T.C.

YILDIZ TEKNİK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

VERİTABANLARININ TERSİNE MÜHENDİSLİĞİ İLE WEB SERVİSLERİNİN OTOMASYONU

Mehmet Raşid GENCOSMANOĞLU

YÜKSEK LİSANS TEZİ

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

Danışman

Dr. Öğretim Üyesi Yunus Emre SELÇUK

Temmuz, 2019

T.C.

YILDIZ TEKNİK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

VERİTABANLARININ TERSİNE MÜHENDİSLİĞİ İLE WEB SERVİSLERİNİN OTOMASYONU

Mehmet Raşid GENCOSMANOĞLU tarafından hazırlanan tez çalışması 11.07.2019 tarihinde aşağıdaki jüri tarafından Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı, Bilgisayar Mühendisliği Programı **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Tez Danışmanı

Dr. Öğretim Üyesi Yunus Emre SELÇUK

Yıldız Teknik Üniversitesi

Jüri Üyeleri

Dr. Öğretim Üyesi Yunus Emre SELÇUK
Yıldız Teknik Üniversitesi _____

Prof. Dr. Oya KALIPSIZ
Yıldız Teknik Üniversitesi _____

Prof. Dr. Selim AKYOKUŞ
İstanbul Medipol Üniversitesi _____

Danışmanım Dr. Öğretim Üyesi Yunus Emre SELÇUK sorumluluğunda tarafımca hazırlanan Veritabanlarının Tersine Mühendisliği ile Web Servislerinin Otomasyonu başlıklı çalışmada veri toplama ve veri kullanımında gerekli yasal izinleri aldığımı, diğer kaynaklardan aldığım bilgileri ana metin ve referanslarda eksiksiz gösterdiğimi, araştırma verilerine ve sonuçlarına ilişkin çarpıtma ve/veya sahtecilik yapmadığımı, çalışmam süresince bilimsel araştırma ve etik ilkelerine uygun davrandığımı beyan ederim. Beyanımın aksinin ispatı halinde her türlü yasal sonucu kabul ederim

Mehmet Raşid GENCOSMANOĞLU

İmza

Aileme

TEŞEKKÜR

Tez çalışmam boyunca değerli desteğinden ötürü danışmanım Dr. Öğretim Üyesi Yunus Emre SELÇUK'a teşekkürlerimi sunarım. Araştırma ve tez yazım süreci boyunca verdiği tavsiye ve yönlendirmeler sayesinde yüksek lisans tezimi sorunsuz bir şekilde tamamlayabildim. Tez yazım sürecinde desteklerini ve fedakarlıklarını esirgemeyen aileme ve eşime de teşekkürü borç bilirim.

Mehmet Raşid GENCOSMANOĞLU

İÇİNDEKİLER

KISALTMA LİSTESİ

API	Application Programming Interface
APIGEN	API Generator
BE	Backend
CG	Code Generation
CL	Change Listener
CLI	Command Line Interface
CRUD	Create Read Update Delete
FE	Frontend
HTTP	Hypertext Transfer Protocol
JPA	Java Persistence API
JSON	JavaScript Object Notation
MPA	Multi Page Application
NPM	Node Package Manager
ORM	Object Relational Mapping
POJO	Plain Old Java Objects
RE	Reverse Engineering
REST	Representational State Transfer
SOA	Service Oriented Architecture
SPA	Single Page Application
UI	User Interface
WS	Web Service

ŞEKİL LİSTESİ

TABLO LİSTESİ

ÖZET

Veri Tabanlarının Tersine Mühendisliği İle Web Servislerinin Otomasyonu

Mehmet Raşid GENCOSMANOĞLU

Bilgisayar Mühendisliği Anabilim Dalı

Yüksek Lisans Tezi

Danışman: Dr. Öğretim Üyesi Yunus Emre SELÇUK

Günümüzde birçok şirketin veritabanında bulunan verilerini REST tabanlı web servisleri aracılığıyla üçüncü şahıs firmaların kullanımına açması oldukça yaygın bir yaklaşımdır. REST(Representational State Transfer), istemci ile sunucu arasında hızlı ve kolay iletişimi sağlayan HTTP tabanlı bir web servis mimarisidir. Bu sebeple birçok IOT ve mobil uygulamada bu mimari tercih edilmektedir. Veritabanları için RESTful servislerin hızlı bir şekilde geliştirilmesini sağlayan birçok yazılım kütüphanesi mevcut olsa da verinin yapısında meydana gelen bir değişim geliştirilen web servisini geçersiz kılabileceği gibi yeni bir geliştirme eforuna neden olacaktır. Bununla birlikte farklı geliştiriciler tarafından geliştirilen birbirine benzeyen servislerin olması (Örn : CRUD: Create, Read, Delete, Update işlemleri) bir standarda uyulmasını zor hale getirmektedir. Yazılım otomasyonu perspektifinden bakıldığında bir veritabanının REST tabanlı web servislerine dönüştürülmesi otomatize edilmesi gereken bir süreçtir. Bu sayede diğer otomasyon uygulamalarında olduğu gibi birim iş için gereken efor azalabileceği gibi ortaya çıkan ürünün kalitesinin de artma potansiyeli olacaktır. Veritabanlarından web servisleri üreten önceki çalışmalar ve araçlar otomasyonu belli bir seviyeye kadar

sağlasa da bu çalışmaların çoğu halihazırda varolan bir veritabanı için çözüm üretmemektedir. Ayrıca önceki çalışmalarda veritabanında meydana gelen değişimlere duyarlı bir yapı bulunmadığından yeni geliştirme eforu gerekebilmektedir. Tez çalışmasında önceki çalışmaların eksik ve iyileştirilebilir yönlerini de ele alarak kendi yaklaşımımızla web servislerini otomatize bir şekilde üretmeyi hedefledik. Bu bağlamda tersine mühendislik ve kod üretimi yöntemleri ile veritabanından REST uyumlu web servislerini otomatize bir şekilde üreten, değişimlere duyarlı APIGEN isimli yeni bir araç geliştirdik.

Anahtar Kelimeler: API Otomasyonu, Yazılım Otomasyonu, Web Servisleri, Tersine Mühendislik

ABSTRACT

Web Service Automation By Database Reverse Engineering

Mehmet Raşid GENCOSMANOĞLU

Department of Computer Engineering

Master of Science Thesis

Advisor: Dr. Yunus Emre SELÇUK

Nowadays, it is a common approach for companies to expose their data that exist in a database through RESTful APIs for variety of clients. REST (Representational State Transfer) is an HTTP-based web service architecture that enables quick and easy communication between client and server. For this reason, REST has started to take a place as a preferred approach in IoT and mobile applications. Although there are many libraries that provide rapid development of RESTful web services with various programming languages, any subsequent modification to the data source requires a new development effort and rewriting the boilerplate code. Any changes that occur in a relational database table is a good. Moreover, the fact that there are similar services (i.e. CRUD: Create-Read-Update-Delete operations) written by different developers makes it difficult to comply with the standards and best practices. In terms of software automation, exposing a database to the world with RESTful architecture is a process that needs to be automated. Also every automated process has the potential to reduce effort and improve quality. While existing tools provide automation to a certain extent, most of these tools don't have a solution for an entire database schema and extra development effort is required when a customization is needed. Besides, in previous studies, there is a possibility for services generated by

automation process to lose their validity when any modification happens to database. In this paper, while discussing the improvable aspects of previous works, we will be presenting our approach with a tool named APIGEN (API Generator) converting a database into RESTful services in an automated manner.

Keywords: API Automation, Software Automation, Web Services, Reverse Engineering

1
GİRİŞ

1.1 Literatür Özeti

Günümüzde birçok şirketin veritabanında bulunan veriyi REST tabanlı web servisleri aracılığıyla üçüncü şahıs firmaların kullanımına açması oldukça yaygın bir yaklaşımdır. REST (Representational State Transfer), istemci ile sunucu arasında hızlı ve kolay iletişimi sağlayan HTTP tabanlı bir web servis mimarisidir. Bu mimaride istemci, hedef sunucudaki veriye ulaşabilmek veya veri üzerinde değişiklik yapabilmek için HTTP isteklerinde bulunur. Sunucu da bu isteklere yine HTTP cevabı ile yanıt verir. Veritabanlarını otomatize bir şekilde RESTful servislere dönüştürmeyi amaçlayan akademik ve ticari seviyede farklı çalışmalar mevcuttur. S-Case ismi verilen bir çalışmada fonksiyonel gereksinimlerden yola çıkılarak RESTful servisler üretilmiştir[1,2]. Model tabanlı tekniklerin kullanıldığı bu çalışmada geliştirici verimliliği, sunulan bir kullanıcı arayüzü yardımıyla arttırılmıştır.

Diğer bir çalışmada üretilecek web servislerinin özellikleri RSDL (Rest Service Description Language) meta modeli yardımıyla belirlenmeye çalışılmıştır[3]. Bu çalışmada farklı istemci tipleri için istemci kütüphaneleri ve test senaryoları otomatize bir şekilde üretilebilmiştir.

JHipster ve EMF-REST ürünleri de model tabanlı yaklaşımlarla web servisi kodu üretmede oldukça başarılılar[4, 5]. Bu çalışmaların tümü model-first yaklaşımını kullandığı için halihazırda varolan bir veritabanını RESTful servislere dönüştürme yeteneğine sahip değiller. Birçok uygulamada veritabanı tasarımı öncelikli olduğundan özellikle eskiden kalan sistemler (legacy systems) için veritabanlarından web servislerinin üretilmesi ihtiyacı olduğunda bu ürünler beklentiyi karşılamayabilmektedir.

Bir önceki paragrafta değinilen çalışmaların aksine, Google Telosys [6], Spring Roo [7], RestifyDB[8], Dream Factory [9] araçları APIGEN'e benzer şekilde tersine

mühendislik yöntemini kullanarak halihazırda bulunan bir veritabanı için web servisleri sağlayabilmektedir. Google Telosys ve Spring Roo ürünleri sundukları Eclipse eklentisi ile RESTful servis kodu üretmeyi sağlamışlardır. Fakat bu iki ürünün odaklandığı asıl nokta web servisi üretmekten çok kullanıcı arayüzü de olan bir web uygulaması üretmek olmuştur. Baghdadi tarafından gerçekleştirilen bir çalışmada ise web servis kodu şu iki patern ile üretilmiştir: Şema dönüşüm deseni ve CRUD işlemleri deseni[10]. Sonuç olarak SOAP tabanlı servisler en sık kullanılan veritabanı sorguları düşünülerek üretilmiştir. Biz ise bu çalışmada RESTful servisleri otomatize bir şekilde üretmeyi hedefledik. Değinilen önceki çalışmalarda üretilen web servislerinin sağladığı basit veritabanı operasyonlarının (CRUD : Create, Read, Delete, Update) dışında değişiklik veya özelleştirmeler için geliştirme eforu gerekmektedir. Ayrıca web servisleri otomatize bir şekilde üretildikten sonra veritabanlarında meydana gelebilecek değişimler bu çalışmalarda ele alınmamıştır. Bu nedenle üretilen servisler veritabanında meydana gelen herhangi bir değişiklik sonrası geçerliliğini yitirebilmektedir. Kendi çalışmamızda veritabanı değişimlerine duyarlı bir gerçekleme ile bu problemin önüne geçmeyi hedefledik. Sonuç olarak kullanıcıları veritabanında meydana gelen değişimler hakkında bilgilendirerek basit önerilerde bulunabildik.

