinden bağımsız gelişmesine olanak sağlamaktadır. Katmanlı sistem nedeni ile güvenlik konusu ve yük dağılımı durumlarında kapsam artışı gözlemlendiğinden veri transferinde gecikme oluşumu konuları geliştirilmeyi gerektirir.

REST mimarisine uygun oluşturulmuş ve Restful Web Servislerini kullanan sistemler, kullanılan dilden bağımsız, bilgi alışverişini XML (eXtensible Markup Language) ya da JSON (JavaScript Object Notation) formatında yaparlar [54]. RESTful Web hizmetleri son zamanlarda en çok kullanılan web veri sunum standartıdır. Kullanıma sunulan API'lerin %81'ini oluşturmaktadır. Bu istatistikler, APIs dizinlerinin en popüler web sitesi olan www.programmableweb.com'dan izlenebilir[55]. Dağıtık sistemlerin geliştirilmesi için RESTful hizmetlerini kullanmak bir eğilim haline gelmiştir.

RESTful web servisleri "basit" bir hizmet olarak algılanması, bilinen W3C/IETF standartlarını ve zaten yaygınlaşmış olan gerekli altyapıyı kullanmasındandır [54]. RESTful web servislerinin temel prensibi, belirli bir zaman anındaki eşlemeye karşılık gelen varlık değil, bir takım varlıklara kavramsal bir eşleme olan kaynaktır[48]. Bir RESTful web servisini çağırmak, dinamik bir web sitesine erişime benzetilebilir bir işlemdir.




**Şekil 2.3 :** REST Mimari Yapısı.

RESTful web servisleri, aşağıdaki özelliklere ve avantajlara sahiptir [48, 56]:

- *Adreslenebilirlik* (Addressability): Kaynaklar URI'larla işaretlenir.
- *Link ve Bağlanabilirlik* (Links and Connectedness): Kaynaklar, gösterimlerde geçerli olan gelecek durumlara işaret eden köprüler kullanılmak suretiyle birbirleriyle bağlantılandırılırlar.
- *Durumsuzluk* (Statelessness): Her bir istek, sunucuların algılaması için gerekli tüm bilgileri içermektedir, bu nedenle, her bir işlem birbirinden bağımsız ve önceki işlemlerle ilgisizdir. Sunucuların istekler arasındaki durumları saklamasına gerek yoktur[57, 58].
- *Tek tip arayüz* (Uniform Interface): Kaynaklar, HTTP metotlarının sabit bir kümesi kullanılarak, özel bir kod olmadan her biri ile temas etmek üzere, işlenir. Bu yöntemler (POST hariç) eş kuvvetlidir.

RESTful web servislerinin, mevcut web standartlarına dayanarak, dağıtımı, yayınlanması, bulunması ve çalıştırılması kolaydır. Kaynaklar açıkça sunulmakta ve dinamik olarak Web üzerinden hesaplanmaktadır. Bu hizmet sunucularının esnekliği ve "gevşek" bağlantıları hem istemciler hem de sunucular için son derece avantajlıdır [59].

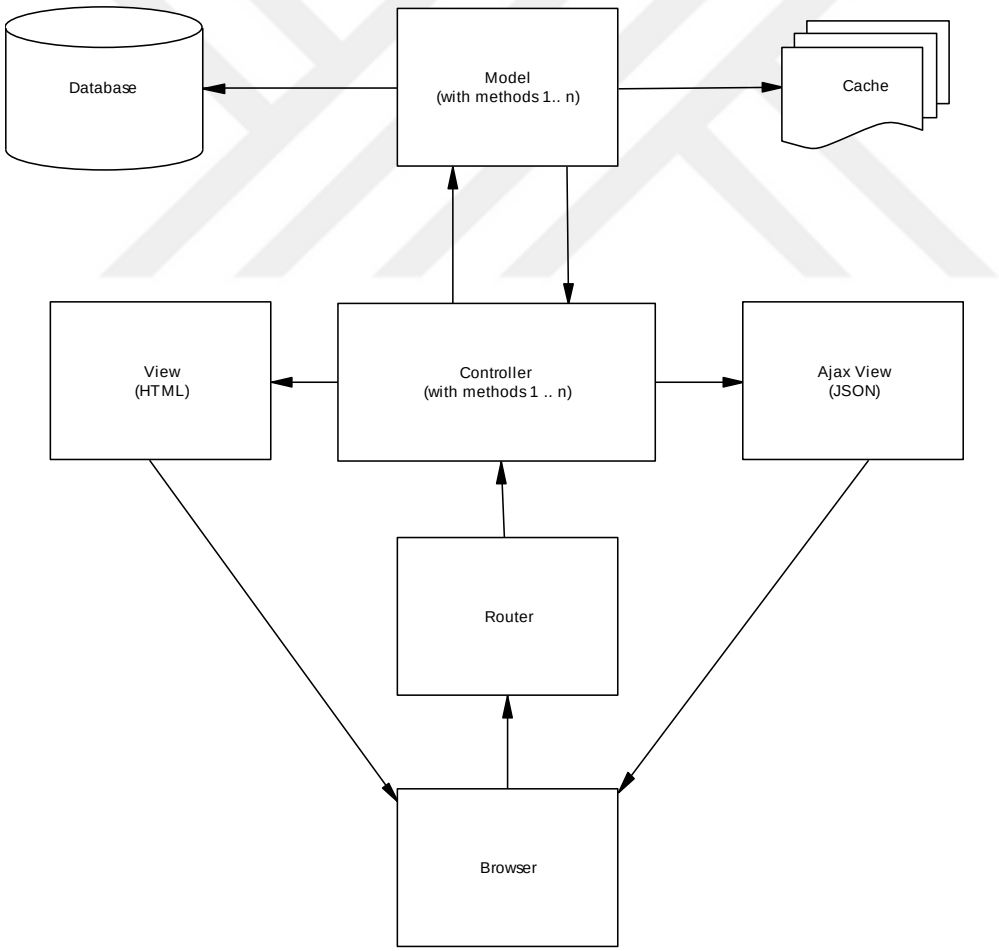
## 2.7 MVC Mimarisi ve xElis Framework

Çalışmada kullanıcı arayüzü web tabanlı MVC (Model-View-Controler) mimariye uygun yeni bir framework tasarlanmıştır. Tasarlanan framework xElis olarak adlandırılmıştır. xElis framework özellikleri aşağıdaki gibi sıralanabilir;

- *Model Katmanı* (Model): Uygulamanın iş mantığının ve veri tabanının yer aldığı katmandır. Veritabanı bazında önceden oluşturulan veya bilinen yapılar (view, stored procedure, function v.b.) burada yer alır.
- *Görsel Katman* (View): Son kullanıcıya hizmet veren arayüzlerin bulunduğu katmandır. İşlenen verilerin web uygulamaları aracılığı ile kullanıcıya sunulur. Php kodlarının, HTML sayfaları, css dosyaları, resimler vb. bu katmanda yer alır.
- *Kontrol Katmanı* (Controller): Kullanıcının isteklerini ilgili katmanlara ileten sınıfların bulunduğu alandır. Uygulamanın çift yönlü karar mekanizmasıdır.

Şekil 2.4.'de tasarlanan MVC mimari çekirdek model yapısı incelenebilir. Genel anlamda anlatılan çekirdek mimari tasarımı, xElis framework'ünde uygulanmıştır. Entegre bilgi sistemine uyarlandığında, veri geçiş yolları HTML veya JSON dur.

Sistemde merkezde bulunan *dispatcher(router),* gelen isteğin hangi veri kaynağından karşılanacağına karar veren yazılımdır. Önce, kontrol katmanında bulunan *controller'dan* cevap ister ve ilgili alana yönlendirilir. Her controller yapısı her işlem için ayrı bir metot modeli içerir. Controller metot aracılığı ile *model* katmanına erişerek, model katmanındaki metot bileşenler ile ilgili veriye ulaşıp, sonucu sunum katmanına döndürür. Model katmanı, içerdiği veri ve veri türlerini sunar. Sunum katmanında sorgu sonucu izlenir, kullanıcı tarafından kaynağı bilinmemesine rağmen, veri anlık ve günceldir.




**Şekil 2.4 :** MVC Mimari Yapısı.

# 3. ENTEGRE BİLGİ SİSTEMİ MODELİ: DataOCEAN

## 3.1 Giriş

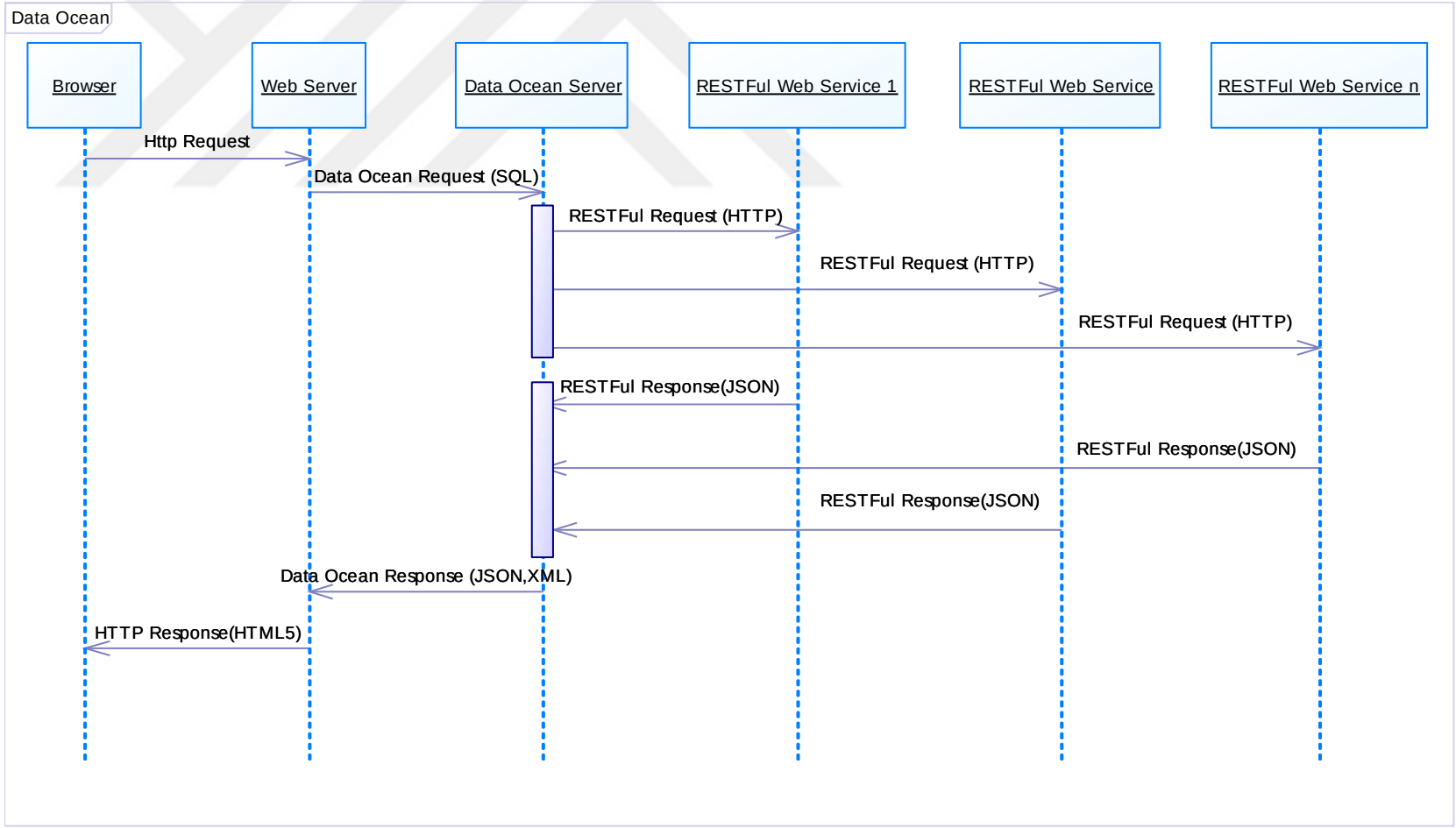
"Entegre Bilgi Sistemi Modeli Tasarımı: DataOCEAN" başlıklı doktora tezi çalışması kapsamında tasarlanan modelde ve uygulanan yapıda veriler birbirinden çok farklı sistemlerde yer almasına rağmen, tek bir veri merkezinde bulunuyormuş gibi, merkezi bir yapıdan veri sorgulaması gerçekleştirmektedir. Web servisleri üzerinden sağlanan erişim, sorguları ilgili veri kaynaklarına gönderilerek, elde edilen veri sonucunu kullanıcıya sunmaktadır. Web servislerinin yönlendirildiği DataOCEAN (Veri Okyanusu) merkezi bilgi kontrol katmanı tasarlanırken hızlı veri girişi ve veriye hızlı ulaşma kuralları korunmuştur.

## 3.2 DataOCEAN Sistem Modeli

DataOCEAN Bilgi Sistemi Modeli[60] (Şekil 3.1.), SOA [40, 61-64] mimarisi üzerine kurulmuş, web'e çıkışı olan, her ortama uyarlanabilir, platform bağımsız bir modeldir(Şekil 1).

DataOCEAN yapısında kullanıcı, sistemde yer alan web servislerinin kaynakları ve yapılarıyla direkt ilişkili değildir. Fakat, sorgulama zaman dilimi içinde, en doğru kabul edilen veriyi kullandığını bilmektedir. Sistemde tanımlı web servisleri içerdikleri verilere ve cevap verebilecekleri sorgu tiplerine göre tutulmaktadır.

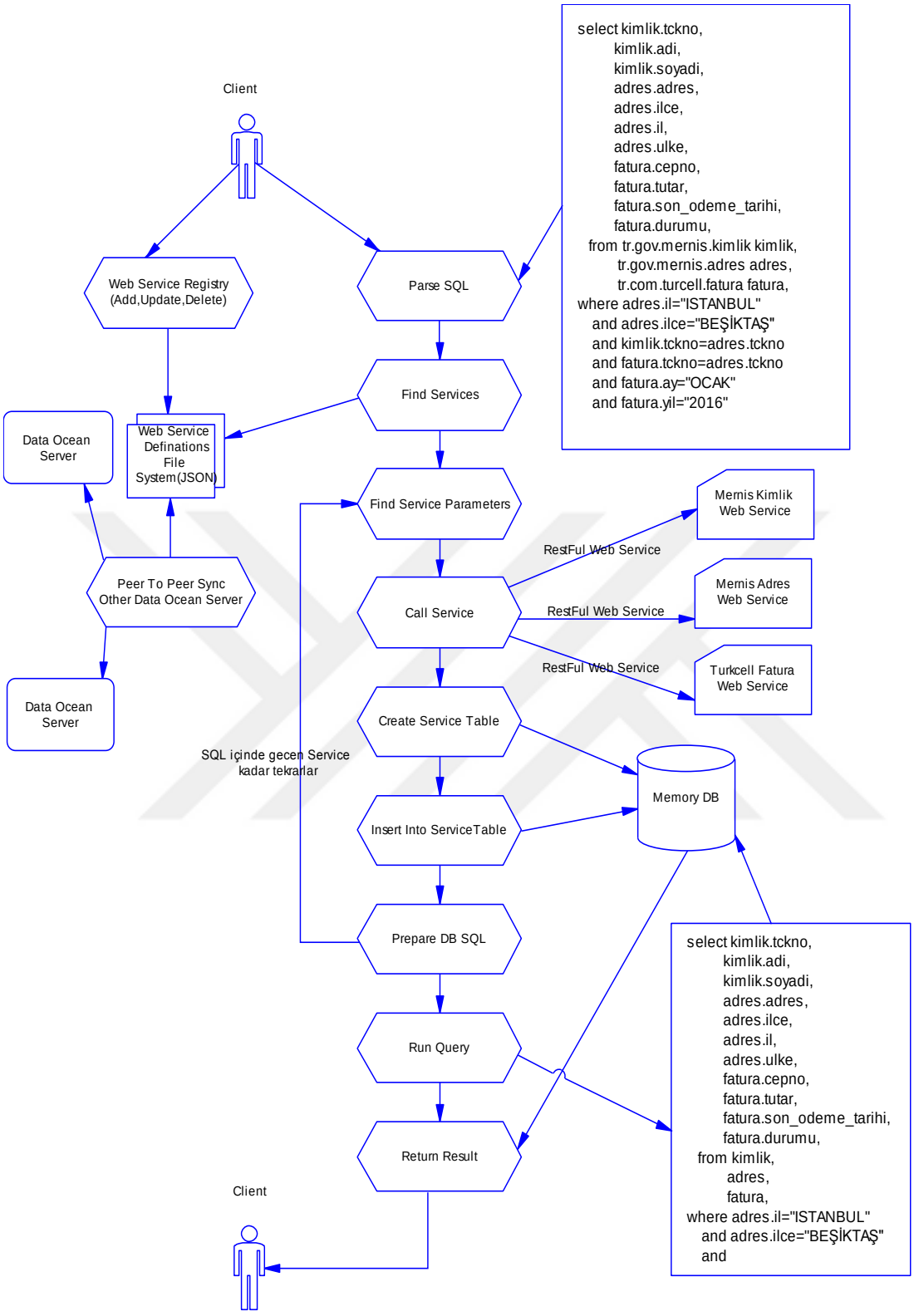
- Web servislerinin OceanLIBRARY-(OLY) kütüphanesindeki tanımı, OOP Namespace (Object Oriented Programming Namespace) yapısını anımsatan bir hiyerarşi güdülerek tanımlanmaktadır.
- DataOCEAN sunucularında kurulu olan arabirim, SQL (Structured Query Language) ile birbirine bağlanarak kolay sorgulanmasını sağlayan bir arabirimdir.




**Şekil 3.1 :** Data OCEAN Sistem Modeli.

DataOCEAN modeli REST mimarisine göre hazırlanmış bir RESTful web servisi olarak çalışmaktadır. Sisteme gelen sorgu -sql cümleciğini alınır ve sistem dinamiği içinde işlendikten sonra JSON yapısında veri döndürür.

