İSTANBUL TEKNİK ÜNİVERSİTESİ
Bilgisayar ve Bilişim Fakültesi

Java ile Web Servis'i Yazılması

STAJ

Kadir Emre Oto
150140032

YAZ / 2019

İstanbul Teknik Üniversitesi

Bilgisayar ve Bilişim Fakültesi

STAJ RAPORU

Akademik Yıl: 2019

Staj yapılan dönem: ☑Yaz ☐Bahar ☐Güz

Öğrenci ile ilgili bilgiler

Adı ve Soyadı:	Kadir Emre Oto
Öğrenci Numarası:	150140032
Bölüm:	Bilgisayar Mühendisliği
Program:	%30 İngilizce
E-posta Adresi:	otok@itu.edu.tr
(Cep) Tel No:	0 (507) 713 83 90
ÇAP öğrencisi misiniz?	☐Evet (ÇAP yaptığınız Fakülte/Bölüm: _____)
	☑ Hayır
Mezuniyet durumunda mısınız?	☐ Evet
	☑ Hayır
Yaz okulunda ders alıyor musunuz?	☐Evet (Ders sayısı: __)
	☑ Hayır

Öğrencinin çalıştığı kurum ile ilgili bilgiler

İsmi:	Erstream Video Delivery Company
Birimi:	Yeni Ürün Geliştirme ve Sistem Departmanları
Web Adresi:	http://erstream.com
Kısa Adresi:	Defterdar Mah. Otakçılar Cad. 80/9, Eyup | 34050 Istanbul | Turkey

Yetkili kişi ile ilgili bilgiler

.

Bölümü:	Yeni Ürün Geliştirme Departmanı
Unvanı:	Matematik Mühendisi
Adı ve Soyadı:	Yağız Buran
(Kurumsal) E-posta:	yagiz.buran@erstream.com
(Kurumsal) Tel. No.:	+90 212 705 95 17

Yapılan iş ile ilgili bilgiler

Staj yeri	☑Türkiye
	☐Yurtdışı
Staj başlangıç tarihi	05.08.2019
Staj bitiş tarihi	09.09.2019
Stajda çalışılan net **gün** sayısı	20
Staj süresince sigortanız var mıydı?	☑Evet, İTÜ tarafından sigortalandım.
	☐Evet, kurum tarafından sigortalandım.
	☐Hayır, yurtdışı stajı yaptım.
	☐Hayır.

İÇİNDEKİLER

150140032 numaralı, Kadir Emre Oto adlı öğrencinin, yukarıda "İçindekiler" bilgisi verilen staj raporu görülmüş ve uygun bulunmuştur.

Formu Dolduran Firma Yetkilisinin Adı ve Soyadı: Yağız Buran

Yetkilinin Ünvanı: Proje Yöneticisi

Müdür – İmza – Kaşe:

(Kurumsal) E-posta: yagiz.buran@inzva.com
(Kurumsal) Tel. No.: +90 212 705 95 17

1. KURULUŞ HAKKINDA BİLGİLER

15 yılı aşkın süredir piyasada olan *Erstream Video Delivery Company* **internet video teknolojileri** alanında ürettiği ürünler ve sunduğu hizmetler ile bu alanda öncü firmalardan biri olmuştur. Dünya çapında ün yapmış içerik sahipleri ve yayıncılar ile birlikte Türkiye'de yayıncılık yapan birçok firma Erstream ile iş birliği içerisinde çalışmaktadır.

Erstream'in dünya çapında yüzlerce internet ve televizyon kanalına yayın sağlayabilmesinin ve CDN (*Content Delivery Network*) hizmetleri verebilmesinin temel nedeni Avrupa'daki birçok ülkenin veri merkezlerinde encoding ve downlink hizmetlerine sahip olmasıdır.

Şekil 1 – Erstream Logosu

2. GİRİŞ

Ersteam staj programına kabul ettiği stajyerlere şirketin üzerinde çalıştığı birçok ürün ve sağladığı hizmetler üzerinde çalışma imkanı sağlamakta ve tecrübelerini aktarmayı hedeflemektedir. Bunlara ek olarak çeşitli araştırma-geliştirme (AR-GE) projelerinde de çalışma imkanı sunmaktadır.