Sonuç başlığı altında önceki çalışmalar ile kendi çalışmamız olan APIGEN ürününün özellikleri Tablo 2'de karşılaştırmalı olarak sunulmuştur.

1.2 Tezin Amacı

Bu tez çalışmasının temel hedefi, ilişkisel veritabanlarında bulunan tablolardan geliştirme eforu gerektirmeksizin, kullanıcı dostu bir arayüz yardımıyla web servisleri üretilmesidir. Bununla birlikte veritabanlarında meydana gelen değişimlere duyarlı bir yapı oluşturularak, otomatize üretilen web servislerin geçerliliğinin korunması da hedeflenmiştir.

1.3 Hipotez

Bu tez çalışmasında aşağıdaki araştırma sorularının yanıtları elde edilmeye çalışılmıştır.

- RESTful servislerin, geliştirme çabasına ihtiyaç duymadan kullanıcılar tarafından özelleştirilebilir mi?
- Veritabanındaki değişiklikler hakkında kullanıcıları bilgilendirebilecek değişime duyarlı bir yapı oluşturulabilir mi?
- RESTful servislerin otomasyonunda kullanmak üzere kullanıcı dostu bir web arayüzü sağlanabilir mi?

Bu bölümden sonraki başlıklarda tez çalışmasının teknik detaylarına değinilecektir. Öncelikle Sistem Tasarımı başlığı altında çalışmamızın genel mimarisi anlatılacaktır. Yine aynı bölümde genel mimari içerisinde servis otomasyonunu oluşturan tüm alt modüllerin görevleri ve modüller arası iletişimin detayları anlatılacaktır. Hedefler ve Kısıtlar başlığı altında tez çalışması ile hedeflenen maddeler ve teknik kısıtlar aktarılacaktır. Kullanım Senaryosu başlığında ise bir son kullanıcının sisteme girişinden servis otomasyonunun son aşamasına kadar hangi adımlardan geçmesi gerektiği gösterilecektir.

Giriş bölümünden sonra "Gerçekleme Detayları" ana başlığı altında APIGEN projesinin tüm modülleri detaylandırılmaktadır. Bu bölüm, projenin teknik gerçekleme detaylarını içermektedir. Tersine Mühendislik kullanılarak veri modellerinin nasıl oluşturulduğu, kullanıcıların APIGEN önyüzü aracılığı ile seçtiği filtrelerden web servislerinin nasıl üretildiği ve değişime duyarlı otomasyonun gerçekleme detaylarına bu bölümde yer verilmektedir. Dokümanın son bölümleri ise UI (Kullanıcı Arayüzü), Testler ve Sonuç başlıklarından oluşmaktadır.

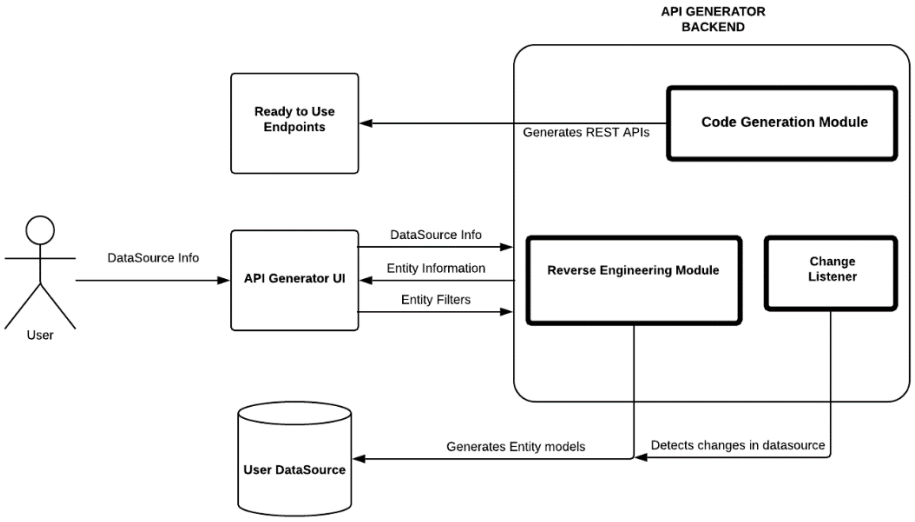
2
ANALİZ VE TASARIM

2.1 Sistem Tasarımı

Tez çalışması kapsamında bir veritabanından web servislerinin oluşturulmasını sağlayacak yardımcı bir araç (scaffolding tool) geliştirilmiştir. APIGEN(API Generator) ismini verdiğimiz uygulamanın çıktısı deploy edilebilir bir Spring Framework projesidir. Spring Framework, Java tabanlı modern uygulamalar için her türlü dağıtım platformunda kapsamlı bir programlama ve yapılandırma modeli sunar[11]. APIGEN aracı FE (Frontend) ve BE (Backend) olmak üzere iki ana modül ve alt modüllerden oluşmaktadır.

FE modülü SPA tabanlı bir web arayüzüdür. VueJS kütüphanesi kullanılarak geliştirilmiştir[12]. Bu modül servis otomasyonu için kullanıcıların veritabanı ile ilgili bilgileri sisteme gireceği ve üretilecek servislerin özelliklerini belirleyebilecekleri arayüzlerden oluşmaktadır.

BE modülü ise Spring Boot tabanlı web servislerinden oluşmaktadır. Bu modül web servis otomasyonun ana omurgasını oluşturan ve detaylarına 2.1.1 alt başlığında değineceğimiz Tersine Mühendislik (RE), Kod Üretici (CG) ve Değişim Dinleyici (CL) alt modüllerinden oluşmaktadır.



***Şekil** Hata! Belgede belirtilen stilde metne rastlanmadı.**.1** APIGEN Genel Sistem Mimarisi*

APIGEN FE ve BE ana modülleri arasındaki haberleşme RESTful servisler aracılığı ile aşağıda belirtilen HTTP url ve metodlar kullanılarak gerçekleştirilir.

POST : /scanDatabase (Tersine mühendislik safhası)

GET : /getGeneratedEntities

POST : /generateAPIs (Servis kodu üretimi safhası)

2.1.1 Modüller, Bağımlılıklar ve İlişkiler

Aşağıda, bir önceki bölümde değinilen BE modüllerinin (Rev Engineering, Code Generation, Change Listener) detayları IEEE 1016 (Yazılım Tasarım Spesifikasyonu) komponent şablonu kullanılarak gösterilmektedir[13]. Kullanıcı arayüzünü oluşturan modüllerin detaylarına 4. Bölüm'de yer verilmektedir.

Tablo 2.1 Tersine Mühendislik Modülü

Tanımlayıcı	Tersine Mühendislik (Reverse Engineering : RE)
Tip	Modül
Amaç	Otomasyon sürecinin en başındaki modüldür. Veritabanında bulunan tüm tabloların uygulama katmanındaki veri modeli karşılıklarını oluşturmak, oluşturulan modellerin tip ve isim bilgilerini servis olarak sunmak bu modülün görevleridir.
Fonksiyon	Veritabanı tablolarının uygulama katmanındaki karşılıklarını üretip tablo, alan ve alan tipi gibi bilgileri servis olarak sunar.
Bağımlılıklar	RE modülü otomasyon sürecinin ilk aşamasıdır. Bu modül çalışmadan veri modeli üretilemeyeceğinden diğer modüllerin çalışması mümkün olmayacaktır. Örneğin servis kodunu üretecek "Code Generation" modülünün kod üretimi esnasında veri modeline ihtiyacı vardır. Aynı şekilde "Change Listener" modülünün değişimleri algılayabilmesi için eski ve yeni veri modeline ihtiyacı bulunmaktadır.
Arayüzler	Bu modül uygulama dışına RESTful arayüzü ile hizmet verdiği gibi. İçeriden kullanıma da açıktır. Aşağıdaki iki kullanım mevcuttur. 1. Önyüz uygulamadan veritabanı bağlantı bilgileri girildiğinde aşağıdaki HTTP metod ve url ile web servis çağrısı yapılarak tetiklenebilir. POST : /scanDatabase 2. Change Listener modülü periyodik olarak çalışarak RE modülünü metod çağrısı ile tetikler.
Algoritma	1. Kullanıcıdan alınan veritabanı bağlantı bilgileri ile şablon projede konfigürasyon dosyalarını oluştur. 2. Veritabanında bulunan tüm tabloları tara, veri modelini POJO kaynak kodu olarak üret. Üretilen kaynak kodu Entity olarak ifade edilir ve veritabanı tablosunun uygulama katmanındaki karşılığını ifade eder.

	3. Eğer verilen bilgiler ile yapılan ilk tarama işlemi ise bir sonraki adıma geçilir. Daha önce bir tarama yapıldıysa farkları tespit ederek kullanıcıya ilet. 4. Tüm modellerin isim, özellikler ve özelliklerin tip bilgileri ortak bir JSON veri yapısında APIGEN UI modülüne ilet.