DataOCEAN Modeli, uygulama kapsamında tanımlı olan web servisi kütüphanesini tablo şeklinde, sql aracılığı ile kullanılmasını sağlar. Tablonun kolonları olarak servise gönderilecek (request) parametreleri ve döndürülecek (response) parametreleri kullanılır. OLY servis kütüphanesinde yer alan servis tanımlarında, tablo kolonları şeklinde bu parametreler tanımlanır. DataOCEAN sistem modelinin çalışma prensibi Şekil 3.2'de ayrıntılı olarak ifade edilmiştir. Şekilde iki kez yer alan **İstemci** figürü işlem akışı sıralı izlenebilmesi açısından işleyişin başında ve sonunda birer ikon olarak eklenmiştir.




**Şekil 3.2 :** Data OCEAN Çalışma Mimarisi.

3.2.1 İstemci (Client).

İstemci, herhangi bir talep unsuru niteliğindedir. Kullanıcı olabileceği gibi, küçük bir uygulama olabilir ve/veya içine web servisleri gömülmüş daha büyük bir sistem de olabilir. İstemci, DataOCEAN yapısından bilgi talebinde bulunduğunda işleyiş başlar. İstemci tarafından talep edilen bilgi yapısı Transact SQL sorgu yapısındadır. Bu yapı aşağıdaki unsurları içermelidir:

- SQL sorgusunda bulunan SELECT ... FROM ... WHERE ... yapısı korunmalıdır.
  **select** * **from** *net.egnar.ws.kimlik* kimlik **where** kimlik.adi='EGNAR'; (3.1)
  Burada veritabanı tablosu görevi alan servisler *Nesne Tabanlı Programlamanın İsim Uzayı* tanımlama (Object Oriented Programming Namespase) yapısı benimsenerek, DataOCEAN kütüphanesine tanımlanmış olmalıdırlar.
- Sorgunun FROM alanında yer alan servislerin DataOCEAN standardında tanımlandığı gibi isim uzayı yapısıyla (3.2) yazılmalı ve mutlaka boşluk içermeyen takma adı (alias) tanımlarının olması gerekmektedir.
  **from** *net.egnar.ws.kimlik* kimlik (3.2)
  isim uzayı (namespace) tanımı ve takma adı (alias) tanımı
- Sorgunun FROM alanında yer alan servislerin sıraları sistemde çağırılacak web servislerin sırasında olmalıdır (3.3).
  **from** *net.egnar.ws.kimlik* kimlik,*net.egnar.ws.adres* adres (3.3)

Kimlik ve adres servisleri kullanılacak ise (3.3), ilk önce *kimlik,* daha sonra *adres* servisi çağırılır. Birden fazla servis çağırılma durumlarında, birbirine bağımlı veri içeren yapılarda servisler sıralı olarak tanımlanmalıdır. Örneğin; *adres* bilgisi ile *kişi* bilgisi bağımlı olduğundan, önce, kişi bilgisi içeren servise bağlanılıp, bilgiler alındıktan sonra adres bilgisine talep yapılmalıdır.

- Sorgunun WHERE alanında yazılan tüm kolon isimleri sırasıyla [alias adı].[kolon adı] şeklinde olmalıdır.
- Sorgunun WHERE kısmında servisin çağırılması (*request*) için gerekli parametreler, her zaman eşitliğin *solunda*, parametreye gidecek (*response*) değer ise eşitliğin *sağında* olması gerekir.
- Sorgunun WHERE kısmında çağırma (*request*) parametrelerinde kullanılacak operatörler sadece [=] ve [in (1,2,3) vb.] kullanılmalıdır.

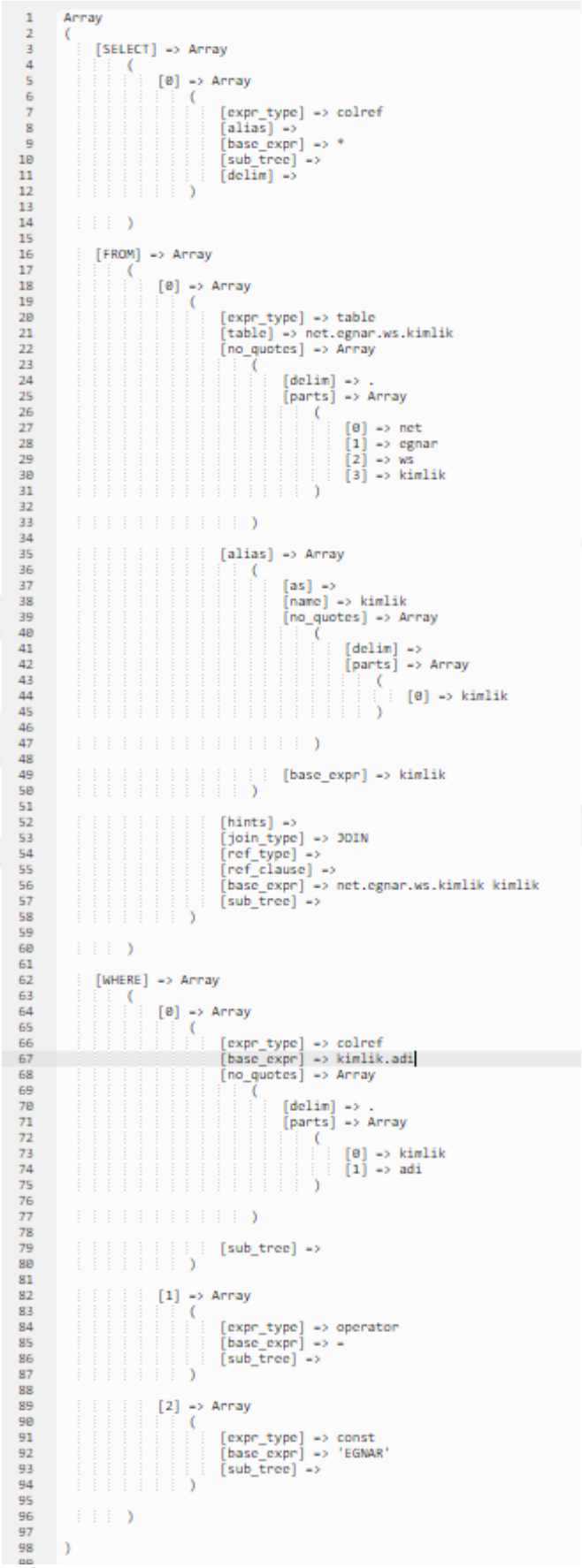
- Sorgunun WHERE kısmında çağırma (*request*) parametrelerinde kullanırken operatörün *sağ* tarafında *sabit* bir değer, sabit değerler kümesi veya başka servisten dönen (*response*) kolon kullanılabilir.

### 3.2.2 SQL Parser (Sorgu Ayrıştırıcı).

SQL sorgulamalarını ayrıştıran bölümdür. Şekil 3.2.'de ParseSQL olarak görünen alanın sağ tarafında örnek SQL sorgusu yer alır. İstemcinin gönderdiği bu vb. SQL cümlelerini ayrıştıran bölümdür.

**select** * **from** *net.egnar.ws.kimlik* kimlik **where** kimlik.adi='EGNAR'; (3.4)

Örneğin, (3.4)'de gibi sorgu yönlendirildiğinde; *select*, *from* ve *where* alanlarında yer alan bilgiler ayrıştırılır. İşlemin Debug Mod görünümü Şekil 3.3. de incelenebilir;




**Şekil 3.3 :** SQL Ayrıştırma (Debug mod).

- Öncelikle *from* alanında yer alan ve bilginin alınacağı servis tespit edilir (3.5).

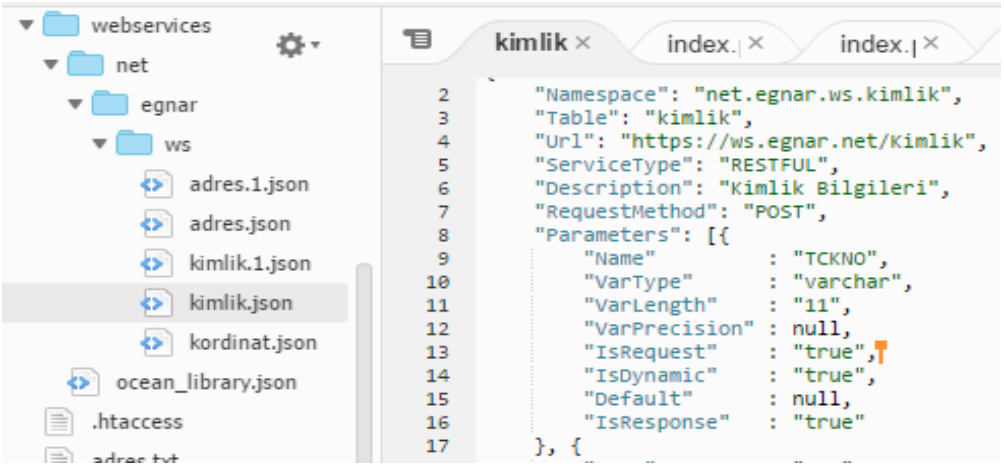
  *net.egnar.ws.kimlik* (3.5)
- Web servisi adından sonra **alias –> [`kimlik`]** bilgisi alınır.
- Sonrasında **`where`** alanında aranan bilgi [`kimlik`].[`adi`] ve sorguda getirilen kısıtlama, örnekte [`'EGNAR'`] alınır.

Örnekte sadece bir servis sorgulandığından, işlem bir kez yapılmıştır. Birden fazla servis sorgulama için bir döngü ile işlem tekrarlanır, ilgili veri belirlenir ve sonlanır. İşlem bir sonraki servis bulma katmanı ile devam eder.

### 3.2.3 Find Service (Servis Bulma).

Servis Bulma (Find Service) katmanında Parse SQL alanında ayrıştırılan bilginin içindeki FROM kısmı alınarak, içinde geçen web servis isim veya isimleri servis kütüphenesi görevi üstlenen, DataOCEAN server'ında yer alan JSON dosyasından okunarak ayrıntılı bilgiye ulaşılır.

DataOCEAN modelinde belirlenen yapıya göre tanımlanan ve dosyada yer alan servis tanımı Şekil 3.4 de ifade edilmiştir. Şeklin sol tarafında servisin isim uzayı (namespace yapısı), sağ tarafta ise kimlik.json dosyasının servis parametreleri yer almaktadır.




**Şekil 3.4 :** JSON Dosyası Görünümü (kimlik.json).

### 3.2.4 Find Service Parameters (Servis Parametrelerini Bulma)

Servis parametreleri bulma işlemi JSON dosyası okunduktan sonra bulunan tüm servisler için where alanında yer alan parametrelerin **`"IsRequest"`** parametre

özelliği taranır ve değeri "**true**" ise sorguya dahil edilir, işlem devam eder. Örnekteki, *net.egnar.ws.kimlik* servisi için bilgiler Şekil 3.5.'de incelenebilir;


**Şekil 3.5 :** JSON Dosyası Parametre Bilgileri (kimlik.json).

**"IsRequest":"true"** olarak bulunduktan sonra, parametre olarak gönderilebileceğini onaylar. Daha sonra, sorguda bulunan kısıtlamanın değerini alır.

Örnekte, **kimlik.adi='EGNAR'.**

3.2.5 Call Service (Servis Çağırma).

Web servisini çağırma aşamasıdır. Servis türüne göre (REST veya SOAP) web servisi çağırılır.Web servisi tür bilgisi servis kütüphanesi tanımlarından **"ServiceType"** alanından alınır (Şekil 3.6.).


**Şekil 3.6 :** JSON Dosyası Servis Bilgileri (kimlik.json).

Bu alanda servis ile ilgili diğer bilgilere de erişilir; Request metodu, URL bilgisi, servis ile ilgili varsa herhangi bir açıklama gibi. Örnekte çağırılan servis türü **"ServiceType": "RESTFUL"** olarak görülür.

3.2.6 Service Table (Servis Tablosu Oluşturma).

Tasarlanan arabirim aracılığı ile istemciden gelen ve web servisleri aracılığı ile cevaplanması hedeflenen sorgu cümleciğinin tamamlanması için, alınan alias ismi ile bir tablo yaratılır. Şekil 3.6.'de görüldüğü gibi **"Table": "kimlik"** örnekte alias

adı [kimlik], tablo adı da "kimlik" olarak oluşturulur. Önceden tanımlanan web servisleri ile birlikte parametre bilgileri de alınarak arka planda Şekil.3.7 gibi bir tablo oluşturma sql cümleciği oluşturularak çalıştırılır.

```
CREATE TABLE kimlik(
                TCKNO varchar(11),
                ADI varchar(100),
                SOYADI varchar(100),
                ANNE_ADI varchar(100),
                BABA_ADI varchar(100),
                ANNE_KIZLIK_SOYADI varchar(100),
                DOGUM_YERI varchar(50),
                DOGUM_TARIHI date,
                MEDENI_DURUMU varchar(30)
               );
```

**Şekil 3.7 :** Servis Tablo Oluşturma [kimlik].

3.2.7 Insert Into Service Table (Servis Tablosuna Veri Ekleme).

Servis Tablosuna Veri Ekleme katmanında DataOCEAN arabiriminde bulunan, gelen talebi, mevcut procedür (stored procedure) aracılığı ile web servisinden alınan bilgi, DataOCEAN yapısındaki igili alias adı ile MemoryDB de oluşturulan tabloya aktarılır.

3.2.8 Prepare DB SQL (Veritabanına SQL Hazırlama).

Veritabanına sorgu hazırlama katmanında istemcinin gönderdiği SQL bilgisinin

```
select * from net.egnar.ws.kimlik kimlik where kimlik.adi = 'EGNAR';
```
(3.6)

içinden servislerin isim uzayı bilgisi silinir (*net.egnar.ws.kimlik*) ve MemoryDB de çalıştırılacak olan SQL oluşturulur;

```
select * from  kimlik where kimlik.adi='EGNAR';
```
(3.7)

3.2.9 Run Query (Sorgu Çalıştır).

Bu katman nihai verinin alındığı kısımdır. MemoryDB'de oluşturulan tabloya sorgulama yapılır. PrepareDB SQL kısmında hazırlanan SQL çalıştırılır (3.7).

3.2.10 Return Result (Sonuç Döndür).

Sorgulama sonrası oluşturulan sonuç veri istemciye JSON formatına çevrilip hazırlanır ve gönderilir.

**"IsRequest":"true"** olarak bulunduktan sonra, parametre olarak gönderilebileceğini onaylar. Sonrasında sorguda bulunan kısıtlamanın değerini alır.

**kimlik.adi='EGNAR'.**

### 3.3 DataOCEAN Web Servisi Tanımlama Arayüzü (Service Registry).

DataOCEAN web servisi tanımlama alanında dört ayrı işlem yapılmaktadır. Bunlar sırasıyla servis ekleme, servis güncelleme, servis silme ve servis senkronizasyonudur.

3.3.1 Servis tanım yapısı.

Nesne yönelimli mimari yapısına uygun olarak web servis tanımları yapılır. İlk tanımlamadan sonraki dönemlerdeki servis yapısındaki değişikliklerde DataOCEAN çalışma şeklinde kod olarak bir değişiklik gerekmemektedir. Sadece servisin yapısı kullanım şekli güncellenmiş olur.

Bu çalışmanın en önemli katkılarından yazılım katmanında yapılan servis tanımlama mantığıdır (Şekil 3.8). DataOCEAN yapısında geleneksel URL okuma ve güncelleme işlemleri yapılması gerekmez. Birbirine bağlı DataOCEAN sunucuları birbirleri üzerindeki servis kütüphanelerini sürekli senkronize ederler. Değişen bir servis diğer sunucularda da güncellenmiş olur.

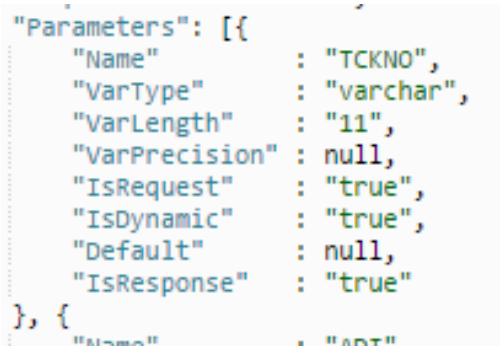

**Şekil 3.8 :** Servis tanım dosyası alanı ve isim uzayı bilgisi.

Servis tanım dosyası yerleşim kuralları(location)**:** Servis tanımının yer aldığı JSON dosyası, sistemde bulunan "webservices" dizininin içindeki alt dizinlere servisin isim uzay(namespace)'ının adlandırma sırasına göre yerleştirilmelidir. DataOCEAN, servis tanım dosyasını namespace'de sıralaması görünen uç dizinde arar.


**Şekil 3.9 :** Servis tanım bilgisi.

Servis Tanımı: Şekil 3.9'da izlenen servisin genel tanımları aşağıdaki maddelerde ayrıntılı açıklanmıştır.

- Namespace: Servisin bulunması ve SQL'lerde kullanılacak isim uzayı tanımıdır.
- Table: SQL'de alias verilmediği durumda MemoryDB'de kullanılacak tablo ismidir.
- Url: Web servisin çağırılması için kullanılacak servis lokasyon bilgisidir.
- ServiceType: Çağırılacak servisin mimari türü (RESTFUL veya SOAP)
- Description: Web servisi hakkında kısa açıklama.
- RequestMethod: Servis çağırılırken kullanılacak HTTP çağırım metodu; (GET, POST, PUT, DELETE)
- Parameters: Servisin çağırılması sırasında kullanılacak olan request parametreleri ve servisten dönecek response parametrelerinin tanımlarının yapıldığı alandır.




**Şekil 3.10 :** Parametre tanım bilgisi.

Parametre tanımı: Web servisinin hem request hem response alanında kullanılacak verilerin yapılarını bildiren tanım kümesidir. MemoryDB'deki tablo kolon isim ve yapılarıdır (Şekil 3.10).