2017 yılında yapmış Erstream şirketinde yaptığım stajda makine öğrenmesi ile kullanıcılara izledikleri filmlere göre yeni filmler öneren bir AR-GE projesi üzerinde çalışmıştım. 2019 yılında yaptığım bu stajda ise yeni ürün geliştirme ve sistem departmanlarında bir ürün olarak kullanılacak olan java programlama dili ile geliştirilmiş bir *web servisi* geliştirdim. Yeni ürün geliştirme departmanında bir projenin nasıl geliştirilmesi gerektiği, hangi adımların takip edilmesi gerektiği, hangi durumlarda hangi teknolojilerin kullanılmasının daha uygun olacağı gibi konularda detaylı bilgi edindim. Sistem departmanında ise geliştirilen ürünün sürekliliği bozulmayacak şekilde nasıl kullanılabilir duruma getirilebileceğini (*deployment*) görmüş oldum.

3. STAJ PROJESİNİN TANIMI VE ANALİZİ

Erstream şirketinde yaptığım bu projede video oynatıcılarında yaşanabilen gecikmeleri (*buffer* ve *start-up* zamanları) belirlenmiş ek parametreler ile birlikte kayıt altına (*logging*) alan ve bu kayıtlar üzerinde çeşitli sorgulamalar yapılabilen bir sistem yapmam bekleniyordu. Sistemin sahip olması istenilen tüm özellikler proje başlamadan önce proje yöneticisi tarafından belirlenmiş ve bana iletilmişti.

Yaşanabilecek bu gecikme kayıtlarının (*latency logs*) her biri aşağıdaki bilgileri içermesi gerekmektedir ve bu bilgileri kullanarak sistemde yaşanan herhangi bir sorunun kaynağının hızlı bir şekilde bulunması hedeflenmektedir:

- *type*: Gecikme türü ("*buffer"* ya da *"startup")*
- *value*: Gecikme süresi
- *user_id*: Gecikmenin yaşandığı kullanıcı bilgisi
- *content_id*: Gecikmenin yaşandığı içerik bilgisi
- *session_id*: Gecikmenin yaşandığı *session* bilgisi
- *public_key*: Servisi kullanacak firmaya atanmış eşsiz bir anahtar (*unique key*)
- *ip*: Gecikme yaşayan kullanıcının *ip* bilgisi
- *ISP*: Gecikmenin yaşandığı kullanıcının bağlı olduğu *ISP* (*Internet Service Provider*) bilgisi
- *country*: Gecikmenin hangi ülkede yaşandığı bilgisi
- *city*: Gecikmenin hangi şehirde yaşandığı bilgisi
- *app_name*: Gecikme yaşanan uygulamanın adı
- *device*: Gecikme yaşanan cihaz
- *sr_height*: Gecikme yaşanan içeriğin ekran yüksekliği
- *sr_width*: Gecikme yaşanan içeriğin ekran genişliği
- *bitrate*: Gecikme yaşanan içeriğin *bitrate* bilgisi
- *medium*: Gecikme yaşanan kanal (örnek: app, web)
- *os*: Gecikme yaşanan cihazın işletim sistemi
- *app_version*: Gecikme yaşanan uygulamanın versiyonu

Yazacağımız bu sistemin gecikme kayıtlarını uygun bir veri tabanına kaydedebilmesi ve 4 farklı sorguya cevap verebilmesi gerekiyordu, bu sorgular şu şekildedir ve ileride bu sorgular hakkında daha detaylı bilgiler verilecektir:

- *Add Latency*: Sisteme yeni bir gecikme kaydı eklemek için kullanılır.
- *Get Latencies*: Sorgulanan aralıkta belirtilen filtrelere uyan tüm gecikme kayıtlarını almak için kullanılır.
- *Get Latency Sum*: Sorgulanan aralıkta belirtilen filtrelere uyan gecikme sürelerinin toplamını almak için kullanılır.
- *Get Latency Sum by Date*: Sorgulanan aralıkta belirtilen filtrelere uyan gecikmeleri sürelerini zamana bağlı bir histogram şeklinde almak için kullanılır.
- *Get Latency Sum by Field*: Sorgulanan aralıkta belirtilen filtrelere uyan gecikmeleri sürelerini verilen *field* altında toplanmış olarak (*aggregation*) almak için kullanılır.

3.1 Projede Kullanılan Teknolojiler

Raporun bu kısmında projede kullanılan ve projeye başlamadan önce bilinmesi gerekli olan teknolojiler hakkında bilgi verilmiştir.