Veri	**İstek(JSON):**
	{
	dataSourceUrl : “”,
	dataSourceUsername : “”,
	dataSourcePassword : “”,
	schemaName : “”
	}
	Yanıt(JSON):
	[
	{
	"entityName": "Country",
	"fields": {
	"id": "java.lang.Integer",
	"countryCode": "java.lang.String",
	"countryDescription": "java.lang.String"
	}
	},
	{
	"entityName": "City",
	"fields": {
	"id": "java.lang.Integer",
	"cityCode": "java.lang.String",
	"cityDescription": "java.lang.String",
	"countryCode": "java.lang.String"
	}
	},
	{
	"entityName": "Port",
	"fields": {
	"id": "java.lang.Long",
	"portCode": "java.lang.String",
	"portDescription": "java.lang.String",
	"cityCode": "java.lang.String",
	"countryCode": "java.lang.String"
	}
	}
	]

Tablo 2.2 Kod Üretici Modül

Tanımlayıcı	Kod Üreticisi (Code Generator : CG)
Tip	Modül
Amaç	Web Servisi kaynak kodunu kullanıcının belirlediği filtrelere göre üretmek. İlgili test sınıflarını(integration tests) üretmek ve çalıştırmak
Fonksiyon	Veritabanı tablolarının uygulama katmanındaki karşılıklarını üretip tablo, alan ve alan tipi gibi bilgileri servis olarak sunar.
Bağımlılıklar	RE modülü otomasyon sürecinin ilk aşamasıdır. Bu modül çalışmadan veri modeli üretilemeyeceğinden diğer modüllerin çalışması mümkün olmayacaktır. Örneğin servis kodunu üretecek CG modülünün kod üretimi esnasında veri modeline ihtiyacı vardır. Aynı şekilde "Change Listener" modülünün değişimleri algılayabilmesi için eski ve yeni veri modeline ihtiyacı bulunmaktadır.
Arayüzler	Bu modül uygulama dışına RESTful arayüzü ile hizmet verdiği gibi. İçeriden kullanıma da açıktır. Aşağıdaki iki kullanım mevcuttur. 1. Önyüz uygulamadan veritabanı bağlantı bilgileri girildiğinde aşağıdaki HTTP metod ve url ile web servis çağrısı yapılarak tetiklenebilir. POST : /scanDatabase 2. Change Listener modülü periyodik olarak çalışarak RE modülünü metod çağrısı ile tetikler.
Algoritma	1. Kullanıcıdan alınan servis üretim isteğini doğrula. 2. Servis üretim isteğinde iletilen her bir veri modeli için 3. Veritabanı ile etkileşime geçecek Repository sınıfını oluştur. 4. Servis üretim isteğinde yer alan filtreler için fonksiyonları oluştur.

	5. Kullanıcıya endpoint bilgilerini ya da üretilen proje zip dosyasını servis cevabı olarak ilet.

Veri	**İstek(JSON):**
	[
	{
	"entityName": "Member",
	"filters": {
	"and": ["lastName","firstName"],
	"or": ["emailAddress","lastName"],
	"equals": "firstName",
	"lessthan": "age",
	"between": "startDate"
	}
	},
	{
	Other Entities...
	}
	]
	Yanıt(JSON):
	{
	"status" : "SUCCESS"
	}

Tablo 2.3 Değişiklik Dinleyici Modül

Tanımlayıcı	Değişiklik Dinleyici(Change Listener)
Tip	Modül
Amaç	Veritabanında meydana gelen değişimleri tespit ederek otomasyon kullanıcılarını haberdar etmek. Bu sayede üretilen servislerin geçerliliğini korumak.
Fonksiyon	Veritabanını periyodik olarak kontrol ederek güncel ve eski veri modellerini karşılaştırır. Değişimleri tespit eder ve kullanıcıya bildirir.
Bağımlılıklar	Veri modelinin güncel halinin periyodik olarak alınabilmesi için RE modülü tetiklenir.
Arayüzler	Bu modülün herhangi bir arayüzü bulunmamaktadır. Periyodik olarak çalışan bir proses olarak gerçeklenmiştir.
Algoritma	1. Konfigürasyon dosyasında belirlenen sürelerde RE modülünü tetikle. 2. Üretilen yeni veri modellerini bir önceki veri modelleri ile karşılaştır ve farklılıkları tespit et. 3. Bir farklılık tespit edilirse kullanıcıya raporla. -
Veri	Bu modülün dışarıdan aldığı tek parametre çalışma periyodunu belirleyen interval parametresidir. Bu parametre konfigürasyon dosyasında tutulmaktadır.

2.2 Kısıtlar ve Zorluklar

Tez çalışması kapsamında karşılaşılan zorluklar ve çalışmanın sınırlı zamanda gerçekleştirilmesi sebebi ile belirlenen kısıtlar aşağıda listelenmektedir.

- Üretilen servislerin veritabanı sorgularından ibaret olması. Web Servisleri bundan çok daha fazlasını yapabilir. (kısıt)
- Üretilen veri modellerinin farklı veritabanlarını destekler nitelikte olması. Örneğin; Oracle veritabanında otomatik arttırımlı birincil anahtar oluşturulamıyor. Bunun yerine SEQUENCE kullanımı gerekiyor[14]. Bu zorluğu aşmak için veritabanına özel konfigürasyon yapmak gerekiyor. (zorluk)
- Tersine mühendislik yapılacak veritabanında karmaşık veritabanı tablo ilişkilerinin(one-to-many, many-to-one, many-to-many vb.) çok fazla olması. (zorluk)
- Veritabanında meydana gelen yapısal değişimlerin üretilen servislerin çalışabilirliğini olumsuz etkilemesi. (zorluk)

2.3 Hedefler

Tez çalışması kapsamında 2.2 başlığında belirtilen kısıtlar ve zorluklar doğrultusunda belirlenen hedefler aşağıda listelenmektedir.

- Basit CRUD sorgularının dışında daha karmaşık veritabanı sorgularını destekleyen web servislerinin otomatize bir şekilde üretilebilmesini sağlamak.
- Kullanıcı dostu ve modern bir arayüz yardımı ile üretilecek web servisleri ile ilgili filtrelerin ve özelliklerin ayarlanabilmesini sağlamak.
- Veritabanında meydana gelen yapısal değişimleri kullanıcıya bildirmek, bu sayede servislerin ve üretilen kodun doğruluğunu sağlayabilmek.

3
GERÇEKLEME

3.1 Tersine Mühendislik ile Veri Modelinin Oluşturulması

APIGEN Web Servis otomasyonunun ilk aşaması kullanıcıdan alınan veritabanı bağlantı bilgileri doğrultusunda veritabanından tablo verilerinin (meta data) çıkartılması ve POJO sınıflarının oluşturulmasıdır. Bu amaçla Hibernate çatısı altında bulunan bazı araçlardan faydalanılmıştır.

Hibernate JPA gerçeklemesi olarak geliştirilmiş açık kaynaklı bir ORM kütüphanesidir. Hibernate Java sınıfları ile veritabanı tablo değerleri (sütun değeri, birden fazla kayıt gibi) arasında ilişkisel eşleştirme yaparak Java programlama dilinde ilişkisel veritabanı ile etkileşimi oldukça kolay hale getirmektedir. Aşağıda Hibernate tarafından yapılmış eşleştirme ile Country isimli bir veritabanı tablosunun POJO karşılığı gösterilmektedir.

```java
/**
 * Country generated by hbm2java
 */
@Entity
@Table(name = "country", catalog = "myschema")
public class Country implements java.io.Serializable {

    private static final long serialVersionUID = 1L;
    private Integer id;
    private String countryCode;
    private String countryName;

    public Country(String countryCode, String countryName) {
        this.countryCode = countryCode;
        this.countryName = countryName;
    }

    @Id
    @GeneratedValue(strategy = IDENTITY)
    @Column(name = "id", unique = true, nullable = false)
    public Integer getId() {
        return this.id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    @Column(name = "country_code", length = 45)
    public String getCountryCode() {
        return this.countryCode;
    }

    public void setCountryCode(String countryCode) {
        this.countryCode = countryCode;
    }

    @Column(name = "country_name", length = 45)
    public String getCountryName() {
        return this.countryName;
    }

    public void setCountryName(String countryName) {
        this.countryName = countryName;
    }

}
```

Şekil 3.*Hata! Belgede belirtilen stilde metne rastlanmadı.**.1* Hibernate tarafından üretilen POJO sınıfı

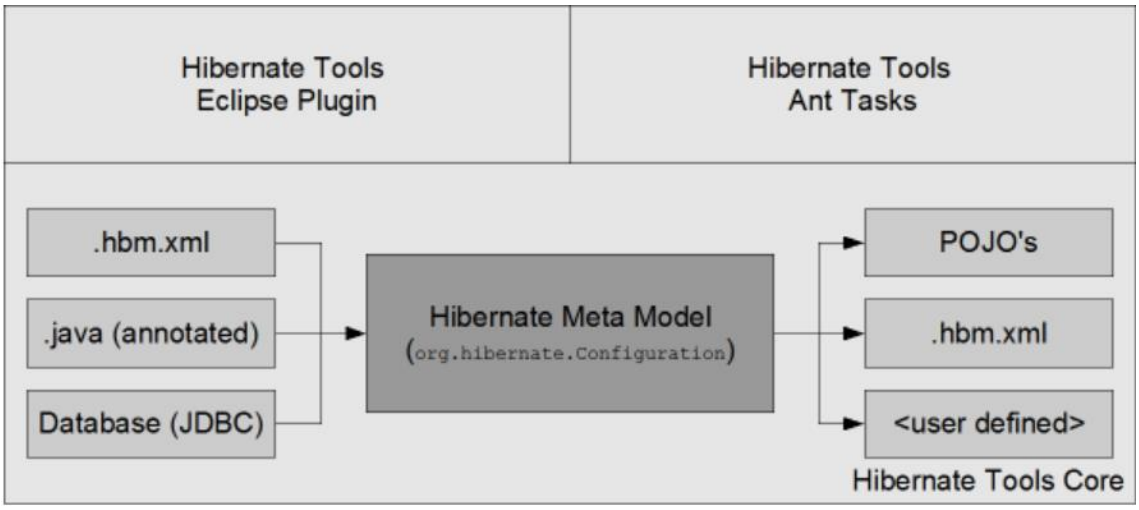
Hibernate yazılım geliştiricilere tersine mühendislik, kod üretimi, görselleştirme ve veritabanı ile etkileşim sağlayan çeşitli araçlar sunmaktadır. Ant görevleri ve Eclipse eklentileri bunlardan en önemlileridir[15]. Bölüm 3.1.1'de Hibernate araçlarını oluşturan temel bileşenlere değinilerek APIGEN ile web servis otomasyonunda kullanımı aktarılacaktır.

3.1.1 Temel Hibernate Bileşenleri

Hibernate kütüphanesinin nesne-ilişkisel eşleştirme yapabilmesi için kullanılan tanımlayıcı model Hibernate Meta Model olarak adlandırılmaktadır. Hibernate Meta Modeli, tablolar, sütunlar, sınıflar, özellikler, komponentler, değerler ve koleksiyonlar gibi bilgileri içermektedir. Bu model ile etkileşim sağlayan API org.hibernate.mapping paketi altında bulunan "Configuration" isimli sınıftır.