- **`Name`** : Verinin adı.

- **`VarType`** : Verinin türü (karakter, sayı, tarih v.b.).
- **`VarLenght`** : Verinin karakter uzunluğu.
- **`VarPrecision`**: Sayısal verilerde virgülden sonraki karakter sayısı.
- **`IsRequest`** : Verinin servis çağırımında kullanılabilirliğini bildirir.
- **`IsDynamic`** : Servis çağırımında zorunlu veya zorunlu olmayan alan bildirir (true – zorunlu değil; false ise zorunlu alan belirler).
- **`Default`** : Servisi çağırırken kullanılacak zorunlu alan olduğu durumlarda ve SQLde parametre olarak girilmemiş olduğunda default değeri içerir.
- **`IsResponse`**: Servisten dönen veri olduğunu gösterir.

### 3.3.2 Servis ekleme (ADD Registry).

DataOCEAN da bulunan ADD Registry OLY kütüphanesine yeni web servisi eklenmesini sağlar. Gelen servis bilgisini okuduktan sonra, gelen servise özel JSON dosyasını oluşturur. İşlemin tamamlanması için ADD Registry servisine gelen parametre içeriği JSON formatında olmak zorundadır. Dosya oluştuktan sonra sunucu üzerinde tanımlı diğer DataOCEAN sunucularında bilgi dağılımı yapılır.

### 3.3.3 Sistemdeki servis tanımında değişiklik yapma (UPDATE Registry).

UPDATE Registry mevcut web servis tanım dosyasında yapılan değişikliklerin güncellenmesi için kullanılan RESTful servisidir. Servise gelen veri JSON formatında olmalıdır. JSON içinde geçen isim uzayı bilgisine bakarak hangi servisin güncellemesi olduğunu tespit edip ilgili servis tanım dosyası güncellenir. Bu servis hem kullanıcıda, hem de sunucular arasında kullanılır. DataOCEAN sunucularında değişiklik yapıldıktan sonra diğer sunuculara da dağıtım yapılır.

### 3.3.4 Servis silme (DELETE Registry).

Web servis tanımının sistemden silinmesi için çağırılan RESTful servistir. Servise gelen veri JSON formatında olmalıdır. JSON içinde geçen namespace'e bakarak hangi servisin silinmesi gerektiği tespit edilip, ilgili servis DataOCEAN servis kütüphanesinden silinir. Bu servis hem kullanıcıda, hem de sunucular arasında kullanılır. Bu değişiklik yapıldıktan sonra diğer sunuculara da bilgi dağıtımı yapılır.

### 3.4 DataOCEAN Servis Senkronizasyonu (DataOCEAN Server Sync)

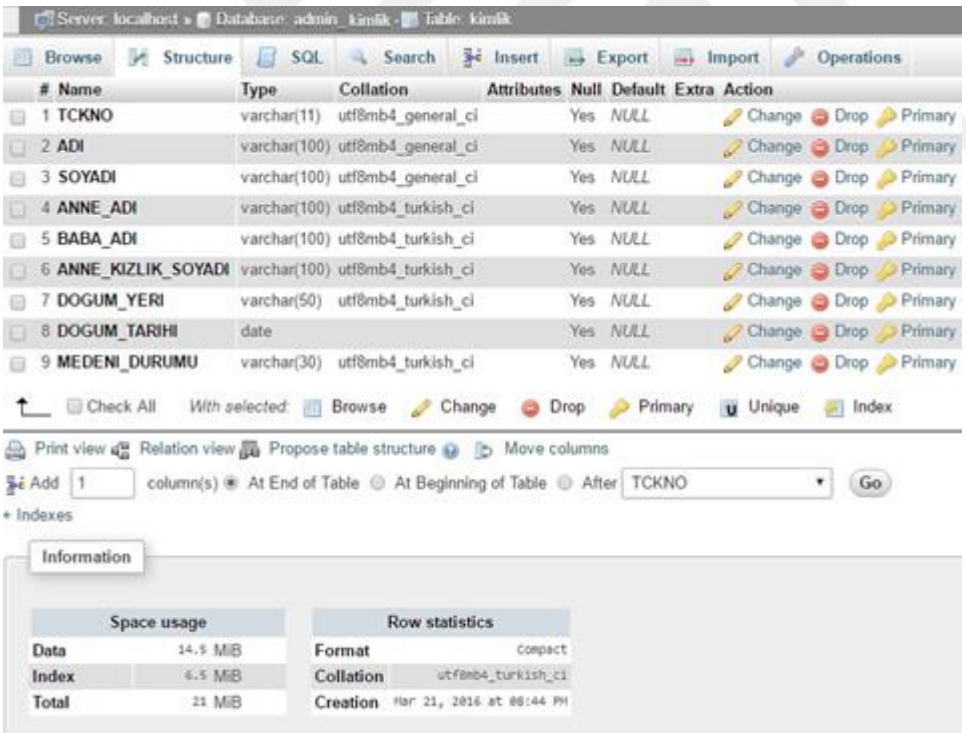
Sunucu üzerinde yapılmış değişiklikler anında sistemde tanımlı diğer sunuculara dağıtılır. Sisteme entegre olan yapılar birbiri ile senkronize tutulur.

### 3.5 DataOCEAN Server'da Tasarlanan Örnek Web Servis Yapıları.

DataOCEAN uygulama kapsamında tasarlanan servis yapılarından bazıları, Kimlik, Adres ve Konum, bu bölümde açıklanmıştır.

3.5.1 Kimlik.

Her sistemin kullandığı temel web servisi KİMLİK olduğundan uygulamada da birincil servis olarak belirlenmiştir. Kimlik Bilgi Servisi ülkemizde yaygın olarak kullanılan Kimlik servisindeki yapıya istinaden hazırlanmıştır. Bütün servislerde kullanılan **veriler gerçeğe uygun üretilmiştir.** Kimlik web servisinin kullandığı kimlik tablosunun yapısı Şekil 3.11'de incelenebilir.




**Şekil 3.11 :** Kimlik tablosunun görünümü.

Şekil 3.12'de kimlik servisi içinde kullanılan bir sorgulama örneği verilmektedir. Üçüncü satırda kullanılan değişken tanımı dinamik sorguda (dynamic query) kullanılabilen alanları olduğunu işaret etmektedir. Web servisine gönderilen TCKNO,

Adi, Soyadi alanları parametre olarak servise geldiğinde sorguya eklenir, gelmemesi durumunda sorgudan çıkartılır.

```
define('DB_QUERY_IDENTIFICATION',
'select * from kimlik where 1=1 '
        .'{{TCKNO,and TCKNO=:TCKNO}}'
        .'{{ADI,and ADI like :ADI}}'
        .'{{SOYADI,and SOYADI like :SOYADI}} limit 100000');
```

**Şekil 3.12 :** Kimlik sorgu örneği.

```
public function Kimlik(){
  header('Content-Type: application/json; charset=utf-8');
  $Db=new Database();
  $Data=array();
  if(isset($_POST['TCKNO'])){
    $Data['TCKNO']=$_POST['TCKNO'];
  }
  if(isset($_POST['ADI'])){
    $Data['ADI']=$_POST['ADI'].'%';
  }
  if(isset($_POST['SOYADI'])){
    $Data['SOYADI']=$_POST['SOYADI'].'%';
  }
  $Result=$Db->Select(DB_QUERY_IDENTIFICATION,$Data);
  echo json_encode($Result,JSON_UNESCAPED_UNICODE);
}
```

**Şekil 3.13 :** Kimlik web servis örneği.

Şekil 3.13'de kimlik web servisinin kodu görünmektedir. Parametrelerden istek olarak gelip, veri kümesine eklenen parametreler görülmektedir. Sonrasında, kimlik sorgusu veritabanında çalıştırılır, sonuç istemciye JSON formatında gönderilir, JSON formatındaki tanım dosyası oluşur.

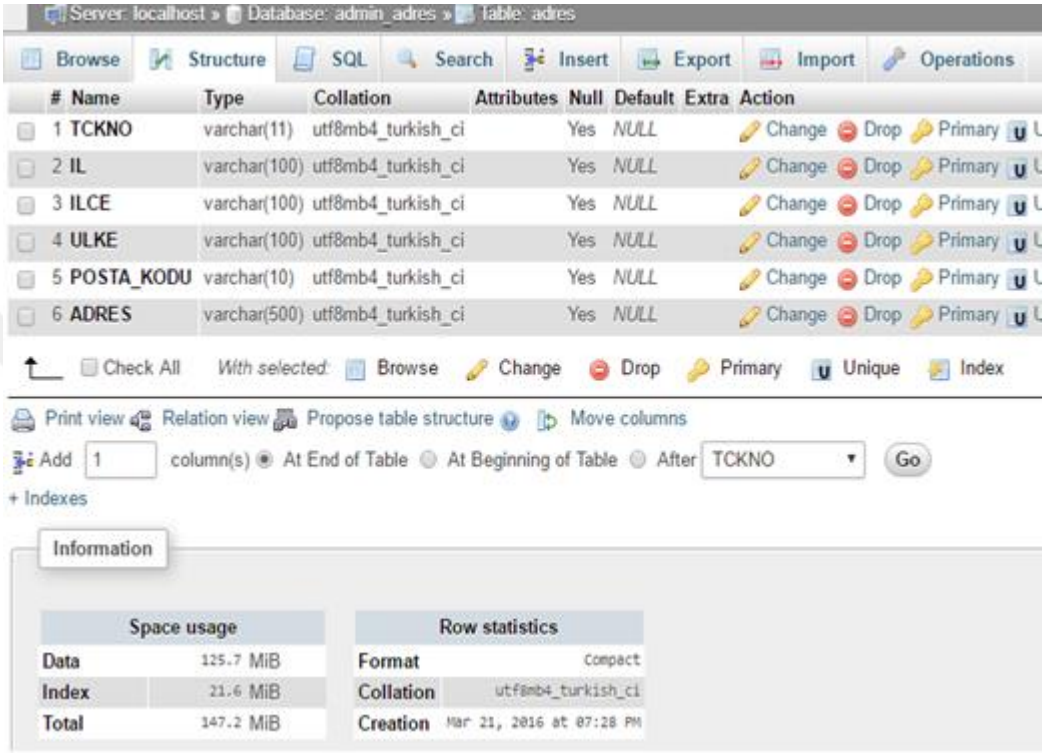
Şekil 3.14'de DataOCEAN kütüphanesi için kimlik servisi ile ilgili oluşturulan JSON formatındaki tanım dosyasının içeriği görülmektedir.

```
{
    "Namespace": "net.egnar.ws.kimlik",
    "Table": "kimlik",
    "Url": "https://ws.egnar.net/Kimlik",
    "ServiceType": "RESTFUL",
    "Description": "Kimlik Bilgileri",
    "RequestMethod": "POST",
    "Parameters": [{
        "Name"         : "TCKNO",
        "VarType"      : "varchar",
        "VarLength"    : "11",
        "VarPrecision" : null,
        "IsRequest"    : "true",
        "IsDynamic"    : "true",
        "Default"      : null,
        "IsResponse"   : "true"
    }, {
        "Name"         : "ADI",
        "VarType"      : "varchar",
        "VarLength"    : "100",
        "VarPrecision" : null,
        "IsRequest"    : "true",
        "IsDynamic"    : "true",
        "Default"      : null,
        "IsResponse"   : "true"
    }, {
        "Name"         : "SOYADI",
        "VarType"      : "varchar",
        "VarLength"    : "100",
        "VarPrecision" : null,
        "IsRequest"    : "true",
        "IsDynamic"    : "true",
        "Default"      : null,
        "IsResponse"   : "true"
    }, {
        "Name"         : "ANNE_ADI",
        "VarType"      : "varchar",
        "VarLength"    : "100",
        "VarPrecision" : null,
        "IsRequest"    : "false",
        "IsDynamic"    : "true",
        "Default"      : null,
        "IsResponse"   : "true"
    }, {
        "Name"         : "BABA_ADI",
        "VarType"      : "varchar",
        "VarLength"    : "100",
        "VarPrecision" : null,
        "IsRequest"    : "false",
        "IsDynamic"    : "true",
        "Default"      : null,
        "IsResponse"   : "true"
    }, {
        "Name"         : "ANNE_KIZLIK_SOYADI",
        "VarType"      : "varchar",
        "VarLength"    : "100",
        "VarPrecision" : null,
        "IsRequest"    : "false",
        "IsDynamic"    : "true",
        "Default"      : null,
        "IsResponse"   : "true"
    }, {
        "Name"         : "DOGUM_YERI",
        "VarType"      : "varchar",
        "VarLength"    : "50",
        "VarPrecision" : null,
        "IsRequest"    : "false",
        "IsDynamic"    : "true",
        "Default"      : null,
        "IsResponse"   : "true"
    }, {
        "Name"         : "DOGUM_TARIHI",
        "VarType"      : "date",
        "VarLength"    : null,
        "VarPrecision" : null,
        "IsRequest"    : "false",
        "IsDynamic"    : "true",
        "Default"      : null,
        "IsResponse"   : "true"
    }, {
        "Name"         : "MEDENI_DURUMU",
        "VarType"      : "varchar",
        "VarLength"    : "30",
        "VarPrecision" : null,
        "IsRequest"    : "false",
        "IsDynamic"    : "true",
        "Default"      : null,
        "IsResponse"   : "true"
    }]
}
```


**Şekil 3.14 :** Kimlik JSON dosyası.

### 3.5.2 Adres

Adres servisi gerçeğe uygun olarak tasarlanmıştır. Adres bilgisi ise internet üzerinden elde edilen rastgele adresler alınarak adres tablosuna eklenerek üretilmiştir. Kimlik verisinde üretilen kimlik numaraları ile eşleştirilmiştir. Adres web servisinin kullandığı adres tablosunun yapısı Şekil 3.15'de verilmiştir.




**Şekil 3.15 :** Adres tablosunun görünümü.

Üretilen bilgide bir TC Kimlik numarasına(TCKNO) birden fazla adres, bir adreste de bağlantılı bir ve/veya birden fazla kişi yerleştirilmesine özen gösterilmiştir.

```
define('DB_QUERY_ADDRESS',
'select * from adres where 1=1 '
        .'{{TCKNO,and TCKNO=:TCKNO}}  limit 100000');
```

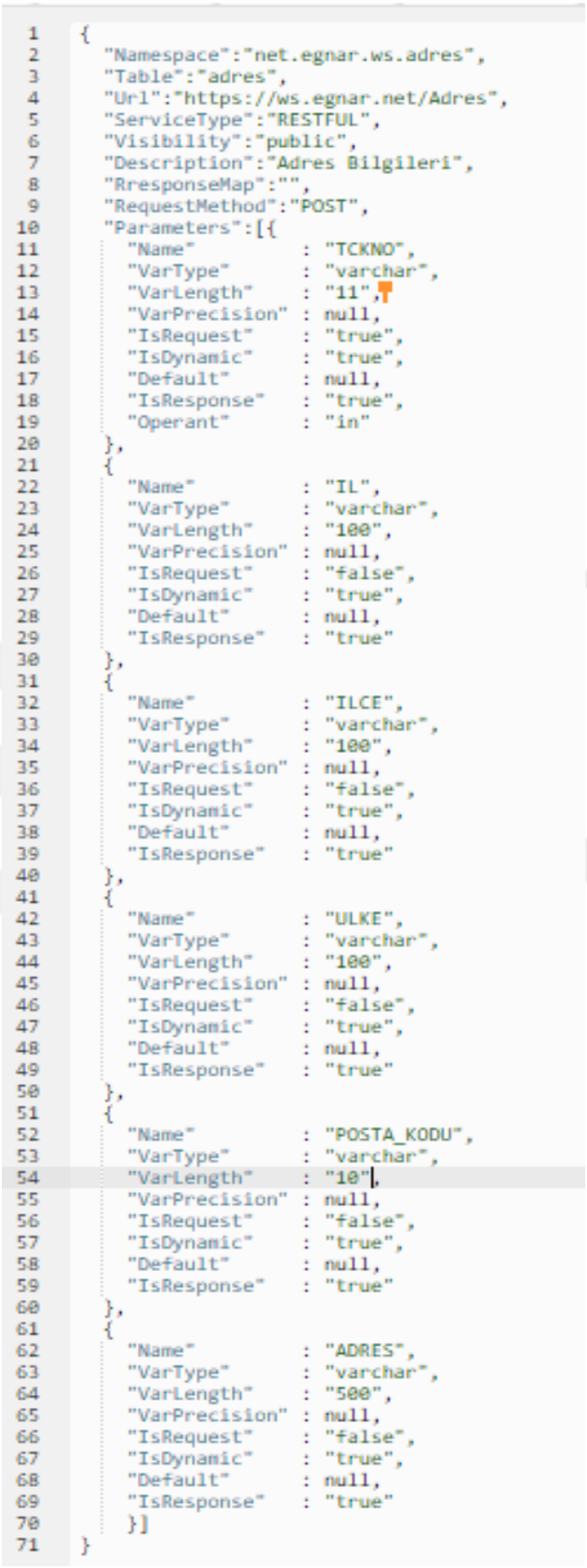
**Şekil 3.16 :** Adres sorgu örneği.