3.1.1 Elasticsearch

Elasticsearch birçok firmanın kullandığı, kolayca birçok sunucuda dağıtılmış olarak çalışabilen (*distributed*), açık kaynak kodlu (*open source*), neredeyse tüm veri tiplerini

destekleyen arama ve analiz motorudur. Veri boyutu arttığında bile yüksek hızlarda çalışabilmesi ve *kibana* gibi sistemdeki verileri görüntülemeye izin veren programların var olması *Elasticsearch*'ün en büyük artılarındandır.

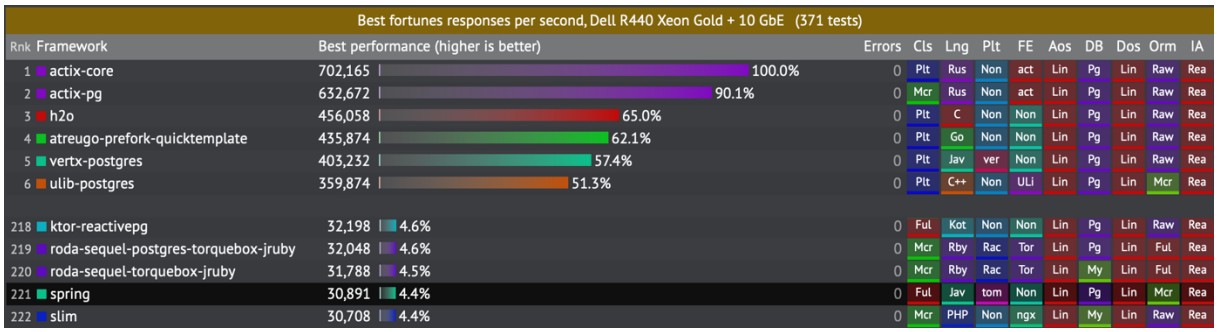
Projedeki sorgularının sayısı, sıklığı ve çeşidi göz önünde bulundurulduğunda *Elasticsearch*'ün proje için çok uygun olduğuna karar verdik.

3.1.2 Java Spring Web Framework

Spring, Java programlama dilinde yazılmış kolayca web uygulamaları geliştirmek için kullanılan bir *web framework*'üdür.

Web frameworklerinin ve kullanılan programlama dilinin saniye başına cevap verebildikleri istek sayısının farklı olması sebebiyle projeye başlamadan önce bu sayının en az kaç olması gerektiğinin belirlenmesi gerekir. Bu sayı incelendiğinde ve şirkette bu tür projelerde çalışan insanların genellikle java ve spring ile geliştirilmiş uygulamalara aşina oldukları göz önüne alındığında bu projede de spring kullanılabileceğine karar verildi.

Projede java programlama dilinin kullanılmasına karar verilmesinde etkili olan bir diğer sebep ise java dilinin dizayn modellerinin (*design patterns*) kolayca uygulanabilir olması idi. Özellikle büyük projelerin geliştirilmeden önce dizayn modellerine uygun bir şekilde tasarlanması ve kodlanması sürdürülebilirlik ve stabillik açılarından oldukça önemlidir.



Şekil 3 – *Spring Web Framework*'ünün Performans Sonuçları

Maven java tabanlı projeleri kolayca ve stabil bir şekilde derlenmesini amaçlayan bağımlı kütüphaneleri otomatik bir şekilde indirip kuran araçlardan biridir. Geliştirdiğimiz bu projeyi derlemek için bu aracı kullanacağız.

3.2 Projenin Yazılması

Raporun ilk kısmında belirtilen teknolojilerin belirlenmesin ardından projenin yazılmasına başlandı ve aşağıdaki adımlar sırasıyla izlendi:

3.2.1 Projenin Geliştirileceği Ortamın Ayarlanması

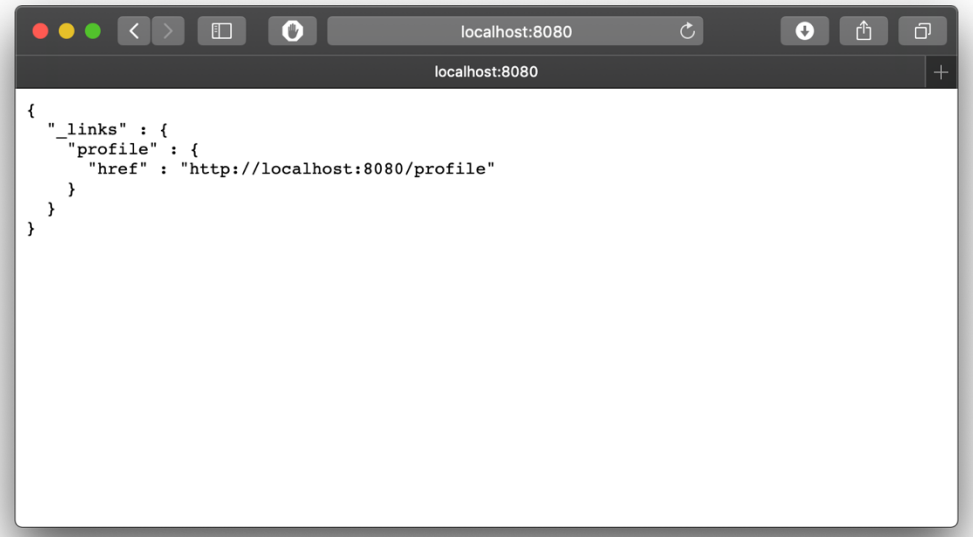
Tüm projeyi *MacOS* işletim sistemli kendi kişisel bilgisayarımda geliştirdiğim için bilgisayarımda projeyi geliştireceğim ortamı ayarlamam ve gerekli programları ve kütüphaneleri kurmam gerekiyordu:

- Java (versiyon: 12.0.1)
- Elasticsearch (version: 6.1.2)
- Intellij IDEA
- Postman

Proje tamamlandığında Linux işletim sistemi dağıtımlarından birinde servis edileceği için geliştirme ortamının da Unix tabanlı bir işletim sistemine sahip olması benim için bir avantaj oldu.

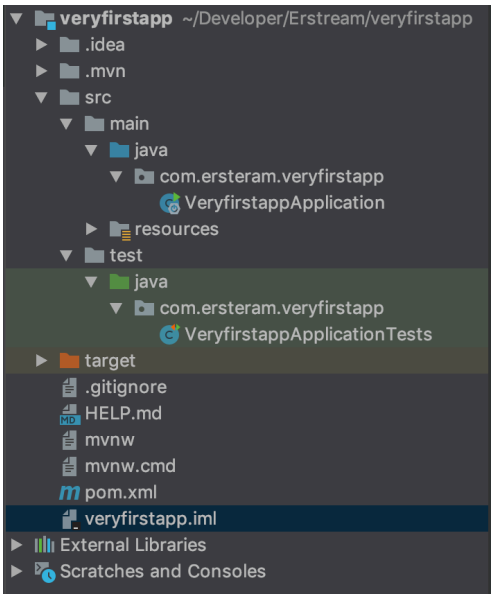
3.2.2 İlk *Spring* Projesi

Gerekli programlar kurulduktan sonra yeni bir *spring* projesi oluşturmak için Intellij IDEA kod editörü içerisinden "*Create a New Project*" butonuna basılması gerekmektedir. **Ek-1'de** bulunan 5 adım takip edildiğinde ve program çalıştırıldığında Şekil-4'teki gibi bir çıktı alınması gerekiyor.



Şekil 4 – *Spring* projesi oluşturulduktan sonra alınan ilk çıktı

Proje oluşturulduktan ve başarı bir şekilde çalıştırıldıktan sonra proje yapısının nasıl olduğunu incelemekte faydalı olacaktır. Şekil-5'te proje yapısının nasıl olduğu gösterilmiştir ve proje için önemli dosya ve klasörlerin ne olduğu kısaca açıklanmıştır.



- **/src/main/…/VeryfirstappApplication**: *Spring*'in nasıl başlatılacağı ile ilgili düzenlemelerin yapıldığı dosyadır.
- **/src/main/java/…/veryfirstapp:** Proje için gerekli tüm dosyalar bu klasör altında toplanmalıdır.
- **/src/test/…/VeryfirstappApplicationTests:** Programdaki herbir kod parçasının test edilebileceği (unit testing) dosyadır.
- **/pom.xmlh:** *Maven* tarafından kullanılan proje konfigürasyon detaylarını içeren *xml* dosyasıdır.
- **/mvnw:** Unix tabanlı sistemler için sistemde maven bulunmasa bile doğru maven versiyonunun indirilip çalıştırılmasını sağlayan dosyadır.
- **/mvnw.cmd:** /mvnw dosyasının *Windows* tabanlı sistemlerde çalışmasını sağlayan dosyadır.