Bu sınıfı built etmenin birden fazla yolu bulunmaktadır.(Core Configuration, Annotation Configuration, JPA Configuration, JDBC Configuration) JDBC konfigürasyonu Hibernate tersine mühendislik aracını kullanır ve veri modeli eşleştirmesi yapabilmek için meta veri ve detaylarına Bölüm 3.1.3'te değineceğimiz tersine mühendislik konfigürasyon dosyasından (reveng.xml) faydalanır. Tez çalışması kapsamında bu konfigürasyon kullanılmıştır.

Eclipse eklentileri tersine mühendislik aşamasında bu konfigürasyonları otomatik olarak kullanırken Ant ile yapılan işlemlerde bu konfigürasyonlar Ant dosyalarında <jdbcconfiguration> altına yerleştirilir. Eclipse Eklentisi ve Ant Görevleri aynı altyapıyı kullanmaktadır.



Şekil 3.*Hata! Belgede belirtilen stilde metne rastlanmadı.**.1* Hibernate temel kavramlar

Eclipse eklentileri Eclipse aracı kullanılarak geliştirilen yazılımlarda geliştiricilere sunduğu çeşitli arayüzler vasıtasıyla veri modellerinin oluşturulmasını kolaylaştırmaktadır. Ant görevleri ise veri modellerinin projenin inşa sürecinde(build process) otomatize bir şekilde üretilmesini sağlayan konfigürasyon dosyalarından oluşmaktadır. Web Servis otomasyonunda süreç tamamen otomatize edileceği için herhangi bir geliştirme aracı veya bir IDE (örn : Eclipse) kullanılmamaktadır. Bu nedenle APIGEN aracında Eclipse eklentileri yerine Ant görevleri tercih edilmiştir.

Ant görevlerini kullanabilmek için hibernatetool görevi tanımlanmaktadır. Bu tanım BE modülü içerisinde apigenerator-template projesi altında .pom uzantılı maven dosyasında yer almaktadır. Bu görev altında aşağıdaki konfigürasyonlar belirtilmelidir.

```
<hibernatetool
  destdir="defaultDestinationDirectory"
  templatepath="defaultTemplatePath"
>
  <classpath ...>
  <property key="propertyName" value="value"/>
  <propertyset ...>
  (<configuration ...>|<annotationconfiguration ...>|
   <jpaconfiguration ...>|<jdbcconfiguration ...>)
  (<hbm2java>,<hbm2cfgxml>,<hbmtemplate>,...)
</hibernatetool>
```

Şekil 3.3 Hibernate Tersine Mühendislik temel konfigürasyonlar

Tablo 3.1 Hibernate Tool konfigürasyonu

Konfigürasyon adı	**Konfigürasyon tanımı**	**Konfigürasyon kullanımı**
destdir	Hibernate tarafından üretilecek dosyaların hedef dizini	Gerekli
templatepath	Kullanıcı-tanımlı şablonların yer aldığı dizin	Opsiyonel
classpath	Eşleştirme ve kullanıcı tipleri gibi kaynakları belirlemede kullanılan classpath	Opsiyonel fakat genellikle gerekli
property ve propertyset	Exporter'ları kontrol eden özellikleri set etmede kullanılır. Çoğunlukla kullanıcı tanımlı şablonlara has özellikler belirlemede kullanılır	Opsiyonel
configuration (annotationconfiguration, jpaconfiguration, jdbcconfiguration)	Hibernate Meta Modeli oluşturmanın dört farklı yolundan biri belirtilmeli. (APIGEN jdbcconfiguration kullanmaktadır.)	Zorunlu
hbm2java(hbm2cfgxml, hbmtemplate, vb.)	Bir veya birden fazla exporter belirtilmeli.	Zorunlu

3.1.2 JDBC Konfigürasyonu

Bir <jdbcconfiguration> etiketi JDBC bağlantısından tersine mühendislik yapmada kullanılmaktadır. Bu konfigürasyon bağlantı bilgilerini hibernate.cfg.xml veya hibernate.properties dosyalarından okur. APIGEN BE modülü içerisinde yer alan RE modülü hibernate.properties dosyasını /scanDataSource çağrısında gelen veritabanı bağlantı bilgilerine göre otomatik olarak oluşturmaktadır. Şekil 3.4'te gösterilen <jdbcconfiguration> etiketi tersine mühendislik sürecinin nasıl gerçekleştirileceğine ilişkin parametreleri içermektedir. Parametreler ve anlamları Tablo 3.2'de açıklanmıştır.

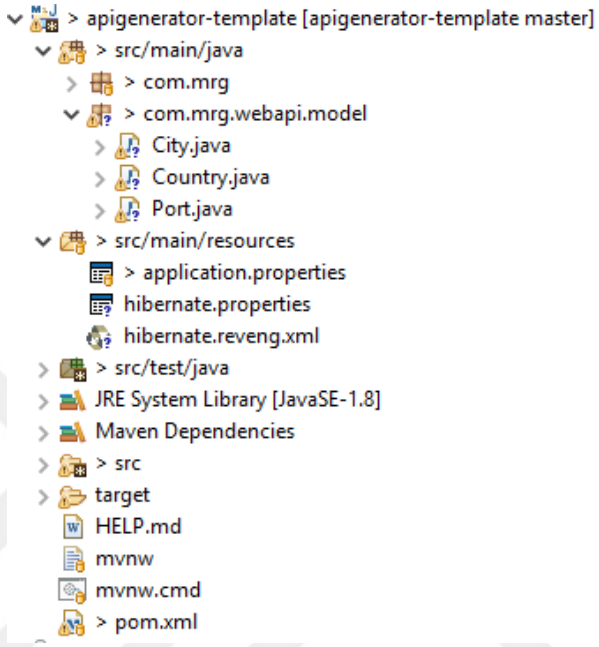
```
<jdbcconfiguration
...
packagename="package.name"
propertyfile=”hibernate.properties”
revengfile="hibernate.reveng.xml"
reversestrategy="ReverseEngineeringStrategy"
detectmanytomany="true|false"
detectoptmisticlock="true|false"
>
...
</jdbcconfiguration>
```

Şekil 3.4 JDBC konfigürasyonu

Tablo 3.2 Tersine Mühendislik için jdbc konfigürasyonu

Parametre adı	Tanım	Parametre kullanımı
packagename	Sınıflar için eşleştirmeler oluşturulduğunda kullanılacak varsayılan paket adı.	Opsiyonel
propertyfile	Veritabanı bağlantı bilgilerini içeren konfigürasyon dosyası	Zorunlu
revengfile	Tersine mühendislik sürecini kontrol eden .xml uzantılı konfigürasyon dosyası.	Opsiyonel
reversestrategy	org.hibernate.cfg.reveng.ReverseEngineeringStrategy. Sınıfını gerçekleyen somut strateji sınıfı. Hibernate aracının Tersine mühendisliği kontrol ederken kullanacağı stratejiyi ayarlamada kullanılır. Örn. Alanların isimlendirmesi, hangi tabloların dahil olacağı veya olmayacağı gibi. Reveng.xml dosyası yerine(veya bu dosya ile) bu sınıfı kullanmak tersine mühendislik sürecinde tam programlama kontrolü sağlar.	Opsiyonel
detectManytoMany	Eğer bu özellik true ise, tamamen many-to-many ilişkiye sahip bağlantılı tablolar eşlenir. Böyle bir tablo, birincil anahtarın, diğer tablolarına işaret eden iki yabancı anahtarı içeren ve başka sütun içermeyen bir tablodur.	Varsayılan : true
detectOptimisticLock	Eğer bu özellik true ise, VERSION veya TIMESTAMP ismine sahip sütunlar ilgili <version> veya <timestamp> etiketine göre iyimser kilit(optimistic locking) ile eşleştirilir.	Varsayılan : true

APIGEN bu konfigürasyonu maven eklentisi olarak kullanır. Kullanıcıdan alınan veritabanı bağlantı bilgileri ile maven eklentisi çalışma zamanında tetiklenerek veri modelinin şablon proje içerisinde oluşturulması sağlanır. Şekil 3.5 çalıştırılan bir jdbc konfigürasyonu sonrası şablon projede üretilen veri modelini göstermektedir.



Şekil 3.5 Şablon proje Country, City ve Port veri modellleri

3.1.3 Tersine mühendisliği kontrol etmek

Bir önceki başlıkta belirtilen <jdbcconfiguration> etiketi kullanıldığında **Tablo 3.2**'de belirtilen **reveng** parametresi olarak **hibernate.reveng.xml** isimli bir konfigürasyon dosyası verilir. Tersine mühendisliğin kontrol edilebilmesi için bazı stratejiler içeren bu dosya el ile oluşturulabileceği gibi Eclipse Hibernate eklentileri ile de oluşturulabilmektedir. Bu dosyada tip eşleştirmeleri, şema ve tablo filtreleri belirtilebilir. Aşağıda **hibernate.reveng.xml** dosyasında belirlenebilecek temel filtreler açıklanmıştır.

<schema-selection> Etiketi

<schema-selection> etiketi tersine mühendisliğin hangi şema üzerinde denenmesi gerektiğini belirtmede kullanılır. Varsayılan olarak veritabanındaki tüm şemaları okunur ve <table-filter> etiketi ile hangi tabloda tersine mühendislik yapılıp yapılmayacağına karar verilir. Bu başlangıç için daha kolaydır fakat çok fazla şemanın olduğu veritabanlarında verimli olmayabilir. Bu sebeple <schema-selection> etiketi ile şemalar limitlendirilmelidir.