Şekil 3.16'da süslü parantez içindeki TCKNO, dinamik sorguyu destekleyen alan olduğunu işaret etmektedir. Web servisine gönderilen TCKNO alan parametresi servise geldiğinde sorguya eklenir, gelmemesi durumunda ise sorgudan çıkartılır.

```php
public function Adres(){
  header('Content-Type: application/json; charset=utf-8');
  $Db=new Database();
  $Data=array();
  if(isset($_POST['TCKNO'])){
    $Data['TCKNO']=$_POST['TCKNO'];
  }
  $Result=$Db->Select(DB_QUERY_ADDRESS,$Data);
  echo json_encode($Result,JSON_UNESCAPED_UNICODE);
}
```

**Şekil 3.17 :** Adres web servis örneği.

Şekil 3.17'de adres web servisinin kodu verilmiştir. Parametrelerden hangisi istek olarak gelir ise, o parametre data objesine eklenir. Daha sonra, adres sorgusu veritabanında çalıştırılır, sonuç istemciye JSON formatında gönderilir.



```
{
  "Namespace":"net.egnar.ws.adres",
  "Table":"adres",
  "Url":"https://ws.egnar.net/Adres",
  "ServiceType":"RESTFUL",
  "Visibility":"public",
  "Description":"Adres Bilgileri",
  "RresponseMap":"",
  "RequestMethod":"POST",
  "Parameters":[{
    "Name"          : "TCKNO",
    "VarType"       : "varchar",
    "VarLength"     : "11",
    "VarPrecision"  : null,
    "IsRequest"     : "true",
    "IsDynamic"     : "true",
    "Default"       : null,
    "IsResponse"    : "true",
    "Operant"       : "in"
  },
  {
    "Name"          : "IL",
    "VarType"       : "varchar",
    "VarLength"     : "100",
    "VarPrecision"  : null,
    "IsRequest"     : "false",
    "IsDynamic"     : "true",
    "Default"       : null,
    "IsResponse"    : "true"
  },
  {
    "Name"          : "ILCE",
    "VarType"       : "varchar",
    "VarLength"     : "100",
    "VarPrecision"  : null,
    "IsRequest"     : "false",
    "IsDynamic"     : "true",
    "Default"       : null,
    "IsResponse"    : "true"
  },
  {
    "Name"          : "ULKE",
    "VarType"       : "varchar",
    "VarLength"     : "100",
    "VarPrecision"  : null,
    "IsRequest"     : "false",
    "IsDynamic"     : "true",
    "Default"       : null,
    "IsResponse"    : "true"
  },
  {
    "Name"          : "POSTA_KODU",
    "VarType"       : "varchar",
    "VarLength"     : "10",
    "VarPrecision"  : null,
    "IsRequest"     : "false",
    "IsDynamic"     : "true",
    "Default"       : null,
    "IsResponse"    : "true"
  },
  {
    "Name"          : "ADRES",
    "VarType"       : "varchar",
    "VarLength"     : "500",
    "VarPrecision"  : null,
    "IsRequest"     : "false",
    "IsDynamic"     : "true",
    "Default"       : null,
    "IsResponse"    : "true"
    }]
}
```

**Şekil 3.18 :** Adres JSON dosyası.

Şekil 3.18'de OLY kütüphanesi için adres servisi ile ilgili oluşturulan JSON formatındaki tanım dosyasının içeriği sunulmaktadır.

### 3.5.3 Konum

DataOCEAN yapısında demografik servislerin yanısıra konumsal servisler de yer almaktadır. Verimli veri erişimi ve paylaşımı olması için birlikte çalışabilirlik ilkelerinin izlenmesi gerekir. Birlikte çalışabilirlik altyapıları ve standartları tüm alanlarda vazgeçilmez olduğu gibi, konumsal veri ve konumsal servisler alanında da önemli bir gereksinimdir.

Konumsal anlamda birlikte çalışabilirlik altyapıları "Konumsal Veri Altyapıları" olarak adlandırılmaktadır. Son yıllarda, yerel yönetimler, kamu, özel kuruluşlar, konumsal veriyi yaşamımızın her alanıyla ilişkilendirmeye çalışmaktadırlar. Her tür veriye anlık erişim olanağı ancak doğru tanımlanan ortak altyapı ve karşıt ilişkiler bazında oluşturulan yapılar aracılığı ile mümkün görünmektedir [28]

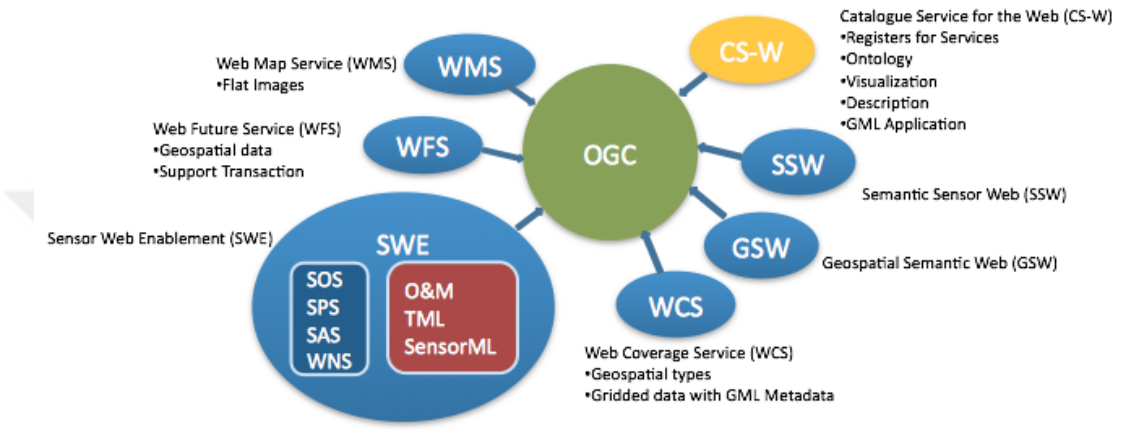
#### 3.5.3.1. Mekansal Veri Standartları

Konumsal veride uluslar arası standart oluşumu ihtiyacı ve coğrafi standartlar üzerinde uzlaşma sağlamak amacıyla **OGC** (The Open Geospatial Consortium) oluşumu, 1994 yılında kurulmuş gönüllü bir uluslararası organizasyondur. Standart bir yapı ile hazırlanan servisler portal, katalog, gösterim ve veri servisleri olmak üzere dört temel grubta incelenirler.

- Portal servisleri, diğer servislere erişimde kullanılan istemci – sunucu yazılımlarını ve aralarındaki etkileşimi yönetmektedir.
- Katalog servisleri, konumsal verilerin *metadata* (metaveri) bilgisini içeren yapılardır.
- Gösterim servisleri, konumsal verilerin görselleştirilmesinde kullanılan bilgiler ile uygun formatlardaki görsellerin kendi paylaşımını kolaylaştırırlar.
- Veri servisleri, konumsal veri ve veri yapılarına erişim ve paylaşım olanağı sağlayan servislerdir.

Veri aktarım ve paylaşımında en çok kullanılan servis SFA (Simple Feature Access - Temel Nesne Tanımlama) servisidir.

Konumsal veri işleme ve veri paylaşma konularında standartlaşma sağlamak için normlar oluşturulmuştur. Bu normlar, ağırlıklı olarak web üzerinden bilgi paylaşan coğrafi web servislerinin belirli standartlarda bilgi üretmesine, bu servislerin kullanımının kolaylaştırılmasına ve yaygınlaştırılmasını desteklemektedir. OGC tarafından tanımlanmış 30'dan fazla standart bulunmaktadır (WMS, WFS, CS-W, WMTS vb.) [27] [65] Bunlardan bazıları Şekil 3.19'da izlenebilmektedir.




**Şekil 3.19 :** OGC Standartları,2012 [27]

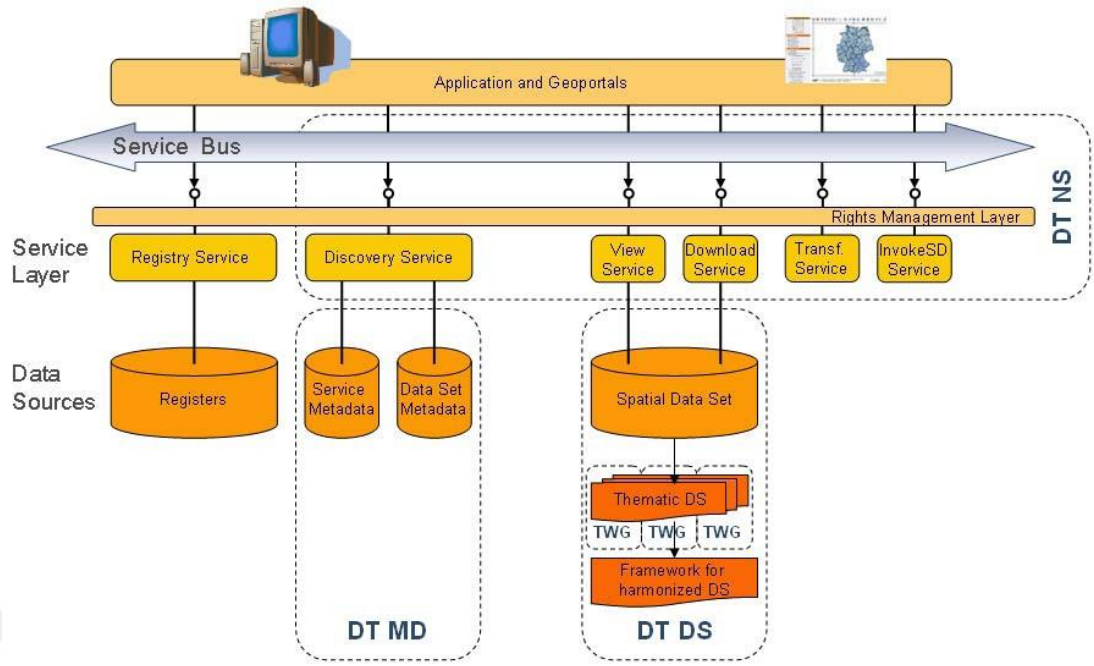
Çalışma süresince yararlanılan bir diğer standart AB üyesi ülkelerin gelişmişlik düzeyine parallel olarak geliştirilen konumsal altyapı bileşenleri arasında farklılıkları ortadan kaldırmayı hedefleyen **INSPIRE** (Infrastructure for Spatial Information in the European Community) standartlar yapısıdır. Avrupa Komisyonu tarafından mekansal bilgi için bir altyapının teknik olarak koordinasyonunu sağlamak amacıyla Eylem 2142–ESDI (Avrupa Konumsal Veri Altyapısı) başlatılmıştır. ESDI, bölgesel veya farklı, birçok kaynaktan gelen coğrafi verileri birleştirerek, kullanıcılara hazır veri setleri sunmayı hedeflemiştir. Küresel veri paylaşımı hedefiyle oluşan bu kuruluş, bölgesel, ulusal ve yerel ölçekteki coğrafi bilginin etkin kullanımı ve paylaşımı için, kullanılacak teknoloji ve standartlarla ilgili politikaları belirlemeyi gerçekleştirmektedir [26]. INSPIRE girişimi, Avrupa mekansal veri altyapısının oluşturulması ve mekansal bilginin paylaşımı için belli ilkeler belirler. Bunlar;

- Veri bir kere toplanmalı ve en yararlı olabileceği seviyede tutulmalıdır.

- Avrupa'dan farklı kaynaklardan elde edilen ve farklı görünen mekansal bilginin birleştirilmesi sağlanmalı ve bu bilgi birçok kullanıcı ve uygulama ile paylaşılmalıdır.
- Tek bir seviyede toplanmış verinin tüm diğer seviyeler arasında da paylaşılması sağlanmalıdır.
- Tüm seviyelerde iyi yönetim için ihtiyaç duyulan coğrafi bilgi fazla olmalıdır ve gerçek kullanım amaçlarını kısıtlamayacak şartlarda geniş ölçüde kullanılabilir olmalıdır.
- Hangi coğrafi bilginin kullanılabilir olduğu, belirli bir kullanım için ihtiyaçları karşıladığı ve hangi şartlarda temin edilip kullanılabileceği gibi bilgilere kolay erişilebilmelidir.
- Coğrafi veri farklı durumlara göre görüntüleneceği ve kullanıcı-dostu bir yolla seçileceği için kolay anlaşılır ve kullanılır şekilde hazırlanmalıdır.

INSPIRE direktifleri çerçevesinde 5 Kasım 2007'de INSPIRE Teknik Mimarisi yayınlanmıştır. Şekil 3.20'de bu mimarinin içerdiği bileşenler, aralarındaki ilişkiler ve standartları oluşturmakta yetkili grupların bilgisi yer almaktadır. Bu mimaride görüldüğü üzere en dikkat çekici servisler katalog ve ağ servisleridir. Ağ (network) servisleri konumsal verilerinin bulunmasını (Discovery Service), dönüştürülmesini (Transformation Service), görüntülenmesini (View Service), indirilmesini (Download Service), ayrıca konumsal veri servisi (InvokeSD Service) ile e-Ticaret servislerinin çağırılmasını olanak tanımaktadır [27].

AB konumsal veri altyapısı uygulamaları INSPIRE direktiflerini baz alarak gelişmeye devam ederken üye ülkelerden de mevcut konumsal veri setleri ve veri servisleri için metaveri toplamaları ve sürekli güncel halde tutmaları beklenmektedir. Ülkemizde de INSPIRE'ı temel alan, TUCBS (Türkiye Ulusal Coğrafi Bilgi Sistemi) ulusal konumsal veri altyapısı kurma faaliyetleri başlatılmış ve devam etmektedir [65].




**Şekil 3.20 :** INSPIRE Teknik Mimari Görünümü (INSPIRE, 2007)

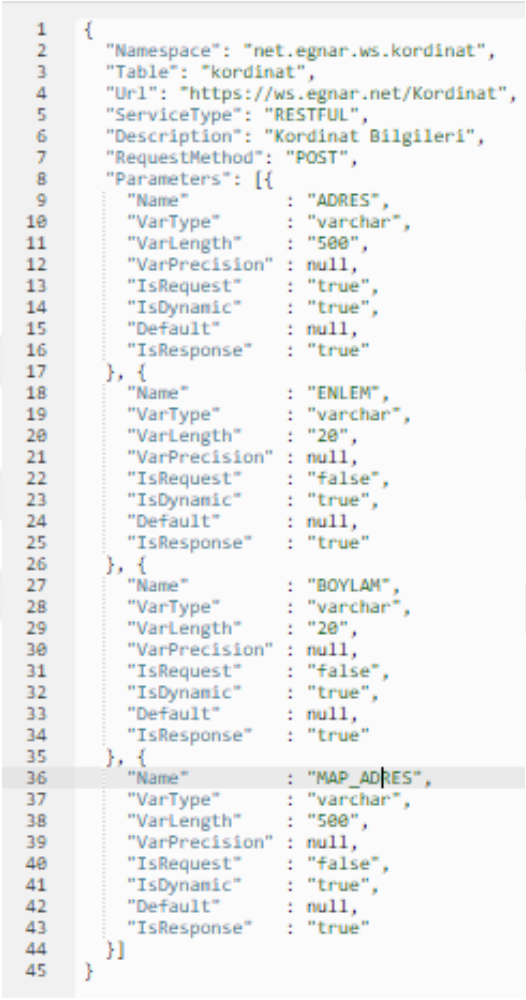
3.5.3.2. Konum Servisi.

Adres Konum servisi **Google MAP'**in [11] kullandığı **GeoCode** [66]servisini çağırmaktadır. Web servisi açık adresin bulunduğu enlem/boylam koordinatlarını döndürmektedir. Bu servis, ticari amaç gütmeyen yazılım geliştiriciler için özel olarak sunulmuş ve günlük 2500 adet sorguya izin verilen bir servistir. Limit üzerinde sorgu yapılabilmesi için Google'dan ücretli olarak lisans satın alınması gerekir. Sistemin konumsal veri ekleme ve görselleştirme yeteneği de bu çalışma için önem arz etmektedir. Şekil 3.19'da servise gelen açık adres bligisinin GoogleMAP GeoCode API'ını kullanarak adresin enlem, boylam ve harita (map) adresini almaktadır. Bu konum bilgisi Google Map üzerinde noktasal olarak gösterilmektedir.

```
public function Kordinat(){
  $Data=array();
  if(isset($_POST['ADRES'])){
    $Request=$_POST['ADRES'];
    $Adres=urlencode($Request);
    $Url="https://maps.google.com/maps/api/geocode/json?address={$Adres}&key=AIzaSyB2ECN1xipCDe6dpo80OGcj28MijgvlcJE";
    $Result=json_decode(file_get_contents($Url),true);
    if($Result['status']=='OK'){
      $Data[0]=array('adres'=>$Request,
                     'enlem'=>$Result['results'][0]['geometry']['location']['lat'],
                     'boylam'=>$Result['results'][0]['geometry']['location']['lng'],
                     'map_adres'=>$Result['results'][0]['formatted_address']);
    }
  }
  echo json_encode($Data,JSON_UNESCAPED_UNICODE);
 }
}
```

**Şekil 3.21 :** DataOCEAN kütüphanesi için konum servisi.

Şekil 3.20'de DataOCEAN kütüphanesi için konum servisi ile ilgli oluşturulan JSON formatındaki tanım dosyasının içeriğidir. DataOCEAN yapısının kullanımını desteklediği her servis uygulamalara kolayca dahil edilebilir. Sistemde bulunan servis güncelleme ve sonrasındaki senkronizasyon işlemi anlatıldığı gibi hızlı ve kolay arka planda yapılmaktadır.



```
{
  "Namespace": "net.egnar.ws.kordinat",
  "Table": "kordinat",
  "Url": "https://ws.egnar.net/Kordinat",
  "ServiceType": "RESTFUL",
  "Description": "Kordinat Bilgileri",
  "RequestMethod": "POST",
  "Parameters": [{
    "Name"         : "ADRES",
    "VarType"      : "varchar",
    "VarLength"    : "500",
    "VarPrecision" : null,
    "IsRequest"    : "true",
    "IsDynamic"    : "true",
    "Default"      : null,
    "IsResponse"   : "true"
  }, {
    "Name"         : "ENLEM",
    "VarType"      : "varchar",
    "VarLength"    : "20",
    "VarPrecision" : null,
    "IsRequest"    : "false",
    "IsDynamic"    : "true",
    "Default"      : null,
    "IsResponse"   : "true"
  }, {
    "Name"         : "BOYLAM",
    "VarType"      : "varchar",
    "VarLength"    : "20",
    "VarPrecision" : null,
    "IsRequest"    : "false",
    "IsDynamic"    : "true",
    "Default"      : null,
    "IsResponse"   : "true"
  }, {
    "Name"         : "MAP_ADRES",
    "VarType"      : "varchar",
    "VarLength"    : "500",
    "VarPrecision" : null,
    "IsRequest"    : "false",
    "IsDynamic"    : "true",
    "Default"      : null,
    "IsResponse"   : "true"
  }]
}
```