Şekil 5 – *Spring* projesinin dosya yapısı (*project structure*)

3.2.3 *Elasticsearch* ve *Spring* Entegrasyonu

Herhangi bir java projesine *elasticsearch client*'ı eklemek için öncelikle **pom.xml** dosyasında "dependencies" bloğu altına aşağıdaki satırları eklememiz gerekiyor. Bu işlemden sonra maven gerekli kütüphaneleri sistemimize indirecektir.

```
<dependency>
    <groupId>org.elasticsearch.client</groupId>
    <artifactId>transport</artifactId>
    <version>6.1.2</version>
</dependency>
```

Bu işlemden sonra client nesnemizi oluşturup bu client üzerinden işlemler yapmaya başlayabiliriz.

```
Settings settings = Settings.builder()
        .put("cluster.name", clusterName)
        .build();

TransportClient client = new PreBuiltTransportClient(settings);
try {
    client.addTransportAddress(new TransportAddress(InetAddress.getByName(host), port));
} catch (UnknownHostException e) {
    System.out.println("Could not connect to Node: " + host + ":" + port);
}
```

3.2.4 *Buffer Logging System* (BLS) İsimli Sistemin Yazılması ve Test Edilmesi

Proje ürünleştirilip satılacağı için sistem dizaynını ve proje kodlarını rapora eklemek mümkün değil. Bu sebeple proje gizliliği bozulmayacak şekilde kod parçacıkları ve servisin işleyişi ile ilgili görseller bu kısımda aktarılmıştır.

Sistemin sahip olduğu 5 *end-point* için ayrı ayrı aşağıda incelenmiştir. Sorgulardaki ortak kısım *"public_key"* alanıdır. Bu *key,* servisi kullanacak şirketlere servisi satın aldıklarında verilir ve yapacakları sorguları 32 karekter uzunluğundaki bu *key* ile yapmaları gerekmektedir. Bu *key* ile şirketlerin servisi ne kadar kullandığını kontrol etme ve kısıtlama imkanına sahip oluyoruz.

3.2.4.1 Add Latency

Sisteme yeni gecikme kaydı eklemek için kullanılan *end-point*'tir. Sorguda belirtilen bilgiler doğru ise kayıt *elasticsearch*'e kaydedilir.

Aşağıdaki sorguda gördüğümüz alanlar incelendiğinde 2 şey dikkatimizi çeker:

- Sorguda daha önceki bölümlerde bahsedilen *"country"*, *"city"* ve *"ISP"* isimli alanlar bulunmamaktadır. Bu alanlar *back-end'te* (java - spring) *"ip"* bilgisi kullanılarak üretilmektedir. Bunun için **maxmind** şirketinin sunduğu **GeoIP** ürününü kullandık. Bu ürün ip adresleri ile ilgili bilgiler local bir veri tabanında tutar ve o veritabanında sorgu yaparak sonuçları üretir. Veriler local veritabanında saklandığı için sorgular çok hızlı cevaplanabilmektedir. Kayıt *Elasticsearch'*e eklenmeden önce bu bilgiler de üretilir ve o şekilde kaydedilir.
- *"immediately"* isimli daha önce bahsedilmemiş bir alan vardır. Servisteki end-point'ler incelendiğinde add_latency end-point'inin diğerlerinden çok daha fazla kullanılacağı tahmin edilmişti. Bu sebeple bu kısımda bazı optimizasyonlar yapmaya karar verdim. Elasticsearch ile olan bağlantı sayısını azaltmak için gelen istekleri 250'lik paketler halinde veritabanına eklemenin bu sorunu çözmekte faydalı olacağına karar verdik.

"immediately" isimli bu alan ise eklenmek istenen kayıt bilgisi kritik öneme sahipse ve hemen eklenmesi gerekiyorsa kullanılan alandır.

```
POST /add_latency {
  "type": "buffer",
  "user_id": "Unique5",
  "content_id": "Content 104",
  "session_id": "Session 3",
  "public_key": "public_key_keo",
  "ip": "31.206.33.82",
  "value": 1.3,
  "app_name": "Test4",
  "device": "Macbook",
  "immediately": true,
  "sr_width": 720,
  "sr_height": 1280.42,
  "bitrate": 14001,
  "medium": "app",
  "os": "windows",
  "app_version": "1.2.2"
}
```

```
{
    "success": true
}
```

3.2.4.2 Get Latencies

Bu end-point ile sorguda belirtilen aralıktaki tüm kayıtlara ulaşmak mümkündür. Belirtilen aralıkta ([*"gte", "lte"*] zaman aralığında) çok fazla kayıt olabileceği için ve tüm bu kayıtların sorgu cevabına eklenmesinin sistemi çok yoracağı ve yavaşlatacağı için bu sorguda sayfalama (*pagination*) yapılmaktadır.