Örnek kullanım :

<schema-selection match-schema=”MY_SCHEMA” />

<schema-selection match-schema=”COMMON_SCHEMA” match-table=”CITY” />

<type-mapping> Etiketi

<type-mapping> alanı veritabanında bulunan JDBC tipleri ile Hibernate tiplerinin nasıl eşleştirileceğini belirtmede kullanılır. Örnek kullanım aşağıdaki gibidir.

```
<type-mapping>
  <sql-type jdbc-type="VARCHAR" length="1" hibernate-type="char"/>
  <sql-type jdbc-type="VARCHAR" hibernate-type="string"/>
</type-mapping>
```

<table-filter> Etiketi

Bu etiket genel filtreleme ve tablo ayarları uygulayabilmek için eşleştirme kuralları belirtmeyi sağlamaktadır. Örneğin şemaya göre bir belirli bir tabloyu tersine mühendislik sürecine dahil edip etmeyeceğimizi bu etiket ile belirleyebiliriz. Örnek kullanım aşağıdaki gibidir.

```
<table-filter
 match-catalog="catalog_matching_rule"
 match-schema="schema_matching_rule"
 match-name="table_matching_rule"
 exclude="true|false"
 package="package.name"
/>
```

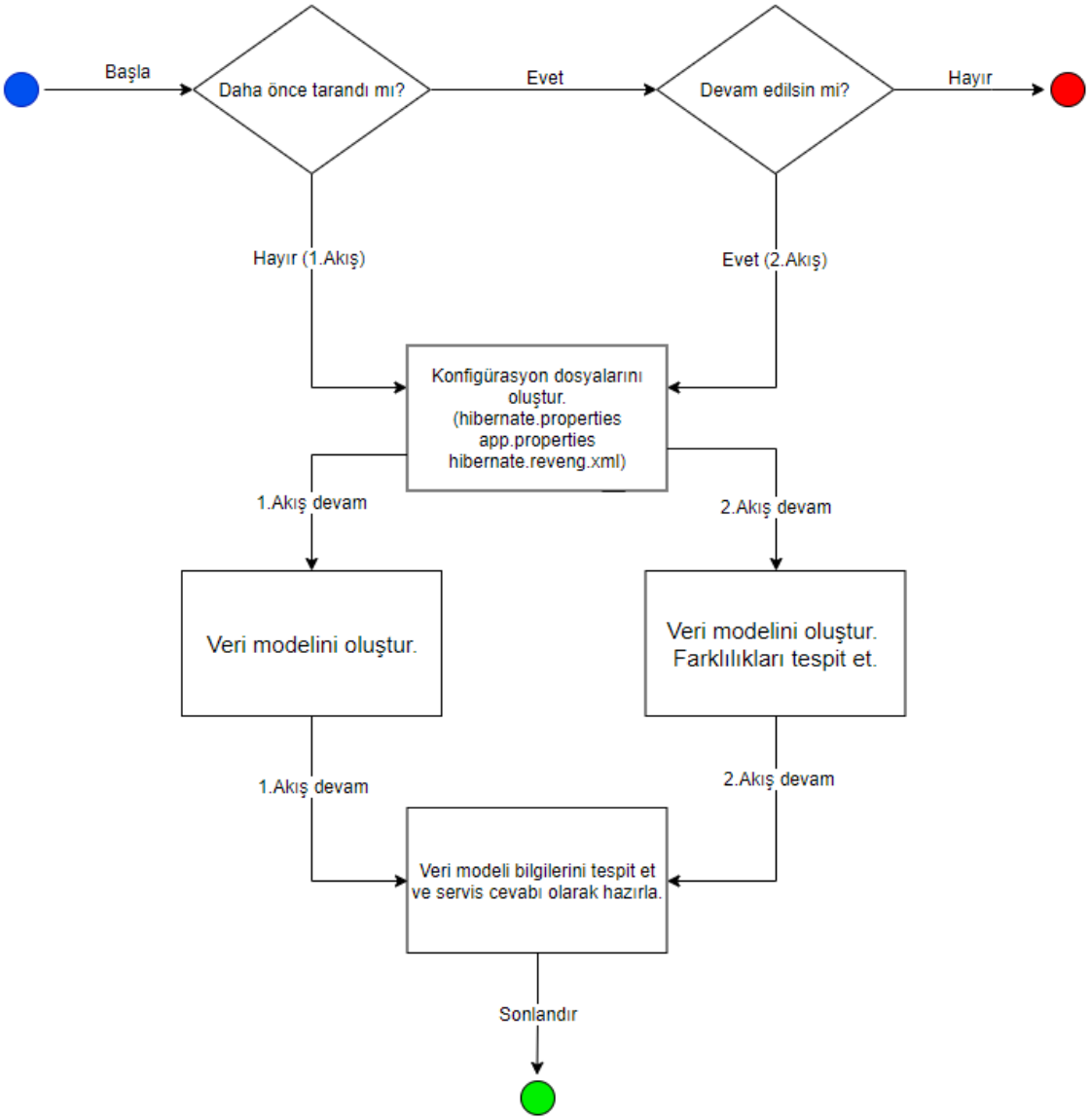
<table> Etiketi

<table> etiketi tablo özelinde tersine mühendisliğin nasıl yapılması gerektiğini belirtmede kullanılır. Bu etiket bir tablo için POJO sınıf isimlendirmesini belirlemede, birincil anahtar için hangi tanımlayıcı üreticinin kullanılması gerektiğini belirlemede kullanılabilir. Ayrıca sütun ve yabancı anahtarlar ile ilgili özellikler de belirtilebilmektedir. Örnek kullanım aşağıdaki gibidir.

```
<table-filter
 match-catalog="catalog_matching_rule"
 match-schema="schema_matching_rule"
 match-name="table_matching_rule"
 exclude="true|false"
 package="package.name"
/>
```

3.1.4 APIGEN Konfigürasyonu

Bu bölümde tez çalışması kapsamında Bölüm 3.1.2 ve 3.1.3'te belirtilen konfigürasyonların APIGEN RE modülünde nasıl kullanıldığı aktarılmaktadır. Şekil 3.6'da verilen akış şeması tersine mühendislik sürecinin işleyişini göstermektedir.



Şekil 3.6 APIGEN Tersine Mühendislik Akış Şeması

APIGEN web servis otomasyonunun ilk aşamasında kullanıcıdan alınan veritabanı bağlantı bilgileri, şema ve tablo filtreleri ile Bölüm 3.1.2'de belirtilen JDBC konfigürasyonu oluşturulmaktadır. Konfigürasyon APIGEN şablon projesinde maven eklentisi içerisinde aşağıdaki gibi oluşturulmuştur.

```
<jdbcconfiguration
        revengfile="src/main/resources/hibernate.reveng.xml"
        packagename="com.apigenerator.template.model"
        detectmanytomany="true"
        propertyfile="src/main/resources/hibernate.properties" />
```

Şekil 3.7 JDBC Konfigürasyonu

JDBC konfigürasyonunda bağlantı bilgilerini içeren hibernate.properties dosyası ve tersine mühendislik stratejilerinin bulunduğu hibernate.reveng.xml dosyası kullanıcının sağladığı bilgiler doğrultusunda APIGEN RE modülü tarafından dinamik olarak oluşturulur. Bu dosyaların proje içerisindeki konumu Şekil 3.6'da görülebilir. Örneğin MYSQL veritabanı için hibernate.properties dosyası şablon proje içerisinde aşağıdaki gibi oluşturulur.

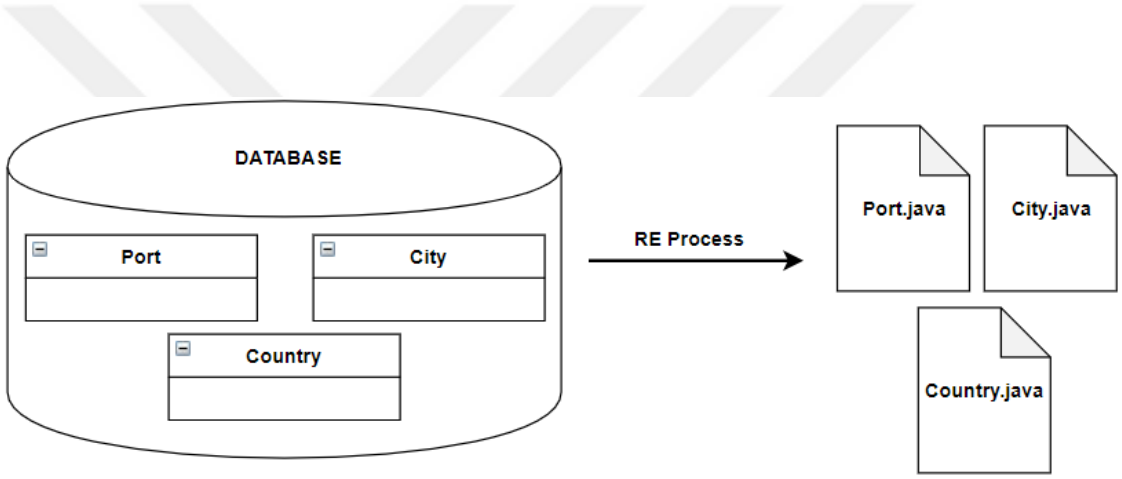
```
hibernate.connection.driver_class=com.mysql.jdbc.Driver
hibernate.connection.url=jdbc:mysql://localhost:3306/myschema
hibernate.connection.username=dbusername
hibernate.connection.password=dbpassword
hibernate.dialect=org.hibernate.dialect.MySQLDialect
```

Benzer şekilde kullanıcıdan alınan bilgiler ile şablon proje içerisinde hibernate.reveng.xml dosyası varsayılan olarak aşağıdaki gibi oluşur. Kullanıcının herhangi bir tablo belirtmemesi durumunda şemadaki tüm tablolar taranmaktadır.

```
<hibernate-reverse-engineering>
      <schema-selection match-schema="userschema" match-table=".*"/>
</hibernate-reverse-engineering>
```

Şekil 3.8 Tersine Mühendislik Konfigürasyonu

Bu iki dosyanın oluşturulması ile şablon proje(apigenerator-template) veri modellerini almaya hazır hale gelir. Bu aşamadan sonra RE modülü maven eklentisi olarak önceden oluşturulan JDBC konfigürasyonunu tetikler ve Hibernate bu konfigürasyona göre veri modellerini oluşturur.



Şekil 3.9 Tersine Mühendislik ile veri modellerinin oluşturulması

Veri modelleri oluşturulduktan sonraki aşamada sonra APIGEN UI modülüne veri modellerinin isim ve sahip olduğu alanların bilgileri tespit edilerek APIGEN UI modülüne JSON formatında cevap olarak iletilir. (Bkz. Şekil 3.7) Bu sayede kullanıcı üretilen veri modelleri için çeşitli sorgu filtreleri düzenleyebileceği ve bu filtreler ile web servislerini otomatize bir şekilde oluşturabileceği APIGEN UI modülünün ilgili ekranına yönlendirilir.

```
[
   {
       "entityName": "Country",
       "fields": {
           "id": "java.lang.Integer",
           "countryCode": "java.lang.String",
           "countryDescription": "java.lang.String"
       }
   },
   {
       "entityName": "City",
       "fields": {
           "id": "java.lang.Integer",
           "cityCode": "java.lang.String",
           "cityDescription": "java.lang.String",
           "countryCode": "java.lang.String"
       }
   },
   {
       "entityName": "Port",
       "fields": {
           "id": "java.lang.Long",
           "portCode": "java.lang.String",
           "portDescription": "java.lang.String",
           "cityCode": "java.lang.String",
           "countryCode": "java.lang.String"
       }
   }

]
```

Şekil 3.10 JSON formatında veri modeli bilgileri

Bir sonraki başlıkta tersine mühendislik ile oluşturulan veri modellerinden web servislerinin nasıl üretildiği gösterilmektedir.