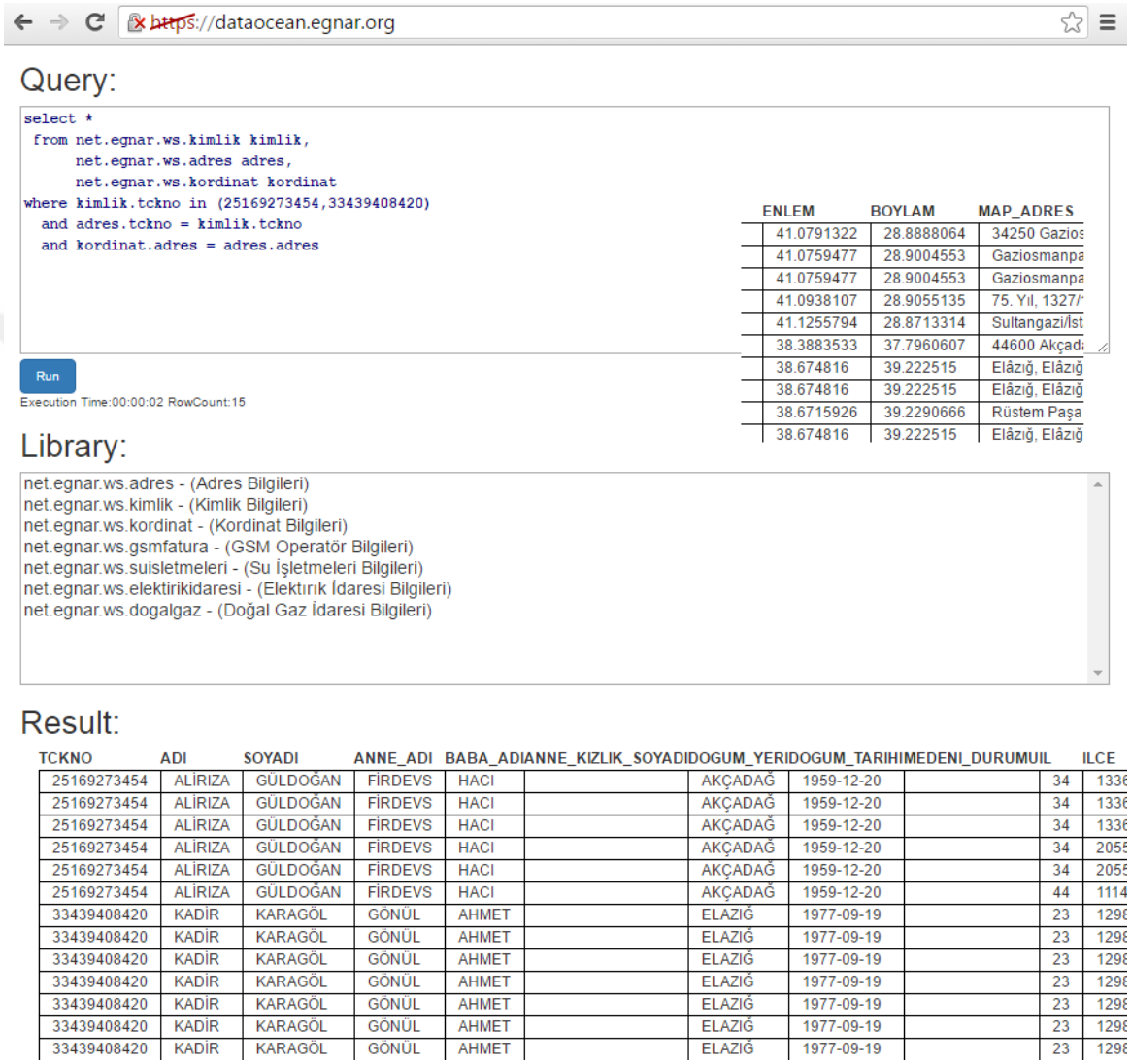
**Şekil 3.22 :** DataOCEAN konum JSON dosyası formatı.

### 3.6 DataOCEAN : İşleyiş

DataOCEAN sisteminde SQL yazıp çalıştırmak üzere tasarlanmış görsel arabirim yazılmıştır. Hazırlanan görsel ekranın sağ üst tarafında yer alan "Library" kısmında, DataOCEAN yapısında tanımlanmış web servisleri isimleri ile yer almaktadır.

QUERY, SQL cümleciğin yazılan kısımdır. QUERY alanının yanında bulunan RUN butonu çalıştırıldığında, sonuç bilgisi hemen alt kısmında bulunan, ve gösterimi grid

şeklinde olan alanda yer alır. Birleştirilmiş görünüm Şekil 5'de incelenebilir. RUN butonunun hemen altında yer alan ve istatistiki bilgi birikimi için kullandığımız çalışma süresi ve dönen kayıt sayısı bilgisi yer almaktadır. Google MAP aracılığı ile harita üzerinde noktasal gösterim ekranı "View in Google MAP" butonu ile izlenebilmektedir.



```
select *
 from net.egnar.ws.kimlik kimlik,
      net.egnar.ws.adres adres,
      net.egnar.ws.kordinat kordinat
where kimlik.tckno in (25169273454,33439408420)
  and adres.tckno = kimlik.tckno
  and kordinat.adres = adres.adres
```

Execution Time:00:00:02 RowCount:15

| ENLEM | BOYLAM | MAP_ADRES |
|---|---|---|
| 41.0791322 | 28.8888064 | 34250 Gazios |
| 41.0759477 | 28.9004553 | Gaziosmanpa |
| 41.0759477 | 28.9004553 | Gaziosmanpa |
| 41.0938107 | 28.9055135 | 75. Yıl, 1327/ |
| 41.1255794 | 28.8713314 | Sultangazi/İst |
| 38.3883533 | 37.7960607 | 44600 Akçad |
| 38.674816 | 39.222515 | Elâzığ, Elâzığ |
| 38.674816 | 39.222515 | Elâzığ, Elâzığ |
| 38.6715926 | 39.2290666 | Rüstem Paşa |
| 38.674816 | 39.222515 | Elâzığ, Elâzığ |

net.egnar.ws.adres - (Adres Bilgileri)
net.egnar.ws.kimlik - (Kimlik Bilgileri)
net.egnar.ws.kordinat - (Kordinat Bilgileri)
net.egnar.ws.gsmfatura - (GSM Operatör Bilgileri)
net.egnar.ws.suisletmeleri - (Su İşletmeleri Bilgileri)
net.egnar.ws.elektirikidaresi - (Elektırık İdaresi Bilgileri)
net.egnar.ws.dogalgaz - (Doğal Gaz İdaresi Bilgileri)

| TCKNO | ADI | SOYADI | ANNE_ADI | BABA_ADI | ANNE_KIZLIK_SOYADI | DOGUM_YERI | DOGUM_TARIHI | MEDENI_DURUMU | IL | ILCE |
|---|---|---|---|---|---|---|---|---|---|---|
| 25169273454 | ALİRIZA | GÜLDOĞAN | FİRDEVS | HACI |  | AKÇADAĞ | 1959-12-20 |  | 34 | 1336 |
| 25169273454 | ALİRIZA | GÜLDOĞAN | FİRDEVS | HACI |  | AKÇADAĞ | 1959-12-20 |  | 34 | 1336 |
| 25169273454 | ALİRIZA | GÜLDOĞAN | FİRDEVS | HACI |  | AKÇADAĞ | 1959-12-20 |  | 34 | 1336 |
| 25169273454 | ALİRIZA | GÜLDOĞAN | FİRDEVS | HACI |  | AKÇADAĞ | 1959-12-20 |  | 34 | 2055 |
| 25169273454 | ALİRIZA | GÜLDOĞAN | FİRDEVS | HACI |  | AKÇADAĞ | 1959-12-20 |  | 34 | 2055 |
| 25169273454 | ALİRIZA | GÜLDOĞAN | FİRDEVS | HACI |  | AKÇADAĞ | 1959-12-20 |  | 44 | 1114 |
| 33439408420 | KADİR | KARAGÖL | GÖNÜL | AHMET |  | ELAZIĞ | 1977-09-19 |  | 23 | 1298 |
| 33439408420 | KADİR | KARAGÖL | GÖNÜL | AHMET |  | ELAZIĞ | 1977-09-19 |  | 23 | 1298 |
| 33439408420 | KADİR | KARAGÖL | GÖNÜL | AHMET |  | ELAZIĞ | 1977-09-19 |  | 23 | 1298 |
| 33439408420 | KADİR | KARAGÖL | GÖNÜL | AHMET |  | ELAZIĞ | 1977-09-19 |  | 23 | 1298 |
| 33439408420 | KADİR | KARAGÖL | GÖNÜL | AHMET |  | ELAZIĞ | 1977-09-19 |  | 23 | 1298 |
| 33439408420 | KADİR | KARAGÖL | GÖNÜL | AHMET |  | ELAZIĞ | 1977-09-19 |  | 23 | 1298 |
| 33439408420 | KADİR | KARAGÖL | GÖNÜL | AHMET |  | ELAZIĞ | 1977-09-19 |  | 23 | 1298 |

**Şekil 3.23 :** DataOCEAN SQL ile servis sorgulama ekranı (ayrıntılı).

Şekil 3.22'de görüldüğü gibi RUN butonunun hemen altında istatistik için kullanılabilecek çalışma süresi ve dönen kayıt sayısı yer almaktadır. Google MAP aracılığı ile harita üzerinde noktasal gösterim ekranı "View in Google MAP" butonu ile izlenebilir.

# 4. UYGULAMA : DataOCEAN SİSTEM GERÇEKLEME ÖRNEĞİ

DataOCEAN yapısı platform bağımsız, her sisteme kolayca uyarlanabilen, entegrasyon işlemini yazılım katmanında sağlayan bir sistem modelidir. Veri ve veri kaynaklarının türü ve yapısı özgün olmakla birlikte, yapıya dahil olan servislere merkezi veri erişimi sunar. Kullanıcı katmanında yazılım bilgisi gerektirmediğinden geniş kullanıcı kitlesi tarafından kullanılabilir.

Bu doğrultuda DataOCEAN örnek uygulaması kurgulanmış ve yazılmıştır. Bu uygulamada Web servislerinde kullanılan verilerin tamamı gerçeğe uygun üretilmiştir. Örnek uygulama PHP dilinde kodlanmıştır. DataOCEAN server işletim sistemi olarak Linux, veritabanı olarak MySQL, mekansal veri gösterimi için Google Earth kullanılmıştır.

## 4.1 Kurgu

İstanbul ilçelerinde planlanan kentsel dönüşüm çalışmaları ve bu kapsamda yeni yapılara olabilecek talep ve potansiyel alıcı kitlesi belirlemek amacıyla çeşitli kurumlardan bilgiler entegre edilerek bir tahmin sonuç üretilmesi amaçlanmaktadır. Son 5 yılda, Beşiktaş ilçesinde yapılan yeni yapılar, değişime uğrayan bölgeler, muhtemel konut fiyatları, kayıp ve artışlarıyla birlikte, mülk satışı değerleri, ortalama ilçe sakinleri gelir / gider bilgileri ve potansiyel alıcı kitlesi belirleme çalışması planlanmıştır. Bu kurguda gerekli olan bilgiler: Yeni yapılar, bulundukları alanlar, bu alanlara yakın ikamet eden kişiler, ilgili kişilerin bankadaki bilgileri, önceden belirlenmiş hesap bakiyesi tutarı olanların kredi notu, tanıtımın yoğunlaşacağı hedef kitlenin belirlenmesi vb. bilgilerdir.

## 4.2 Örnek Sorgulama Algoritması

4.2.1 İlgili bölgelerde yaşayanların tespiti.

- Harita üzerinden çalışılacak bölge seçilir.

- Seçilen alanda ikamet edilen kişilerin bilgisi alınır. Adres servisi aracılığı ile adres veri tabanından alınan anlık veri bazlı, uygulama aracılığı ile üretilen tabloya aktarılır.

- Aktarılan tablodan (Adres) TC Kimlik No elde edilir.

4.2.2 Kişilerin banka bilgisine erişilmesi.

- Elde edilen TC Kimlik No lara istinaden WS_Banka servisi ile, önceden belirlenmiş kriterler doğrultusunda (where couse) sorgu oluşturulur, servise aktarılır, bilgilere ulaşılır, uygulama aracılığı ile Banka tablosuna/JSON dosyasına aktarılır.

4.2.3 Kredi notu tespiti.

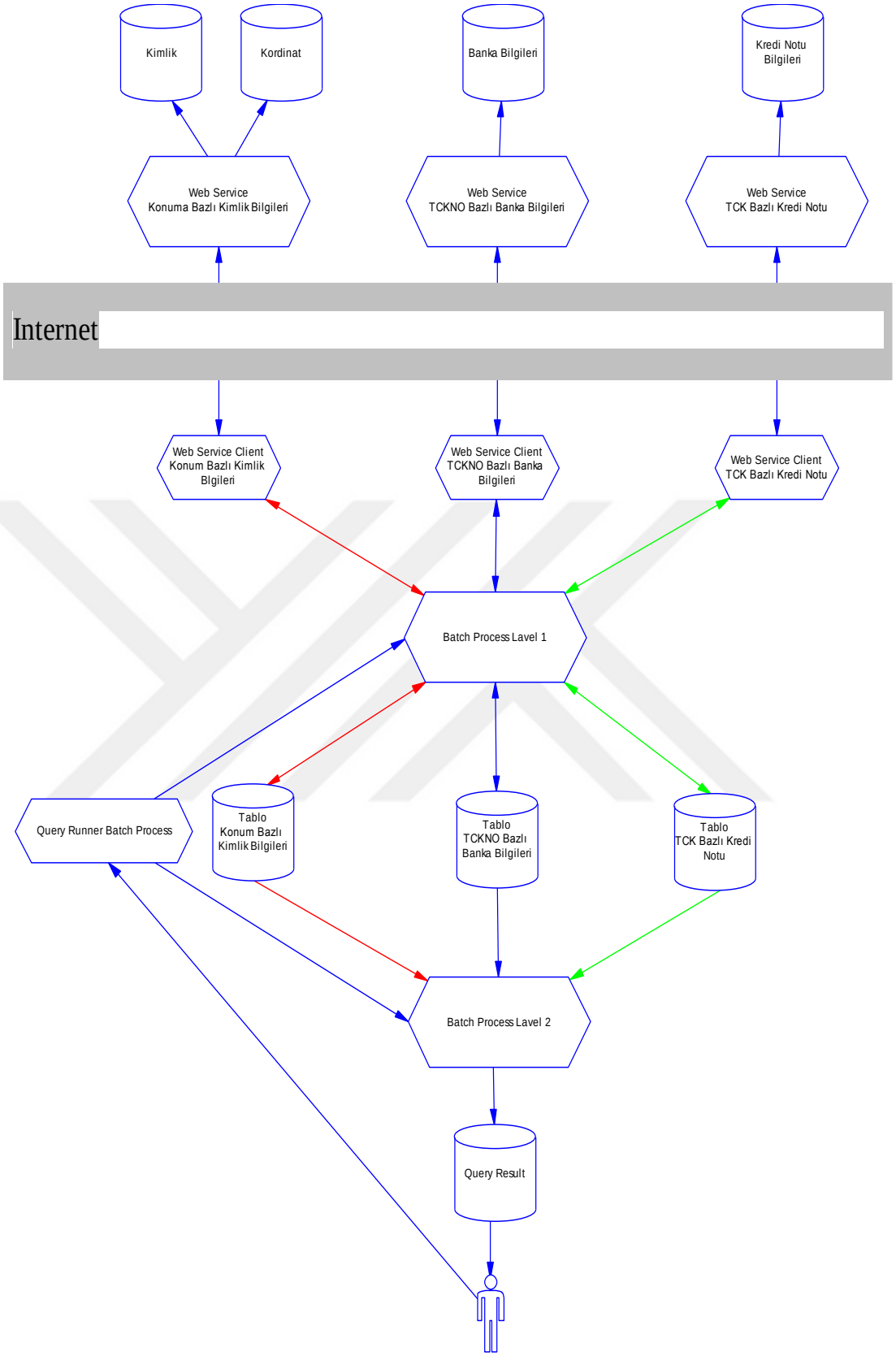
WS_Banka servisindeki bilgiye istinaden oluşturulan Banka tablosundaki kişilerin WS_KrediNotu servisiyle sorgulanır, veriler uygulama aracılığı ile tabloya/JSON dosyasına aktarılır.

4.2.4 Operasyon

Bu kurguda toplam 4 web servisi kullanılmış, 3 ayrı tablo/JSON dosyası oluşturulmuştur. Mekansal sorgulama sonrası elde edilen veri, kullanılan referans harita kullanılarak görselleştirilmiştir. Şekil 4.1'de tasarlanan modelin şematik gösterimi yer almaktadır. İzlendiği gibi, web servisinin *request* objesi ile hangi elemanları sorgulanacak ise, yeni bir *where couse* yaratılmaktadır.

Sistemde birden fazla veriyi sorgularken;

- Sisteme gelen istek sonrası merkezi bilgi kontrol katmanında bulunan algoritma ile ulaşılacak web servisleri ve erişim sırası belirlenmektedir.
- Tasarlanan bilgi sistemi modelinde web servisleri ile entegre olan veri tabanlarının yapısı ve türü sistemin genel yapısını etkilemediği için göz ardı edilmiştir.
- Web servisleri "birlikte çalışabilirlik" esasına uyularak standartlar çerçevesinde tasarlanmışlardır.
- Konumsal servislerde OGC yapıları esas alınmıştır.
- Prototip sistemde bulunan veriler gerçeğine uygun olarak üretilmiştir.
- Çalışmadaki sonuçlar gerçeğe yakın ortam yaratılarak incelenmiş ve yorumlanmıştır.