3.2 Web Servis Kaynak Kodu Üretimi

Bu bölümün alt başlıklarında APIGEN tarafından kod üretim işlemlerinin nasıl yapıldığı aktarılmaktadır.

3.2.1 JavaPoet Kütüphanesi Kullanımı

APIGEN veri modelinden web servis kodu üretirken JavaPoet kütüphanesinden faydalanmıştır. JavaPoet Java kaynak kodu üretmek için uygulama programlama arayüzleri (API) sunan bir kütüphanedir[16]. Bu arayüzler kullanılarak ilkel tipler, sınıflar, fonksiyonlar, parametreler, anotasyonlar vb. oluşturulabilmektedir. JavaPoet üretilen sınıfın bağımlı olduğu sınıfları da otomatik olarak üretilen sınıfa dahil edebilmektedir (import).

JavaPoet sınıf ve arayüzler için TypeSpec, sınıf alanları için FieldSpec, metod ve kurucu fonksiyonlar için MethodSpec, parametre tanımları için ParameterSpec ve anotasyonlar için AnnotationSpec modellerine sahiptir. JavaPoet bu modelleri metod zincirleme yapısı ile kullanmaya olanak sağlar. Şekil 3.11'de tez çalışmasında metod zincirleme yapısı kullanımı ve örnek sonucu üretilmiş kaynak kodu (CountryRepository) gösterilmiştir.

```
TypeSpec entityRepository = TypeSpec.interfaceBuilder("CountryRepository")
        .addAnnotation(RepositoryRestResource.class)
        .addModifiers(Modifier.PUBLIC)
        .addSuperinterface(ParameterizedTypeName
        .get(ClassName.get(PagingAndSortingRepository.class),
         ClassName.get("com.apigenerator.model", "Country"),
         ClassName.get(Long.class)))
        .build();
```

```
@RepositoryRestResource
public interface CountryRepository extends PagingAndSortingRepository<Country,
Long> {

}
```

Şekil 3.11 JavaPoet metod zincirleme ile kod üretimi

3.2.2 Veri Modelinden Web Servisleri Üretmek

APIGEN web servis kodunu üretirken veri modeli ve kullanıcının bu veri modeline göre uygulama arayüzünden belirlediği sorgu filtreleri baz alınmaktadır. Kullanıcı servis üretimi sürecini başlattığında önyüzden oluşturulan JSON tipinde veri yapısı yorumlanmak üzere APIGEN CG modülüne gönderilir. Veri modellerini ve modellere ait filtreleri içeren JSON veri yapısı Şekil 3.12'de gösterilmiştir.

```
[
  {
    "entityName": "Member",
     "filters": {
        "and": ["lastName","firstName"],
        "or": ["emailAddress","lastName"],
        "equals": "firstName",
        "lessthan": "age",
        "between": "startDate"
     }
  },
  {
        Other Entities...
  }
]
```

Şekil 3.12 JSON CG servis üretim isteği

APIGEN CG modülü Şekil 3.12'deki gibi aldığı servis üretim taleplerini yorumlar ve Şekil 3.13'dakine benzer bir servis kodu üretir.

```
@RepositoryRestResource
public interface MemberRepository extends PagingAndSortingRepository<Member, Long>{
  List<Member>findByLastnameAndFirstname(String lastName,String firstName);
  List<Member>findByEmailAddressOrLastName(String email,String lastName);
  List<Member>findByFirstName(String firstName);
  List<Member>findByAgeLessThan(int age);
  List<Member>findByStartDateBetween(Date startDate, Date endDate);

}
```

Şekil 3.13 APIGEN tarafından üretilen örnek Java kodu

Şekil 3.13'te gösterilen servis katmanı kodu Spring Data REST altyapısını kullanmaktadır. Spring Data REST projesi Spring projesinin bir alt projesidir[17]. Bu proje sayesinde hypermedia tabanlı REST web servisleri üretilebilmektedir. Ayrıca Spring Data REST'in sunduğu sorgu konfigürasyonu mekanizması sayesinde Şekil 3.13'te üretilen fonksiyonlar aynı zamanda ilgili veri modelleri için arka planda veritabanı sorgularının oluşturulmasını sağlar.

Tablo 3.3'de Spring Data Rest tarafından desteklenen ve APIGEN tarafından otomatize bir şekilde üretilebilen bazı metod örnekleri gösterilmiştir.

Tablo 3.3 Spring Data Rest metodları

Anahtar Kelime	Örnek	Sorgu
And	findByLastnameAndFirstname	... where x.lastname = ?1 and x.firstname = ?2
Or	findByLastnameOrFirstname	... where x.lastname = ?1 or x.firstname = ?2
Is, Equals	findByFirstname, findByFirstnameIs, findBYFirstnameEquals	... where x.firstname = ?1
Between	findByStartDateBetween	... where x.startDate between ?1 and ?2
LessThan	findByAgeLessThan	... where x.age < ?1
LessThanEqual	findByAgeLessThanEqual	... where x.age <= ?1
GreaterThan	findByAgeGreaterThan	... where x.age >?1
GreaterThanEqual	findByAgeGreaterThanEqual	... where x.age >= ?1

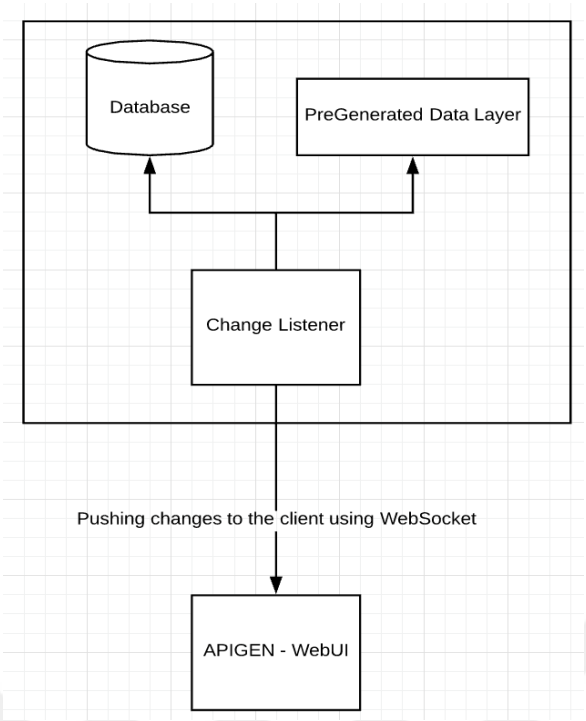
Bu mekanizma find…By, read…By, query…By, count…By ve get…By gibi ön ekleri metoddan çıkarır ve ismin geri kalanını çözümler. Oluşturulan metod isminde "By" anahtar kelimesi sorgu kriterinin başlangıcını gösteren bir sınırlandırıcı olarak işlev görür. Bu sayede veritabanı modeli özellikleriyle ilgili koşullar tanımlanabilir ve bunlar Ve ve Veya ile birleştirilebilir[18].

3.3 Değişime Duyarlı Otomasyon Yaklaşımı

Veritabanında meydana gelen yapısal bir değişiklik otomatize bir şekilde üretilen web servislerini geçersiz kılabilmektedir. Örneğin bir tablonun veya sütunun silinmesi, adının değişmesi APIGEN tarafından üretilen web servislerini geçersiz kılabilecektir. Tez çalışması kapsamında bu problemin önüne geçebilmek için veritabanındaki değişimlere duyarlı bir yapı gerçeklenmiştir. Bu sayede APIGEN tarafından otomatize bir şekilde üretilen web servislerinin geçerliliğinin korunması ve üretilen kaynak kodun doğru şekilde tekrar üretilebilmesi hedeflenmiştir.

3.3.1 Web Soketleri

Veritabanındaki değişimleri periyodik olarak kontrol eden modülün yapısal değişimleri APIGEN UI kullanıcılarına raporlayabilmesi HTTP protokolü ile mümkün olmamaktadır. Çünkü HTTP protokolü istemciden sunucuya yarı çift yönlü haberleşmeyi desteklediğinden sunucudan istemciye veritabanındaki değişimler ile alakalı bildirimde bulunmak mümkün olmamaktadır [19]. Bu nedenle APIGEN CL modülü gerçeklenirken tam çift yönlü haberleşmeyi destekleyen Web Socket protokolü kullanılmaktadır[20].



Şekil 3.14 APIGEN Web Soketi Kullanımı

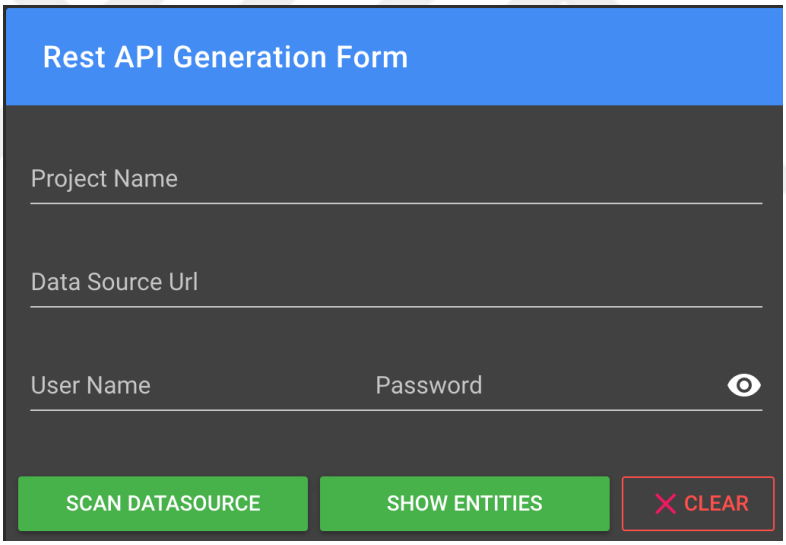
APIGEN CL modülü periyodik olarak çalışan bir arkaplan görevi(background process) olarak geliştirilmiştir. Bu modülün çalışma prensibi aşağıdaki gibidir.

1. Kullanıcının belirlediği periyodlarda TM modülünü tetikleyerek veritabanını tara.
2. Tarama sonucu bulunan veri modelini bir önceki veri modeli ile karşılaştır.
3. Karşılaştırma sonucu yapısal değişiklik olarak tespit edilen ve web servisinin geçerliliğini bozabilecek değişimleri kullanıcıya raporla.

4
KULLANICI ARAYÜZÜ

APIGEN UI modülü çalışmamız kapsamında geliştirdiğimiz web servis otomasyon aracının web arayüzünü ifade eder. APIGEN web arayüzü JavaScript/VueJS tabanlı bir SPA uygulamasıdır. Web arayüzünün SPA mimarisiyle geliştirilmesinin nedeni kullanıcılara daha interaktif bir uygulama deneyimi sunabilmektir. [21]

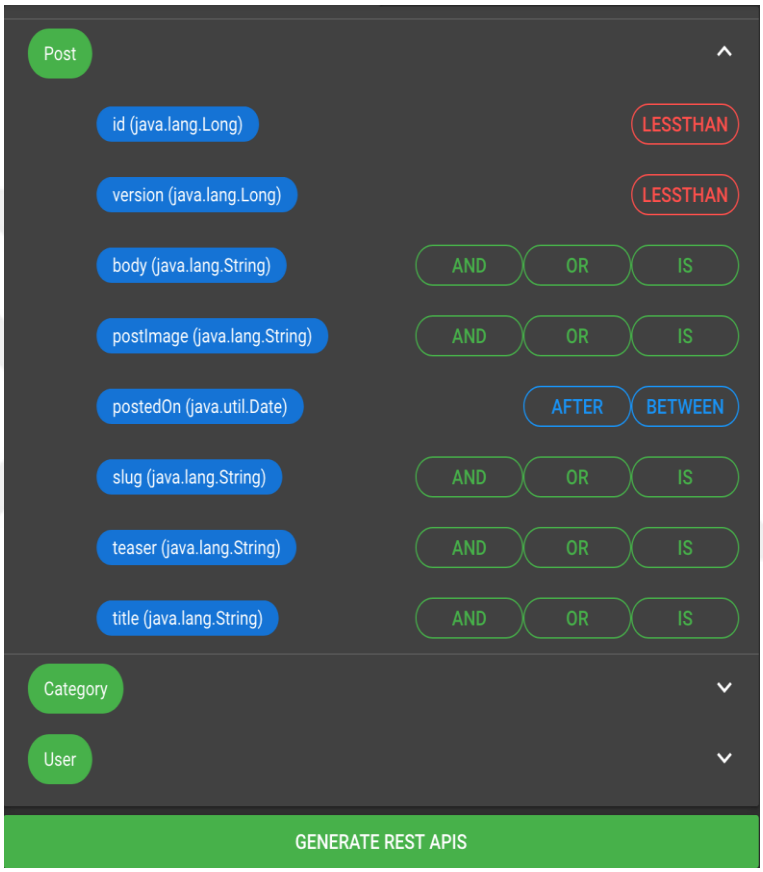
Web arayüzüne giriş yapan bir kullanıcı web servis otomasyonunu başlatmak üzere veritabanı bağlantı bilgilerini girebileceği bir form ile karşılaşmaktadır. Bu formda girilmesi gereken bilgiler Şekil 4.1'de belirtilmektedir.



Şekil 4.1 APIGEN veritabanı bilgilerinin sağlandığı form

Project Name(Proje Adı) farklı web servis otomasyon projelerini ayırt etmede yardımcı bir parametredir. Bu parametre kullanılarak önceden oluşturulmuş projeler ile ilgili bilgiler görüntülenebilecektir. Data Source Url(Veri Kaynağı Url Bilgisi) olarak belirtilen alan ise tersine mühendislik işleminin yapılacağı veritabanının adres bilgisini ifade etmektedir. Username ve Password alanları da bu

veritabanın da işlem yapmaya yetkili kullanıcı bilgilerini içermelidir. Formda gerekli tüm alanları dolduran kullanıcı SCAN DATASOURCE butonuna tıkladığında RE modülünü tetiklemiş olur. Bu sayede 3.1 başlığında belirtilen işlemler başlar. RE modülü veritabanı tarama işini bitirdiğinde web arayüzüne Şekil 3.9'daki bilgiler iletilir. Bu bilgiler sayesinde kullanıcıların web arayüzünde çeşitli filtreler ile web servis üretim isteğinde bulunabileceği Şekil 4.2'de gösterilen arayüz oluşturulur.



Şekil 4.2 Veri modeli bilgileri ve oluşan filtreler

Tarama işleminin sonunda web arayüzüne gelen veri modeli bilgileri, kullanıcının hangi veritabanı sorgu operasyonları için web servislerini oluşturabileceğini belirlemektedir. Örneğin; String tipinde bir veri için LESSTHAN filtresinin kullanımı mümkün olmayacağı gibi Long tipinde bir veri için de AFTER ve BETWEEN filtreleri kullanımı mümkün olmayacaktır. Kullanıcılar bu bilgiler doğrultusunda diledikleri tablo için basit CRUD işlemlerinin dışında ek filtreler belirleyebilmektedir. Ayrıca

sürecin dışında tutulmak istenen tablolar seçilebilmektedir. Hiçbir filtrenin seçilmemesi durumunda varsayılan ayar geçerlidir. Varsayılan ayarla süreç başlatıldığında tüm tablolar için temel CRUD operasyonlarını içeren RESTful servisler proje kodu içerisinde oluşturulmaktadır. Süreç, seçilen filtreler ile başlatıldığında ise bu filtrelere karşılık gelen RESTful servisler proje kodu içerisinde oluşturulmaktadır. Sürecin sonunda dağıtıma hazır bir web servis uygulaması üretilmiş olur.

5
SONUÇ VE ÖNERİLER

Tez çalışması kapsamında geliştirilen APIGEN aracının testleri MySQL ve Oracle olmak üzere iki farklı veritabanı sisteminde gerçekleştirilmiştir. İlk testler Hibernate aracının varsayılan tersine mühendislik konfigürasyonu ile hiçbir ekstra filtre ve konfigürasyon belirtmeden yapılmıştır. Bu testlerde farklı veritabanı tedarikçilerinin benzer yapılar için farklı gerçeklemelerinin bulunmasından ötürü APIGEN RE ve CG modüllerinde farklı konfigürasyonlara ihtiyaç duyulduğu ortaya çıkmıştır. Örneğin MySQL veritabanı sisteminde otomatik artış özelliğine sahip birincil anahtar tanımlanabiliyorken Oracle veritabanlarında versiyon 12c'ye kadar bu mümkün değildi. [22] Bu nedenle versiyon 12c öncesinde otomatik artış özelliği veritabanı sekansları ile sağlanmaktadır. Ayrıca Oracle veritabanında birincil anahtar veri tipi varsayılan olarak NUMBER tipinde iken MySQL veritabanında birincil anahtar INT tipinde olabilmektedir. Üretilecek web servis kodunda bu tiplerin JDBC tipi karşılığı yer alacağı için veri modeli üretimi esnasında NUMBER ve INT tiplerinin aynı Hibernate veri tipine dönüştürülmesi gerekmektedir.

Bu ve benzeri farklılıklardan ötürü APIGEN RE modülünün veri modelini üretirken veritabanı sistemine göre farklı tersine mühendislik konfigürasyonlarını dinamik bir şekilde oluşturmasını gerektirmektedir. Bölüm 3.1'de Hibernate tersine mühendislik aracı kullanılarak veri modelinin hangi konfigürasyonlar ile üretilebileceği gösterilmektedir.

EdmGen++ olarak isimlendirilen bir tersine mühendislik çalışmasında Hibernate gibi ORM kütüphaneleri tarafından üretilen veritabanı ilişkilerinin karmaşık ilişkileri kapsayamadığı dile getirilmiştir. [23] Bu çalışmada geliştirilen bir araç sayesinde karmaşık ilişkileri de hesaba katabilen bir varlık veri modeli(Entity Data Model) geliştirilmiştir. Biz kendi çalışmamızda çok karmaşık veritabanı ilişkilerini ele almadığımız için Hibernate kütüphanesi ile kendi testlerimizde başarılı sonuçlar aldık. RESTful servislerin otomasyonunu ele alan diğer birçok çalışmada model

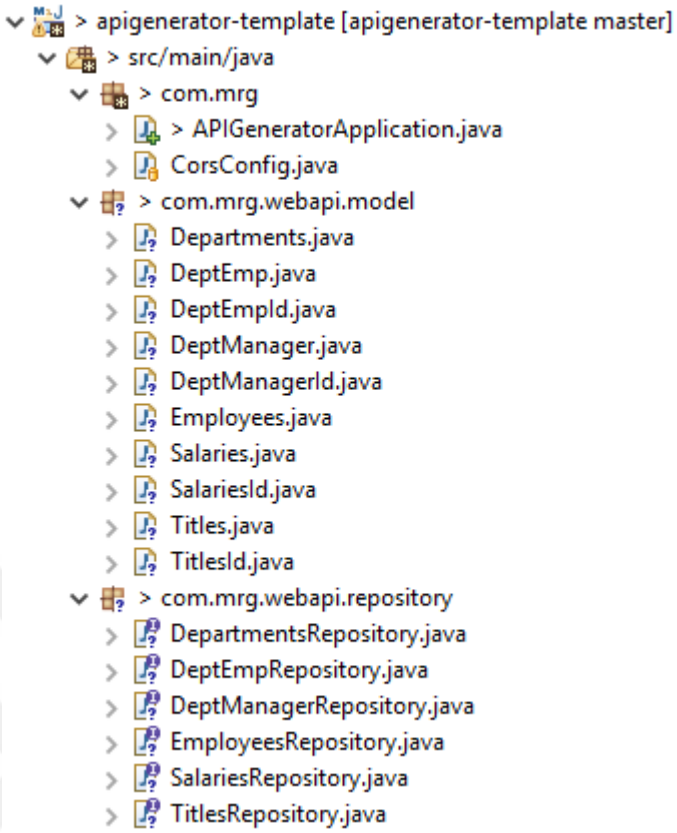
öncelikli(model-first) yaklaşım kullanılmaktadır. [24] Bu nedenle bu yaklaşımlar ile varolan tabloları web servislerine dönüştürmek mümkün olmamaktadır. APIGEN aracı ile yapılan testlerde küçük ve orta ölçekli bir projenin veritabanı RESTful servislere dönüştürülebilmiştir. Yaptığımız testler sonucunda APIGEN özellikleri ile benzer amaca hizmet eden diğer ürünleri Tablo 6.1'de karşılaştırdık.