**Şekil 4.1 :** DataOCEAN Modeli ile hazırlanan algoritmanın şematik gösterimi.

- Konumsal veri bilgisi gelen servislerde bulunmakla birlikte, boyut kaygısı taşınarak (örn. uydu görüntüleri) sistemde bulunan merkezi yapıda da yer almaktadırlar.

Yapılan çalışmada kullanıcı talebi sonrasında oluşan ilgili girdi parametreleri DataOCEAN kontrol katmanı aracılığı ile önceden belirlenmiş algoritma kullanılarak ilgili servislere yönlendirilmiş ve çalışma başlatılmıştır. Kullanılan web servisi sayısı, servislerin aktarıldığı tablo yapı ve sayıları talep edilen sonuç doğrultusunda tamamen dinamik olarak oluşturulmuştur.

Şekil 4.2 de kurgulanan yapının sonuç gösterimi yer almaktadır. QUERY alanında sisteme gönderilen sorgu, Library alanında ise sistemde yer alan servisler incelenebilir. Sorgulamada yönlendirilen sonuç bilgisi Google Map altlıklı haritada ve grid şeklinde yer almaktadır. Kullanılan toplam 4 servisten sorgulanan veri süresi 00:01:53 olarak görülmektedir.

**Query:**

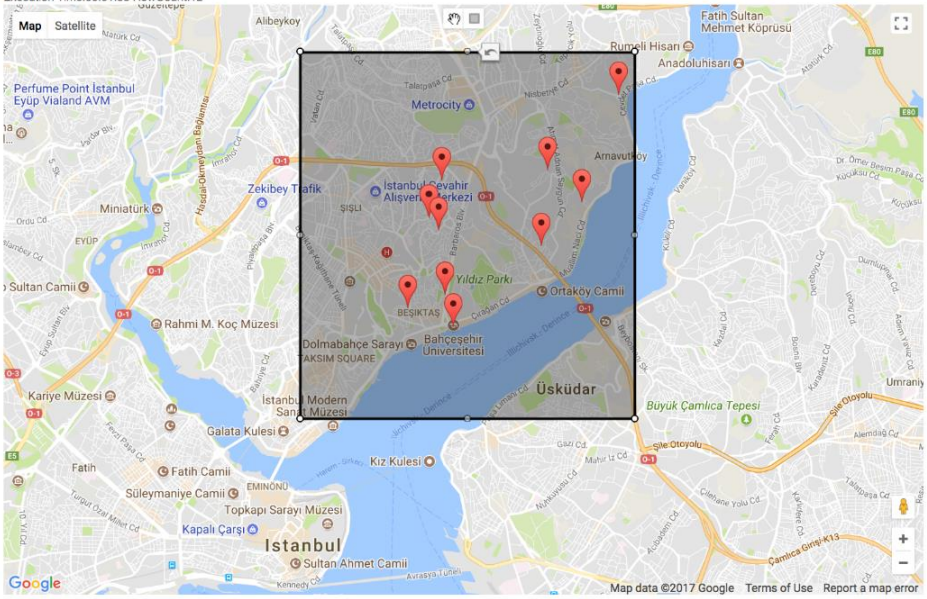
```
select *
  from net.egnar.ws.krediderece krediderece,
       net.egnar.ws.kimlik kimlik,
       net.egnar.ws.adres adres,
       net.egnar.ws.kordinat kordinat
 where krediderece.puan > 1000
   and kimlik.tckno=krediderece.tckno
   and adres.tckno=kimlik.tckno
   and adres.adres like '%BEŞİKTAŞ%'
   and kordinat.adres = adres.adres
```

**Library:**

net.egnar.ws.adres - (Adres Bilgileri)
net.egnar.ws.kimlik - (Kimlik Bilgileri)
net.egnar.ws.kordinat - (Kordinat Bilgileri)
net.egnar.ws.krediderece - (Kredi Derece Bilgileri)
net.egnar.ws.gsmfatura - (GSM Operatör Bilgileri)
net.egnar.ws.suisletmeleri - (Su İşletmeleri Bilgileri)
net.egnar.ws.elektirikidaresi - (Elektrik İdaresi Bilgileri)
net.egnar.ws.dogalgaz - (Doğal Gaz İdaresi Bilgileri)

Run

Execution Time:00:01:53 RowCount:12




**NE Lat:** 41.0838708393766
**NE Lng:** 29.0457916259765
**SW Lat:** 41.0276918100181
**SW Lng:** 28.9778137207031

**Result:**

| TCKNO | PUAN | ADI | SOYADI | ANNE_ADI | BABA_ADI | ANNE_KIZLIK_SOYADI | DOGUM_YERI | DOGUM_TARIHI | MEDENI_DURUMU | IL | ILCE | ULKE | POSTA_KODU | A |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 12160697204 | 1200 | DİLEK | KARAMUK | MELAHAT | ALİ DURSUN | | ALAÇAM | 1978-12-06 | | 34 | 1183 | TR | 34349 | |
| 22405598678 | 1018 | İBRAHİM | ÖZDOĞAN | HİKMET | MUSTAFA | | TAVŞANLI | 1963-01-01 | | 34 | 1183 | TR | 34100 | |
| 24830500844 | 1051 | SEYFETTİN | DEMİRCİ | ZEHRA | EYYÜP | | TERME | 1965-06-15 | | 34 | 1183 | TR | 34349 | |
| 24830500844 | 1051 | SEYFETTİN | DEMİRCİ | ZEHRA | EYYÜP | | TERME | 1965-06-15 | | 34 | 1183 | TR | 34354 | |
| 24842151782 | 1012 | ERDAL | YILDIZ | AYŞE | CEMALETTİN | | ÇATALZEYTİN | 1957-03-07 | | 34 | 1183 | TR | 34050 | |
| 27937950956 | 1871 | NEJAT | ALTAN | HAMİ PEYKER | M.CELALETTİN | | İSTANBUL | 1950-05-31 | | 34 | 1183 | TR | 80280 | |
| 29020271092 | 1215 | FURKAN | DEMİRBAŞ | GÜLER | HÜSEYİN | | ÜSKÜDAR | 1989-07-14 | | 34 | 1183 | TR | 34353 | |
| 30524355478 | 1679 | BÜLENT | KOÇAK | NURİYE | HAMDİ | | İSTANBUL | 1965-08-04 | | 34 | 1183 | TR | 34330 | |
| 30848260922 | 1800 | ALİ CİHAN | GİRİT | RAHMİYE | İLYAS | | RİZE | 1981-04-22 | | 34 | 1183 | TR | 34342 | |
| 44821751192 | 1799 | MUSTAFA | AYKUT | AYŞE | FESİH | | MUŞ | 1984-04-18 | | 34 | 1183 | TR | 34357 | |
| 49798154816 | 1312 | SELAHATTİN | USLU | ŞERİFE | HASAN | | BÜYÜKYILDIZ | 1955-10-10 | | 34 | 1183 | TR | 34340 | |
| 54166092294 | 1001 | ZEYNEL HALDUN | TÜZEL | FERİHA | FETHİ | | İSTANBUL | 1953-09-28 | | 34 | 1183 | TR | 34439 | |

**Şekil 4.2 :** DataOCEAN Sonuç Ekran Görüntüsü.

# 5. SONUÇ VE KATKILAR

DataOCEAN sistem modeli SOA yapılarının entegrasyon süreçlerinde daha verimli kullanılması amacı ile tasarlanmış, prototip yazılım ve gerçeğe yakın kurgulanan uygulama çalışma ortamı ile birlikte sunulmuştur. Modelin tasarım aşamasında belirlenen temel amaç genelde sistemler arası entegrasyon işleminde ve özelde *yazılım hazırlama sürecini kısaltmaktır.*

Yazılım Ölçütleri, tasarım ve üretim aşamasında; tasarımın ve kodun anlaşılabilirliği, kodlama kolaylığı, hızı, maliyet, üretim zamanı, bakım, test ve kullanım kolaylığı süreci açısından değerlendirmelerini içermektedir. Bu ölçütleri statik ve dinamik metrikler ile tanımlanmaktadır. Statik metrikler: tasarım sonrası, programlama döneminde, henüz yazılım çalıştırılmadan, sadece kaynak kodu incelemek için kullanılırlar. Dinamik metrikler ise: yazılım çalışma sırasında toplanan verilerden elde edilirler.

Tasarlanan DataOCEAN modeli ve yapılan yazılımda yer alan nesneler, sınıflar, aralarındaki ilişkiler, ilişkilerin yapısal ve davranışsal özellikleri nesne yönelimli tasarım metrikleri kullanılarak değerlendirilmiştir. Statik metrikler, Jean-Françoia Lepine tarafından programlanan ve günümüzde yaygın kullanılan birçok metrik ölçümleri içeren "Php Metrics – Statistic analyzer for PHP" [67] uygulamasıyla sonuç rapor elde edilmiştir. Uygulama kodu belirlenen yol izlenerek incelemeye sunulmuştur. Rapora https://dataocean.egnar.org/myreport.html linkinden erişilebilmektedir. Metrikler konusunda genel bilgiler ve raporların tamamına çalışmanın Ekler bölümünde (EK A.1) ayrıntılı olarak değinilmiştir.

Php Metrics'den üretilen raporda uygulamanın olabildiğince az satır sayısı ile gerçekleştirilmiş olduğu görülmektedir. Çalışmada yapılan mantıksal değişiklik sonrası kodlamada gelinen son iki versiyon için sonuç rapor elde edilmiştir. Versiyon 2.02'de, Ver.2.01'de tekrarlayan iki sınıf çıkarıldıktan sonra, satır sayısında da azalma olmuştur. Kodda tespit edilen hata sayısı 2 dir ve bu sayı işleyiş için kritik değildir.

Yazılım metriklerine göre; sınıf sayısının çokluğu yazılımın tekrar kullanılabilirliğini azalttığından, mümkün olan en az sınıf sayısı ile tanımlanması tercih edilmiştir. Sınıfın ortalama döngüsel karmaşıklığı (Average cyclomatic complexity by class) ana sayfada Maintainability / complexity alanından izlenebilir (Şekil A.1). Yazılan sınıfların tespiti program tarafından karmaşık olarak nitelenmiştir. Bu da yeniden kullanılabilirliğinin, sürdürülebilirliğinin karmaşık olduğunu göstermektedir (Şekil A.1.). Ancak, tasarım birden fazla mimariyi esas aldığından, kompleks bir yapıya sahiptir ve yazılım tarafından karmaşık kabul edilmesi beklenilen bir sonuçtur.

Sınıflar arasındaki etkileşim kabul edilebilir, hatta, iyi sayılabilecek normlar içerisindedir (Şekil A.6). Kod karmaşıklığı bazı noktalarda tam anlamdırılmadığından erişim (violation) konusunda da inceleme gerektirdiğini bildirmektedir (Şekil A.2). Kod satır sayısı ile yapılan işlem orantısı karşılaştırıldığında; yazılım açısından başarılı olarak değerlendirilmiştir (Şekil A.1).

Program incelemesinde, nesne yönelimli yapıya uygunluk açısından bakıldığında, anlaşılabilirlik değeri en düşük olan sorgu cevabını döndüren ve veritabanı katmanı (output ve database) olduğu görünmektedir.

Yazılımın kalite düzeyini yükselten test edilebilirlik, dayanıklılık ve etkinlik değerleridir. Yeniden kullanılabilirlik özelliği, karmaşık ve zor anlaşılabilir yapısından dolayı orta seviyededir.

Yazılımın çalışma esnasında, dinamik metriklerin elde edilebilmesi için metrik ölçüm kümeleri kullanılmaktadır. DataOCEAN metrik değerlendirme aşamasında C&K (Chidamber & Kemerer) kümeleri seçilmişir. Sınıfın Ağırlıklı Metod Sayısı (WMC) bir sınıftaki metodların karmaşıklık derecesi veya sayısı olarak belirlenir. DataOCEAN'da metotların sayı azlığı ile birlikte, karmaşıklığı da sınıfın geliştrilmesine ve bakımına harcanacak zaman-çaba oranını kabul edilebilir sınırlar içerisinde belirlerlemektedir.

Kalıtım ağacının derinliği (DIT), bir sınıfın kalıtım ağacının köküne olan uzaklığını gösteren metriktir. DataOCEAN'da tüm sınıflar köke yakınlığı ile incelendiğinden yazılımın verimliliğini, yeniden kullanımını, anlaşılabilirliğini ve test edilebilirliği olabildiğince iyi değerlere sahiptir (Şekil A.4).

DataOCEAN'da alt sınıf sayısı (NOC) değeri az olduğundan, yeniden kullanım yüksek, hata riski azdır. Nesne sınıfları arasındaki bağımlılık (CBO) değeri düşük

olduğundan, yazılımın verimlilik ve yeniden kullanırlık değerini yüksek olarak belirler (Şekil A.3).

Sınıfın tetiklediği metot sayısı (RFC) ve metotlardaki uyum eksikliği (LCOM) değerleri, bir sınıfta yazılan ve çağrılan toplam metot sayısı doğru orantılı olduğundan yazılımın anlaşılabilirlik, dayanıklılık, karmaşıklık, test edilebilirlik özelliklerini ölçebilir ve kabul edilebilir düzeydedir (Şekil A.5).

Önerilen modelde web üzerinde yer alan ya da web'e çıkışı mümkün olan sistemlerin bir yazılım aracılığı ile entegre olabilmektedir. Yazılan uygulama platform bağımsızdır. Veri kaynakları çeşitli tür ve boyuttadır. Bir dosya olabileceği gibi, küçük veya daha büyük ve karmaşık sistemler de olabilirler. Sistem kapsamını arttırmak amacıyla, çok yönlü veri akışına olanak tanıyan yapılar kullanılmıştır. Veriyi taşıyan web veri servisleridir. Uluslararası standartlara uygun üretilmiş olmaları, sisteme yeni entegre olan her türden veri, basit, hızlı ve kolay bir şekilde birleştirilmiş veriye eklenebilmektedir. Eklenen verilerin mutlaka kabul edilmiş web servisi standartlarını sağlayan yapıda olma şartı güdülmektedir. *Çalışmanın en önemli yeniliği web servislerini sistem kütüphanesine eklerken izlenen nesne tabanlı tanımlama mantığı ve algoritmasıdır.* Önerilen nesne tabanlı tanımlama sayesinde web servisleri yalnız kolay tanımlanmakla kalmaz, işlem sırasında kolay bulunur ve yerel değişiklikler önerilen yapıdaki ara katman işleyişini etkilememektedir. Tasarlanan DataOCEAN Mimarisi'nin birbirinden farklı sistemlerin veri alışverişinde bulunduğunu, özgün sorgulama platformu sunan arayüzü ile sorgulama anında sistemde aktif olan web servisleri arasından doğruluk derecesi en yüksek olan servis/servisler seçilerek, işlemi kısa sürede hızlı ve doğru yapabildiği gözlemlenmiştir. Web servislerin doğruluk derece değerleri, servisleri OceanLibrary kütüphanesine tanımlama aşamasında bir kabul doğrultusunda belirlenerek eklenmektedir. Eklenen web servisinin doğruluğu ham veriyi servis veritabanına ekleyen kurumun güvenilirliği doğrultusunda kararlaştırılmaktadır. Bu aşamada henüz kalıcı bir kriter belirlenmemiştir. Birbiriyle etkileşimde olan servislerin parametre bilgileri sürekli güncel tutulmaktadır. Sistem senkronizasyonunu sağlayan altyapı mevcuttur.

DataOCEAN sisteminin sonraki geliştirme aşamalarında kontrol katmanı aracılığı ile gönderilen talepler hazır SQL sorgular halinde, sürükle - bırak eklentisiyle gerçekleştirilmesi, kullanım kolaylığı arttırılması amacıyla hedeflenmektedir. Bu çalışma kapsamında da son kullnaıcıdan bir yazılım bilgisi beklenmemektedir.

DataOCEAN, servis yönelimli mimariye uygun tasarlanmış, nesne yönelimli mimari yapısına benzerlikler taşımaktadır. Kullanılan framework MVC mimari tabanlı, özgün bir çalışmadır. REST yaklaşımı ile tasarlanmış, RESTful ve SOAP servisleri hazırlanmış, sistemin yönetim panelinin içereceği alanlar yazılmış ve test edilmiştir. Uygulama dili olarak PHP tercih edilmiş, mekansal veri gösterimi için Google Earth kullanılmıştır.

Bilginin bilgisi, birbiri ile ayrı ayrı birer sistem olup efektif olarak kullanılan, yapıların entegre oldukları bölümde, anlık veri erişimi yanısıra güncelleme de aynı doğruluk ve hız çerçevesinde yapılmaktadır. Anlık sorgulanan verinin veritabanına eklenmesi ile ilgili bir algoritma geliştirildiğinde, DataOCEAN, uzun vadede mükerrer veri-kayıt içermeyen arşiv verisi de oluşturabilecek bir uygulamadır. Hızlı veri paylaşımına uygun mimarisi ile servis odaklı ve zamana bağlı bir model olarak son kullanıcının hizmetine sunulmuştur.

## KAYNAKLAR


[1] **A. P. Sheth and J. A. Larson.** (1990). Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys (CSUR), 22*(3), 183-236.

[2] **R. Jakobovits.** (1997). Integrating autonomous heterogeneous information sources. *Principles for Digital Library Development Communications of the ACM, 44*(5), 49-54.

[3] **V. Raman, I. Narang, C. Crone, L. Haas, S. Malaika, T. Mukai, D. Wolfson and C. Baru.** (2002). Data access and management services on grid. *GGF5*, http://www. cs. man. ac. uk/grid-db.

[4] **E. Franconi and A. Kamblet**. (2004). *A data warehouse conceptual data model.* Paper presented at the Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on Scientific and Statistical Database Managemen

[5] **E. Franconi and U. Sattler.** (1999). A data warehouse conceptual data model for multidimensional aggregation: a preliminary report. *Italian Association for Artificial Intelligence AI* IA Notizie, 1*, 9-21.

[6] **M. Jarke, M. Lenzerini, Y. Vassiliou and P. Vassiliadis**. (2013). *Fundamentals of data warehouses*: Springer Science & Business Media.

[7] **K. Bennett, M. Munro, J. Xu, N. Gold, P. Layzell, N. Mehandjiev, D. Budgen and P. Brereton**. (2002). *Prototype implementations of an architectural model for service-based flexible software.* Paper presented at the System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on.

[8] **M. M. Lehman**. (1996). *Laws of software evolution revisited.* Paper presented at the European Workshop on Software Process Technology.

[9] **T. Glatard, M.-É. Rousseau, S. Camarasu-Pop, R. Adalat, N. Beck, S. Das, R. F. da Silva, N. Khalili-Mahani, V. Korkhov, P.-O. Quirion, P. Rioux, S. D. Olabarriaga, P. Bellec and A. C. Evans.** (2017). Software architectures to integrate workflow engines in science gateways. *Future Generation Computer Systems, 75*, 239-255. doi: http://dx.doi.org/10.1016/j.future.2017.01.005

[10] **N. Ghadiri, M. Ghaffari and M. A. Nikbakht.** (2017). BigFCM: Fast, precise and scalable FCM on hadoop. *Future Generation Computer Systems, 77*, 29-39. doi: http://dx.doi.org/10.1016/j.future.2017.06.010

[11] **S. Al-Kiswany, L. B. Costa, H. Yang, E. Vairavanathan and M. Ripeanu.** (2017). A cross-layer optimized storage system for workflow applications. *Future Generation Computer Systems, 75*, 423-437. doi: http://dx.doi.org/10.1016/j.future.2017.02.038

[12] **D. E. Perry and A. L. Wolf.** (1992). Foundations for the study of software architecture. *SIGSOFT Softw. Eng. Notes, 17*(4), 40-52. doi: 10.1145/141874.141884

[13] Url -1 (2017). integration. Retrieved 10.11.2017, 2017, from <http://www.turkceanlaminedir.com/integration-34917>

[14] Url -2 (2017). integration2. Retrieved 10.11.2017, 2017, from <http://www.toplumdusmani.net/modules/wordbook/entry.php?entryID=2635/entegrasyon-nedir+entegrasyon-ne-demek>

[15] Url -3 (2017). integration_dy. Retrieved 09.11.2017, 2017, from <http://www.endustri40.com/yatay-ve-dikey-entegrasyon-nedir/>

[16] **O. M. Duschka, M. R. Genesereth and A. Y. Levy.** (2000). Recursive query plans for data integration. *The Journal of Logic Programming, 43*(1), 49-73.

[17] **M. Lenzerini**. (2002). *Data integration: A theoretical perspective.* Paper presented at the Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems.

[18] **K. Bennett, P. Layzell, D. Budgen, P. Brereton, L. Macaulay and M. Munro**. (2000). *Service-based software: the future for flexible software.* Paper presented at the Software Engineering Conference, 2000. APSEC 2000. Proceedings. Seventh Asia-Pacific.

[19] **C. Ghezzi**. (2002). *Ubiquitous, decentralized, and evolving software: Challenges for software engineering.* Paper presented at the International Conference on Graph Transformation.

[20] **H. Kreger.** (2001). Web services conceptual architecture (WSCA 1.0). *IBM Software Group, 5*, 6-7.

[21] Url -4 (2017). Microsoft. Retrieved 26.03.2017, 2017, from <www.microsoft.com>

[22] **P. C. Muehrcke.** (1978). MAP USE, Reading, Analysis, and Interpretation. *Madison, WI: J.P. Publications*.

[23] (2015). Oracle 8i. Retrieved 12.02.2015, 2015, from https://docs.oracle.com/cd/A87860_01/doc/server.817/a76965.pdf

[24] **C. Chomicki.** (1994). *Temporal Query Languages: A Survey*. Paper presented at the Temporal Logic, First International Conference, Verlag, LNAI 827.

[25] **J. Wiles and J. Watson.** (2005). Patterns in complex systems modeling. In M. Gallagher, J. Hogan and F. Maire (Eds.), *Intelligent Data Engineering and Automated Learning Ideal 2005, Proceedings* (Vol. 3578, pp. 532-539).

[26] **T. Francis, E. Herness, R. High Jr, J. Knutson, K. Rochat and C. Vignola**. (2004). *Professional IBM WebSphere 5.0 Application Server*: John Wiley & Sons.

[27] Url -5 (2015). INSPIRE. Retrieved 02.01.2015, 2015, from <http://inspire.jrc.ec.europa.eu/>

[28] Url -6 (2015, 28.11.2017). OGC. <http://www.opengeospatial.org/standards/sfs>. Retrieved 04.04.2015, 2017, from <http://www.opengeospatial.org/>

[29] **F. Zhu, M. Turner, I. Kotsiopoulos, K. Bennett, M. Russell, D. Budgena, P. Breretona, J. Keane, P. Layzell and M. Rigby**. (2004). *Dynamic data integration using web services.* Paper presented at the Web Services, 2004. Proceedings. IEEE International Conference on.

[30] **M. P. Papazoglou and W.-J. Heuvel.** (2007). Service oriented architectures: approaches, technologies and research issues. *The VLDB Journal—The International Journal on Very Large Data Bases, 16*(3), 389-415.

[31] **C. Bussler**. (2013). *B2B integration: Concepts and architecture*: Springer Science & Business Media.

[32] **A. Umar and A. Zordan.** (2009). Reengineering for service oriented architectures: A strategic decision model for integration versus migration. *Journal of Systems and Software, 82*(3), 448-462.

[33] **K. Channabasavaiah, K. Holley and E. Tuggle.** (2003). Migrating to a service-oriented architecture. *IBM DeveloperWorks, 16*.

[34] **S. Burbeck.** (2000). The tao of e-business services: The evolution of web applications into service-oriented components with web services. *IBM DeveloperWorks, October*.

[35] **S. Furnell, J. Kim and K. Lim.** (2007). An approach to service-oriented architecture using web service and BPM in the telecom-OSS domain. *Internet research, 17*(1), 99-107.

[36] **O. Marjanovic**. (2006). *BPM-Bridging the gap between Business Processes and technologies for Process Management.* Paper presented at the Full Proceedings of the 2nd International Conference on Information

Management and Business (IMB2006) Sydney, Australia www. aimb. org 13-16 Feb, 2006.

[37] **D. McGoveran.** (2004). An introduction to BPM and BPMS. *Bus. Integr. J*, 2-10.

[38] **A. Arsanjani.** (2002). Introduction. *Communications of the ACM, 45*(10), 30-34.

[39] **H. Kreger.** (2003). Fulfilling the Web services promise. *Communications of the ACM, 46*(6), 29-ff.

[40] **D. Krafzig, K. Banke and D. Slama**. (2005). *Enterprise SOA: service-oriented architecture best practices*: Prentice Hall Professional.

[41] Url -7 **w3c**. Retrieved 09.02.2017, from <https://www.w3.org/TR/soap12-part1/>

[42] **X. Shi.** (2006). Sharing service semantics using SOAP-based and REST Web services. *IT Professional, 8*(2), 18-24.

[43] Url -8 SOAP. Retrieved 15.02.2017, 2017, from <http://www.w3.org/TR/soap12-part1/>

[44] **D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S. Thatte and D. Winer.** (2000). Simple object access protocol (SOAP) 1.1.

[45] Url -9 WSDL. Retrieved 15.02.2017, 2017, from <http://www.w3.org/TR/wsdl20-primer/>

[46] **S. Overhage**. (2002). *On Specifying Web Services Using UDDI Improvements.* Paper presented at the 3rd Annual International Conference on Object-Oriented and Internet-based Technologies, Concepts, and Applications for a Networked World Net. ObjectDays, Germany

[47] Url -10 (2017). UDDI Spec TC. Retrieved 14.02.2017, 2017, from <http://www.uddi.org/pubs/uddi-v3.0.2-20041019.htm>

[48] **R. T. Fielding**. (2000). *Architectural styles and the design of network-based software architectures.* University of California, Irvine.

[49] **J. Yang.** (2003). Web service componentization. *Communications of the ACM, 46*(10), 35-40.

[50] **A. Dhesiaseelan and A. Ragunathan**. (2004). *Web services container reference architecture (WSCRA).* Paper presented at the Web Services, 2004. Proceedings. IEEE International Conference on.

[51] **A. Anagol-Subbaro.** (2005). J2EE Web Services on BEA WebLogic: Prentice-Hall, Upper Saddle River.

[52] **D. Chappell**. (2004). *Enterprise service bus*: " O'Reilly Media, Inc.".

[53] **P. Sarang**. (2006). *Pro Apache XML*: Apress.

[54] **C. Pautasso, O. Zimmermann and F. Leymann**. (2008). *Restful web services vs. big'web services: making the right architectural decision.* Paper presented at the Proceedings of the 17th international conference on World Wide Web.

[55] Url -11 Programmable Web. Retrieved 16.02.2017, 2017, from <https://www.programmableweb.com/api-research>

[56] **J. Meng, S. Mei and Z. Yan**. (2009). *Restful web services: A solution for distributed data integration.* Paper presented at the Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on.

[57] **L. Richardson and S. Ruby**. (2008). *RESTful web services*: " O'Reilly Media, Inc.".

[58] **L. Richardson, S. Ruby and T. Demmig**. (2007). *Web-services mit REST*: O'Reilly Germany.

[59] **S. Vinoski.** (2008). Restful web services development checklist. *IEEE Internet Computing, 12*(6).

[60] **E. Özdikililer and Ç. Göksel.** (2017). DATAOCEAN : A MODEL FOR INTEGRATED INFORMATION SYSTEMS WITH OBJECT-ORIENTED LOGIC AND ALGORITHM. *SYLWAN, 161*(8), 18.

[61] **E. Söderström and F. Meier.** (2007). Combined SOA maturity model (CSOAMM): Towards a guide for SOA adoption *Enterprise Interoperability II* (pp. 389-400): Springer.

[62] **Y. Baghdadi.** (2012). A methodology for web services-based SOA realisation. *International Journal of Business Information Systems, 10*(3), 264-297.

[63] **N. Pathak**. (2011). *Pro WCF 4: Practical Microsoft SOA Implementation*: Apress.

[64] **A. Bahree, D. Mulder, S. Cicoria, C. Peiris and N. Pathak**. (2007). *Pro WCF: practical Microsoft SOA implementation*: Apress.

[65]

[66] Url -12 **php**. (2017). PHP. Retrieved 10.07.2017, 2017, from <http://php.net/>

[67] Url -13 **J.-F. Lepine**. (2017). Metrics for PHP. Retrieved 14.11.2017, 2017, from <http://www.phpmetrics.org//documentation/index.html>

## EKLER

**EK A.1 :** Yazılım Ölçütleri - Metrik Değerlendirme;

ISO 9126' ya göre kalite sınıfları ve alt sınıfları için aşğıdaki gibi sıralanabilir:

- İşlevsellik: Yazılımın talep edilen unsurları gerçekleştirme özelliğidir. İşleyişte doğruluk, birlikte çalışabilirlik, güvenlik ve uygunluk gibi konuları içermektedir.
- Güvenilirlik: Yazılımın doğdu çalışmasını kapsar. Bu kapsamda hata toleransı ve ihtiyaç halinde veri kurtarma ve önceki versiyonlara geri dönebilme özelliğini tanımlar.
- Kullanılabilirlik: Yazılımın kullanım kolaylığı sunma özelliğidir., Anlaşılabilirlik ve akıcı kullanıcı etkileşimi konularını kapsar.
- Verimlilik: Yazılımın ihtiyaç duyulan performansla çalışabilme yeteneğini tanımlar. Zaman ve kaynak kullanımı incelenmekltedir.
- Değiştirilebilirlik: Yazılımın düzeltme veya istenen değişiklişklere uyarlanma yeteneği olarak tanımlanır. Test ve analiz edilebilirlik özelliklerini de kapsar.
- Taşınabilirlik: Yazılımın çalışma platformu değişim ihtiyaçlarına uyumluluk yeteneğini tanımlar. Adaptasyon, yeniden yüklenebilirlik, ortam değişikliğine imkan tanımlama özelliği incelenir.

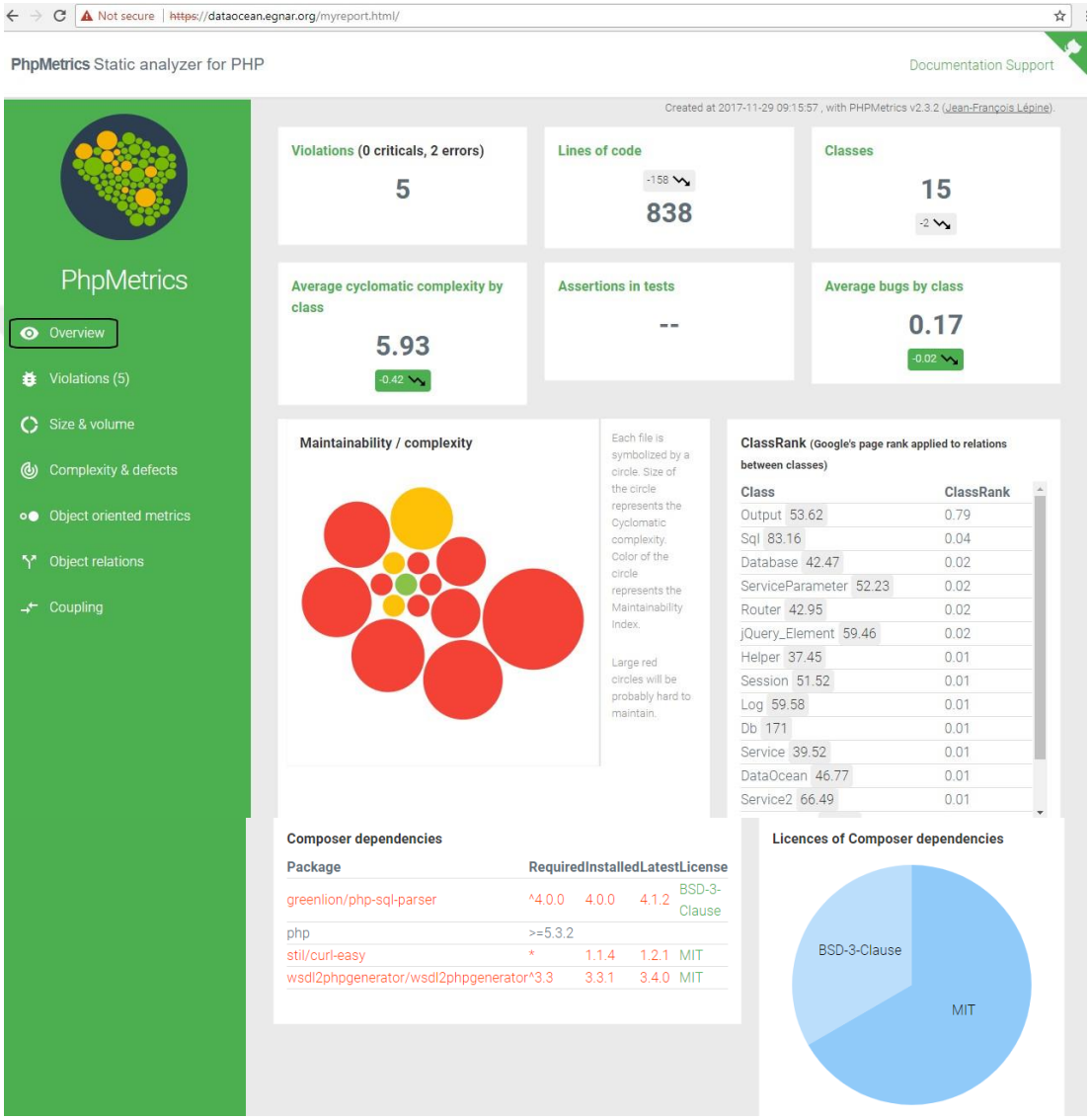
Tasarlanan modelde ve sonrasında yapılan yazılımda yer alan nesneler, sınıflar, aralarındaki ilişkiler, ilişkilerin yapısal ve davranışsal özellikleri nesne yönelimli tasarım metrikleri kullanılarak değerlendirilir. Nesne yönelimli tasarım metrikleri etkinlik, verimlilik, karmaşıklılık, anlaşılabilirlik, yeniden kullanılabilirlik, test edilebilirlik ve dayanıklılık olarak sıralanabilir. Bu metrikler kalite düzeyinin ölçülmesinde kullanılır. Ölçümlerin yapılabilmesi için ise, metrik ölçüm kümeleri

kullanılır. Bunlardan bazıları Chidamber & Kemerer, Brito'e Abreu, Bansiya & Davis metrik kümeleridir.