Tablo 5.1 Karşılaştırma Tablosu

Sütun : Araç **Satır : Özellik**	Spring Roo	Telosys	DreamFactory	APIGEN
Reverse Engineering	+	+	+	+
Customization of Services	Using Command Line	-	Using Custom Scripts	Using User Interface
Ready to Deploy Application	+	+	On Premise	+
Programming Language	Java	Java	PHP	Java
User Interface Generation	+	+	-	-
Change Awareness	- (Incremental Reverse Engineering)	-	-	+

5.1 APIGEN test sonuçları

APIGEN testleri Oracle ve MySQL veritabanlarında gerçekleştirilmiştir. Testler için Employees veritabanından faydalanılmıştır. Bu veritabanı 6 adet tablodan oluşmaktadır. APIGEN aracı kullanılarak Employees veritabanında bulunan tabloların web servislerine dönüştürülmesi test edilmiştir.



Şekil 5.1 Employees veritabanı test sonuçları

Şekil 5.1 Employees veritabanında bulunan DEPARTMENTS, DEPTEMP, DEPTMANAGER, EMPLOYEES, SALARIES, TITLES tabloları için tersine mühendislik ve kod üretim aşamaları sonunda APIGEN tarafından üretilen projeyi ve .java kaynak kodu dosyalarını göstermektedir. Tersine mühendislik modülünün çalışması sonucu com.mrg.webapi.model paketi altındaki dosyalar, kod üretici modülün web servislerini üretmek üzere çalışması sonucu com.mrg.webapi.repository paketi altındaki .java kaynak kodu dosyaları otomatize bir şekilde söz dizimi ve derleme hataları bulunmadan üretilebilmiştir. Üretilen uygulama proje dosyası başarılı şekilde çalıştırılabilmiş, Şekil 5.2'de belirtilen web servisleri kullanılabilir duruma getirilmiştir.

```
{
  "_links" : {
    "titleses" : {
      "href" : "http://localhost:8080/titleses{?page,size,sort}",
      "templated" : true
    },
    "employeeses" : {
      "href" : "http://localhost:8080/employeeses{?page,size,sort}",
      "templated" : true
    },
    "salarieses" : {
      "href" : "http://localhost:8080/salarieses{?page,size,sort}",
      "templated" : true
    },
    "departmentses" : {
      "href" : "http://localhost:8080/departmentses{?page,size,sort}",
      "templated" : true
    },
    "deptEmps" : {
      "href" : "http://localhost:8080/deptEmps{?page,size,sort}",
      "templated" : true
    },
    "deptManagers" : {
      "href" : "http://localhost:8080/deptManagers{?page,size,sort}",
      "templated" : true
    },
    "profile" : {
      "href" : "http://localhost:8080/profile"
    }
  }
}
```


Şekil 5.2 APIGEN testlerinde üretilen web servisleri

Sonuç olarak bu çalışma kapsamında veritabanlarında bulunan tablolar için veritabanı operasyonlarının yapılmasını sağlayan web servisleri otomatize bir şekilde üretilebilmiştir diyebiliriz. Geliştirdiğimiz APIGEN isimli aracın sunduğu web arayüzü sayesinde kullanıcılar veritabanlarında bulunan tabloları kullanarak çeşitli filtreler ile web servislerini Spring tabanlı bir proje olarak üretebilmişlerdir.

Bölüm 5'te elde edilen test sonuçları MySQL ve Oracle veritabanlarında yapılan testlerden elde edilmiştir. Sistemimizin MySQL ve Oracle dışındaki çeşitli veritabanı sistemleri için web servislerini üretmede ne kadar verimli olduğu gelecek testlerde görülebilecektir. Bölüm 5'te değinilen testler sonucunda eski bir veritabanı şeması için tabloların web servisleri oluşturulmuş, veri modellerinde yapılan küçük düzenlemeler sonrasında APIGEN'in ürettiği servis projesi çalışır hale getirilmiştir. Ayrıca gerçek bir veritabanında da testler yapılmıştır. İki geliştirici tarafından 10

adam günde geliştirilen bir admin projesinin web servislerinin büyük bir kısmı APIGEN kullanılarak çok kısa bir sürede çalıştırılır hale getirilmiştir.

Çalışmamız kapsamında üretilecek web servislerinde çeşitli özelleştirmelere olanak tanısak da üretilen servislerin sahip olduğu özellikler veritabanı sorguları ile sınırlı kalmaktadır. Web servisleri genellikle bundan daha fazlasını yapmaktadırlar. Çoğu zaman karmaşık iş mantıkları ve çeşitli hesaplamalar web servisleri aracılığı ile sunulmaktadır. Gelecek çalışmalarda web servis otomasyonunda daha kapsamlı gereksinimlerin ele alınarak, kullanıcıya sunulan özelleştirmelerin veritabanı sorgularının ötesine geçmesi faydalı olacaktır.

Ek olarak otomasyon sonucu üretilen web servislerin birim ve entegrasyon testleri bulunmadığından doğruluk ve çalışabilirlik, yapılan manuel testler aracılığı ile sağlanmıştır. İleride APIGEN için birim ve entegrasyon testlerinin otomatize bir şekilde üretilmesi planlanmaktadır.

Son olarak, APIGEN web servis otomasyon sürecinde deploy edilebilir bir Spring projesi üretmektedir. Proje çıktısı uygulamanın bir CI/CD sürecine dahil edilerek uygulama dağıtımı ve testlerinin otomatize bir şekilde yapılması sağlanabilir. Bu sayede uçtan uca bir otomasyon mümkün olabilecektir.

Web Servis otomasyonu SOA mimarisi(Servis Tabanlı Mimari) ile çalışan ve verilerini müşterilere sunmak isteyen yazılım mühendisleri ve şirketler için oldukça verimli bir yöntem olarak gözükmektedir. Özellikle hızlı prototip geliştirmek isteyenler için çok faydalı olabileceğini söyleyebiliriz.

KAYNAKÇA

[1] C. Zolotas, T. Diamantopoulos, K. Chatzidimitriou and A. Symeonidis, "From requirements to source code: a Model-Driven Engineering approach for RESTful web services", *Automated Software Engineering*, vol. 24, no. 4, pp. 791-838, 2016.

[2] "Easily Create Restful APIs From Requirements". s-case.github.io/, (23 Temmuz 2019).

[3] V. Schreibmann and P. Braun, "Model-driven Development of RESTful APIs", *Proceedings of the 11th International Conference on Web Information Systems and Technologies*, vol. 1, pp. 5-14, 2015.

[4] J. Team, "JHipster - Generate your Spring Boot and Angular/React applications!", *Jhipster.tech*, 2019. https://www.jhipster.tech/. (24 Temmuz 2019)

[5] H. Ed-douibi, J. L. C. Izquierdo, A. Gomez, M. Tisi, J. Cabot. "EMF-REST: Generation of RESTful APIs from Models, "*31st ACM/SIGAPP Symposium on Applied Computing SAC'16,* Italy, *2016*

[6] "A next-generation rapid applications development tool for Java developers Spring Roo Project. (2019). Spring Roo Project. [online] Available at: https://projects.spring.io/spring-roo/ (24 Temmuz 2019).

[7] "The simplest and lightest code generator - Telosys" telosys.org. https://www.telosys.org/, (23 Temmuz 2019).

[8] "Expose your databases as REST web services in minutes" Restify Project https://restifydb.com, (23 Temmuz 2019).

[9] "DreamFactory API management", *Dreamfactory.com*, 2019. [Online]. Available: https://www.dreamfactory.com/. (24 Temmuz 2019)

[10] Y. Baghdadi, "Reverse engineering relational databases to identify and specify basic Web services with respect to service oriented computing", *Information Systems Frontiers*, vol. 8, no. 5, pp. 395-410, 2006.

[11] "Spring Framework". spring.io. https://spring.io/projects/spring-framework, (23 Temmuz 2019).

[12] "VueJS Official Website". https://vuejs.org/, (23 Temmuz 2019).

[13] "IEEE 1016-2009 - IEEE Standard for Information Technology--Systems Design--Software Design Descriptions", *Standards.ieee.org*, 2019. https://standards.ieee.org/standard/1016-2009.html. (24 Temmuz 2019).

[14] "Oracle Database Online Documentation - Database Administrator's Guide". https://docs.oracle.com/cd/B28359_01/server.111/b28310/views002.htm, (23 Temmuz 2019).

[15] M. Andersen, O. Chikvina, S. Mukhina. “Hibernate Tools Reference Guide”. http://docs.jboss.org/tools/latest/en/hibernatetools/, (23 Temmuz 2019).

[16] “A Java API for generating .java source files – JavaPoet”. https://github.com/square/javapoet, (23 Temmuz 2019).

[17] “Spring Data Rest”. https://spring.io/projects/spring-data-rest, (23 Temmuz 2019).

[18] O. Gierke, T. Darimont, C. Strobl, M.Paluch, J. Bryant. “Spring Data JPA Reference Documentation.” https://docs.spring.io/spring-data/jpa/docs/current/reference/html/, (23 Temmuz 2019).

[19] V.Choudhary. “Difference between http and WebSocket (HTTP 2.0)”. https://developerinsider.co/difference-between-http-and-http-2-0-websocket/, (23 Temmuz 2019).

[20] “The WebSocket API (WebSockets).” https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API, (23 Temmuz 2019).

[21] M. Wenzel, S. Smith. “Choose Between Traditional Web Apps and Single Page Apps”. https://docs.microsoft.com/en-us/dotnet/standard/modern-web-apps-azure-architecture/choose-between-traditional-web-and-single-page-apps, (23 Temmuz 2019).

[22] "Introduction to Tools and Products that Support Migration", *docs.oracle.com*, 2019. https://docs.oracle.com/database/121/DRDAA/migr_tools_feat.htm, (24 Temmuz 2019)

[23] A. Malpani, P. A. Bernstein, S. Melnik and J. F. Terwilliger, "Reverse engineering models from databases to bootstrap application development," *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*, Long Beach, CA, 2010, pp. 1177-1180.

[24] X. Qafmolla and V. C. Nguyen, "Automation of Web services development using model driven techniques," *2010 The 2nd International Conference on Computer and Automation Engineering (ICCAE)*, Singapore, 2010, pp. 190-194.

Tezden Üretilmiş Yayınlar

İletişim Bilgisi: mehmet.rasid.gencosmanoglu@std.yildiz.edu.tr

mrgenco@gmail.com

Konferans Bildirileri

1. Gencosmanoglu M.R. ve Selcuk Y.E., (2019) "Automation of RESTful Services Using Reverse Engineering and Code Generation", 6th International Symposium on Engineering, Artifical Intelligence and Applications(ISEAIA), 6-8 March 2019, Girne, North Cyprus