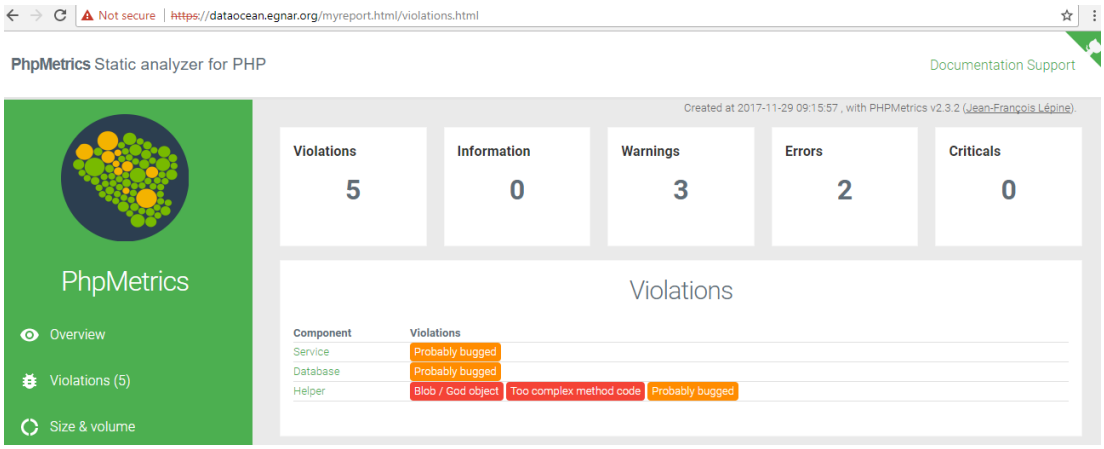
Çalışmanın yazılım ölçütleri değerlendirilme kapsamında en yaygın kullanılan Chidamber & Kemerer'in tanımladığı C&K olarak bilinen metrik kümesi kullanılmıştır. CK kümesinin barındırdığı metrik değerleri aşağıdaki gibi sıralanabilir:

- Sınıfın Ağrılıklı Metod Sayısı (WMC): Bir sınıftaki metodların karmaşıklık derecesi veya sayısıdır. Metotların karmaşıklık veya sayısı, sınıfın geliştrilmesine ve bakımına harcanacak zaman-çaba oranı hakkında bilgilendirmektedir. Bir sınıfın "anlaşılabilirlik, yeniden kullanılabilirlik ve dayanıklılık" ölçütü üzerinde yorum yapmak mümkündür.
- Kalıtım Ağacının Derinliği (DIT): Bir sınıfın kalıtım ağacının köküne olan uzaklığını gösteren metriktir. Bu metrik yardımıyla yazılımın verimliliğini, yeniden kullanımını, anlaşılırlığını, test edilebilirliğini ölçmek mümkündür. Sayısal metrik değeri test edilebilirlik ve verimlilik yetenek özelliği ile ters orantılıdır.
- Alt Sınıf Sayısı (NOC): Bir sınıftan doğrudan türetilmiş alt sınıfların sayısını gösteren metriktir. Eğer alt sınıf sayısı fazla ise yeniden kullanım yüksek, hata riski fazladır. Dolayısıyla, bu metrik ile yazılımın verimlilik, yeniden kullanılabilirlik, test edilebilirlik gibi özelliklerini ölçmek mümkündür.
- Nesne Sınıfları Arasındaki Bağımlılık (CBO): Bir sınıf içindeki özellik ya da metotların diğer sınıf ya da sınıflarda kullanılması ve sınıflar arasında kalıtımın olmaması durumunda iki sınıf arasında bağımlılık ilişkisi vardır. Bu metrik ile yazılımın verimlilik, yeniden kullanırlık değerleri ölçülebilir.
- Sınıfın Tetiklediği Metot Sayısı (RFC): Bir sınıftan bir nesnenin metotları çağrılması durumunda, bu nesnenin tetikleyebileceği tüm metotların sayısı RFC değerini verir.Kısaca bir sınıfta yazılan ve çağrılan toplam metot sayısıdır. Bu metrik kullanılarak yazılımın anlaşılabilirlik, dayanıklılık, karmaşıklık, test edilebilirlik gibi özelliklerini ölçmek söz konusudur.
- Metotlardaki Uyum Eksikliği (LCOM): N adet kümenin kesişiminden oluşan kümelerdeki uyumsuzlukların sayısıdır ve metotlardaki benzerlik

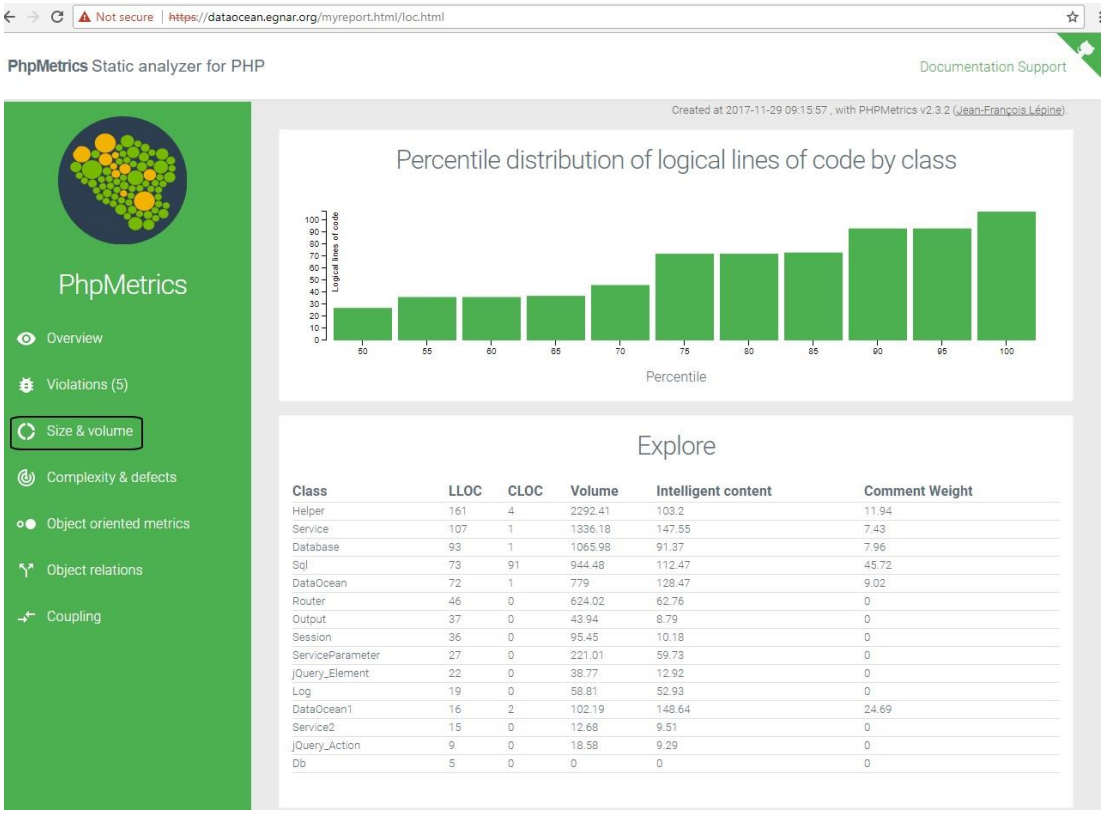
derecesini ölçer. Bu metrik değeri kullanılarak yazılımın verimlilik , yeniden kullanılabilirlik gibi değerlerinin ölçülmesi mümkündür.



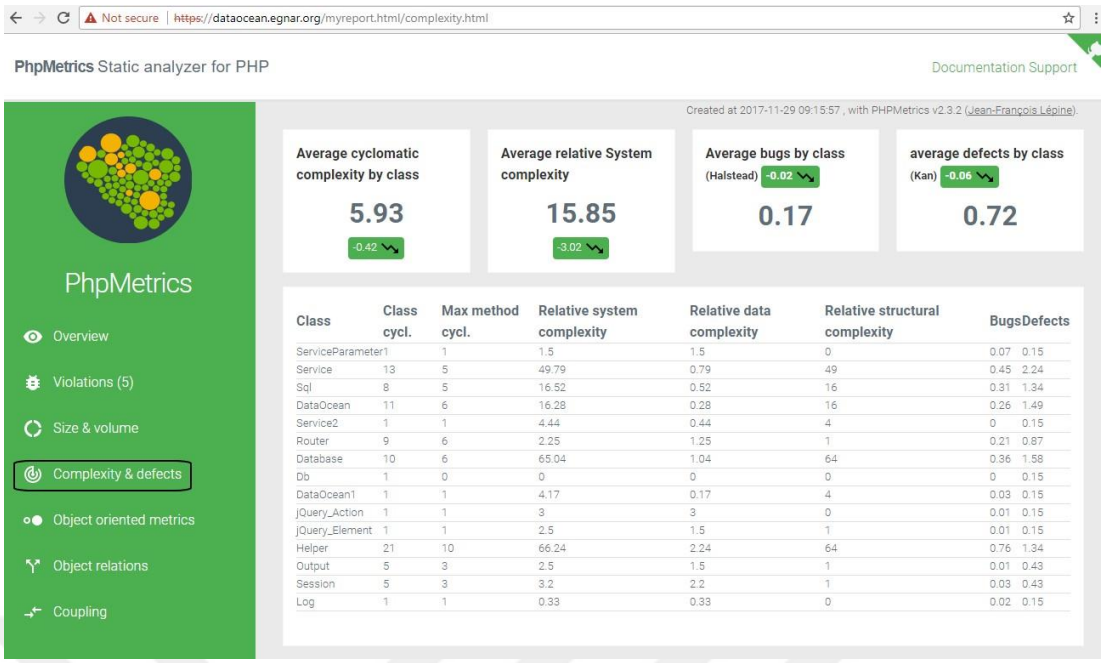

**Şekil A.1 :** PHP Metrics genel inceleme (Overview).



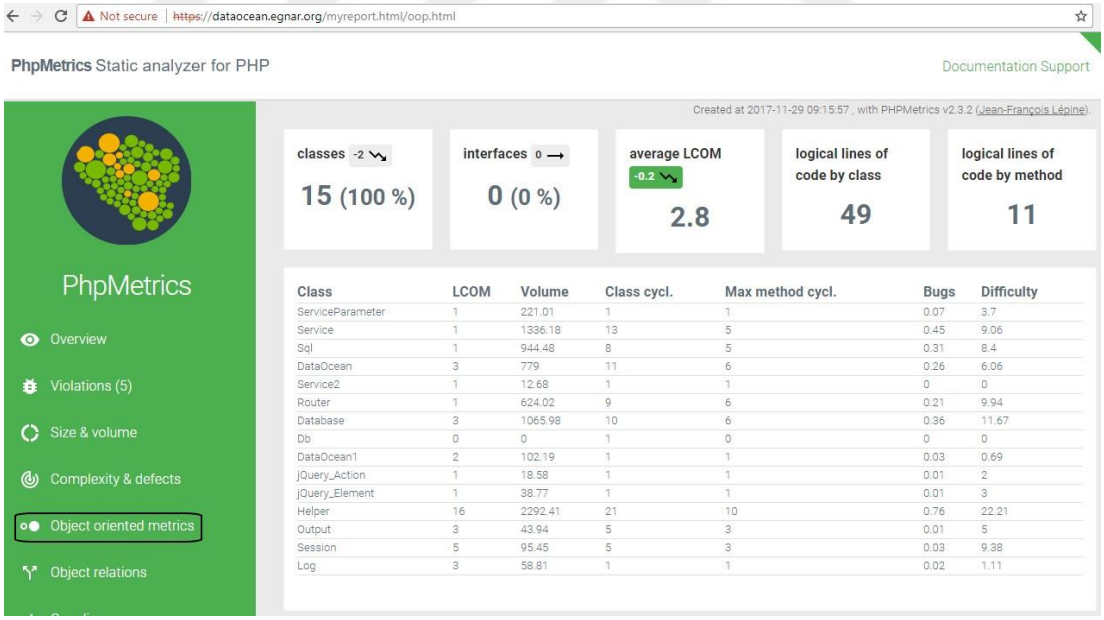

**Şekil A.2 :** İhlaller (Violations).



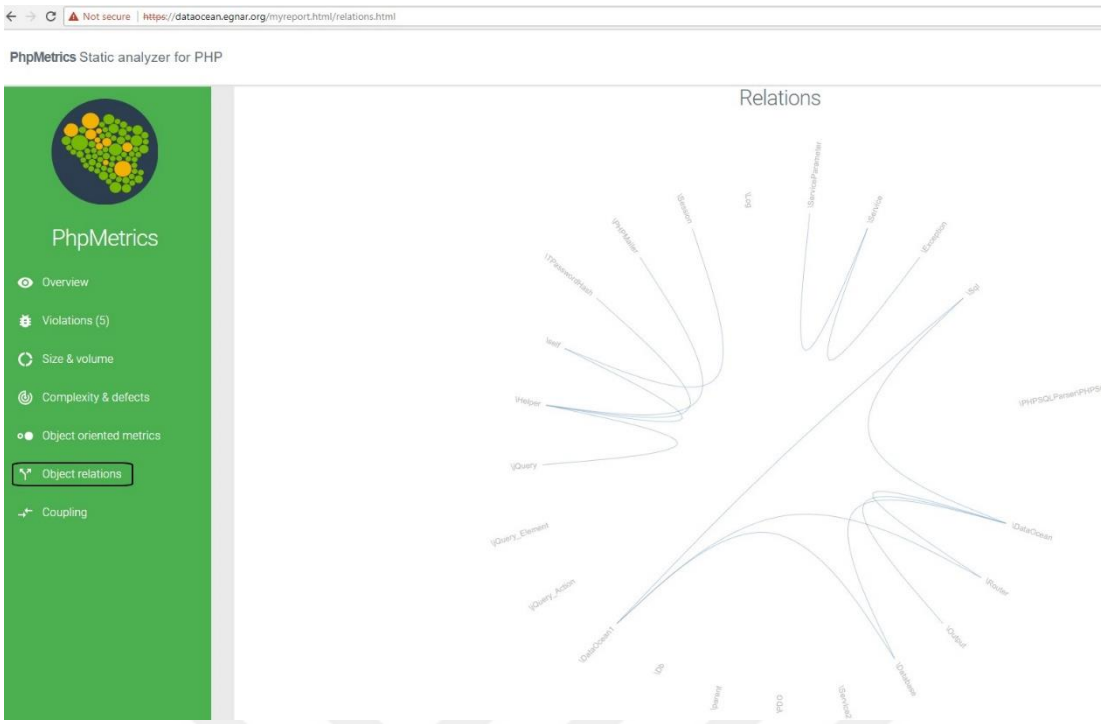

**Şekil A.3 :** Boyut ve hacim metrikleri (Size & Volume)



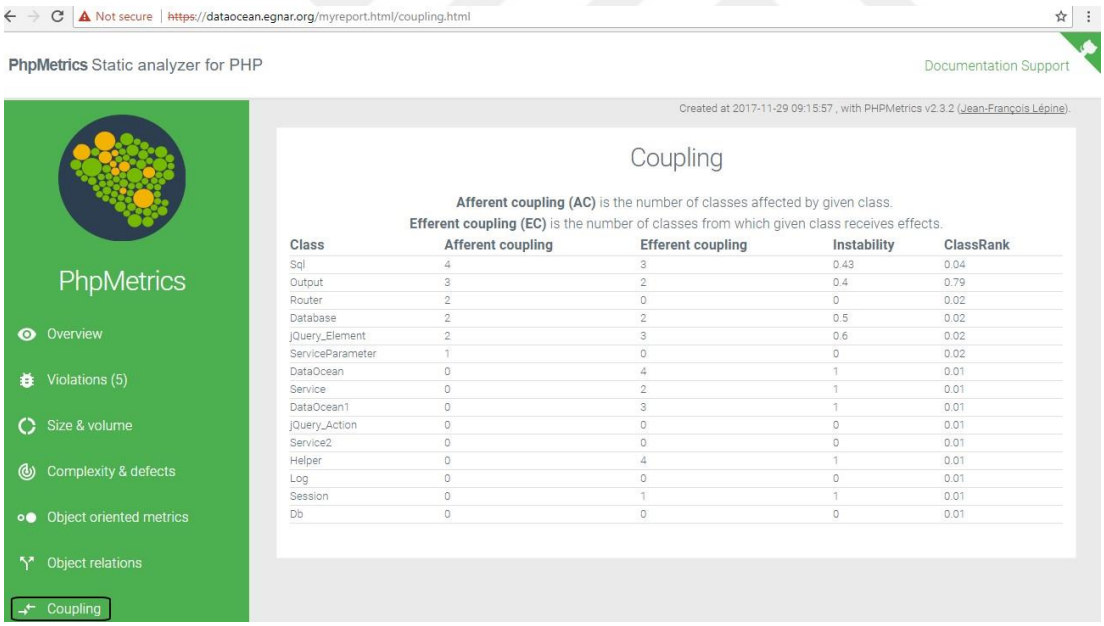

**Şekil A.4 :** Karmaşıklık ve kusurlar (Complexity & defects).




**Şekil A.5 :** Nesneye dayalı metrikler (Object Oriented Metrics)




**Şekil A.6 :** Yapılar arasındaki etkileşim. (Object Relations)




**Şekil A.7 :** Gevşek bağ bilgisi. (Coupling)

## ÖZGEÇMİŞ

**Ad Soyad:** Egnar ÖZDİKİLİLER

**Doğum Yeri ve Tarihi:** Bulgaristan - 1978

**E-Posta:** ozdikililer@itu.edu.tr

**Lisans:**

Selçuk Üniversitesi, 1996-2000

Mühendislik Mimarlık Fakültesi, Bilgisayar Mühendisliği

**Yüksek Lisans:**

Trakya Üniversitesi, (2002-2004)

Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği

**Mesleki Deneyim ve Ödüller:**

09.2003 - Halen İstanbul Teknik Üniversitesi

Yazılım Mühendisi/Veritabanı Yöneticisi

03.2013 - Halen Uydu Haberleşmesi Uzaktan Algılama UygAr Merkezi

03.2010- 02.2013 Öğrenci İşleri Daire Başkanlığı

10.2006- 02.2010 Bilgi İşlem Daire Başkanlığı

09.2003- 09.2006 Personel Daire Başkanlığı

06.2002- 05.2003 Doğan Holding A.Ş.-Yazılım Yönetmeni/Veritabanı Yöneticisi

04.2001- 05.2002 İnfoTRACE A.Ş. – Yazılım Mühendisi

08.2000- 03.2001 Özak Elektronik A.Ş. – Planlama ve Proje Mühendisi

## TEZDEN TÜRETİLEN YAYINLAR/SUNUMLAR

- **Özdikililer E.**, Göksel Ç., 2017: DATAOCEAN: A Model for Integrated Information Systems with Object-Oriented Logic and Algorithm, ISSN: 0039-7660, SYLWAN.English Edition Journal, Article, Vol:2017 – Issue:08, 2017 Warszawa, Poland.

- **Özdikililer E.**, Göksel Ç., Paker S.: 2017: Design a Model for Distributed Systems using Web Services, *International Conference on Engineering Technologies*, December 07-09, 2017 Konya, Turkey.