Dolayısıyla bu sorgu için "*public_key", "gte", "lte", "page_index", "page_length"* alanları zorunlu kısımlardır. *"country"* ve *"device"* gibi ekstradan sağlanan alanlar ise sorguda filtreleme yapmak için kullanılır.

```
POST /get_latencies {
  "public_key": "public_key_keo",
  "gte": "12/07/2019",
  "lte": "13/06/2020",
  "page_index": 1,
  "page_length": 50,
  "country": "Turkey",
  "device": "Macbook"
}
```

```
{
    "success": true,
    "result": {
        "items": [
            {
                "type": "buffer",
                "value": 1.3,
                "app": "Test4",
                "device": "Macbook",
                "user_id": "Unique5",
                "content_id": "Content 104",
                "session_id": "Session 3",
                "ip": "31.206.33.82",
                "ISP": "Vodafone NET",
                "country": "Turkey",
                "city": "Istanbul",
                "medium": "app",
                "os": "windows",
                "created_date":
1562954662272
            }
        ],
        "totalCount": 1,
        "pageSize": 1000,
        "pageIndex": 1
    }
}
```

3.2.4.3 Get Latency Sum

Bu end-point ile sorguda verilen zaman aralığındaki toplam gecikme sayısını ve toplam kayıt sayısını öğrenmek mümkündür.

- "*public_key*", "*gte*", "*lte*" alanları zorunlu alanlardır.
- Filtreleme yapmak için *"type", "content_id"* gibi ekstradan alan eklemek de mümkündür.

```
POST /get_latency_sum {
  "public_key": "public_key_keo",
  "gte": "05/03/2010",
  "lte": "05/03/2020",
  "type": "startup",
  "content_id": "12715"
}
```

```
{
    "success": true,
    "result": {
        "latencySum": 271.112,
        "documentCount": 53
    }
}
```

3.2.4.4 Get Latency Sum by Date

Bu end-point ile sorguda verilen zaman aralığındaki kayıtları zaman histogramı halinde görmek mümkündür.

- "*public_key*", "*gte*", "*lte*" ve "*date_interval*" alanları zorunlu alanlardır.
- "*date_interval*" alanı histogramın zaman uzunluğunu temsil etmektedir ve alabildiği değerler şunlardır: *"quarter", "second", "minute", "hour", "day", "week", "year"*
- Filtreleme yapmak için yine *"type", "country"* gibi ekstradan alan eklemek de mümkündür.

```
POST /get_latency_sum_by_date {
  "public_key": "public_key_keo",
  "gte": "10/06/2018",
  "lte": "13/06/2019",
  "date_interval": "day",
  "type": "buffer",
  "country": "Australia"
}
```

```
{
    "success": true,
    "result": [
        {
            "latencySum": 138.637,
            "documentCount": 113,
            "distinctUser": 9,
            "date": "2019-05-
31T00:00:00.000Z"
        },
        {
            "latencySum": 41025.553,
            "documentCount": 1262,
            "distinctUser": 93,
            "date": "2019-06-
01T00:00:00.000Z"
        },
        {
            "latencySum": 49382.64550000005,
            "documentCount": 2058,
            "distinctUser": 49,
            "date": "2019-06-
09T00:00:00.000Z"
        }
    ]
}
```

3.2.4.5 Get Latency Sum by Field

Bu end-point ile sorguda verilen zaman aralığındaki kayıtları yine sorguda belirtilen alana göre toplanmış halde (*aggregation*) görmek mümkündür.

- *"public_key", "gte", "lte" ve "field"* alanları zorunlu alanlardır.
- *"field"* alanına sadece kayıtların sahip olabildiği alanlar yazılabilmektedir.
- Filtreleme yapmak için yine *"type", "content_id"* gibi ekstradan alan eklemek de mümkündür.

```
POST /get_latency_sum_by_field {
  "public_key": "public_key_keo",
  "gte": "10/06/2010",
  "lte": "13/06/2020",
  "field": "country",
  "type": "buffer",
  "content_id": "253255"
}
```

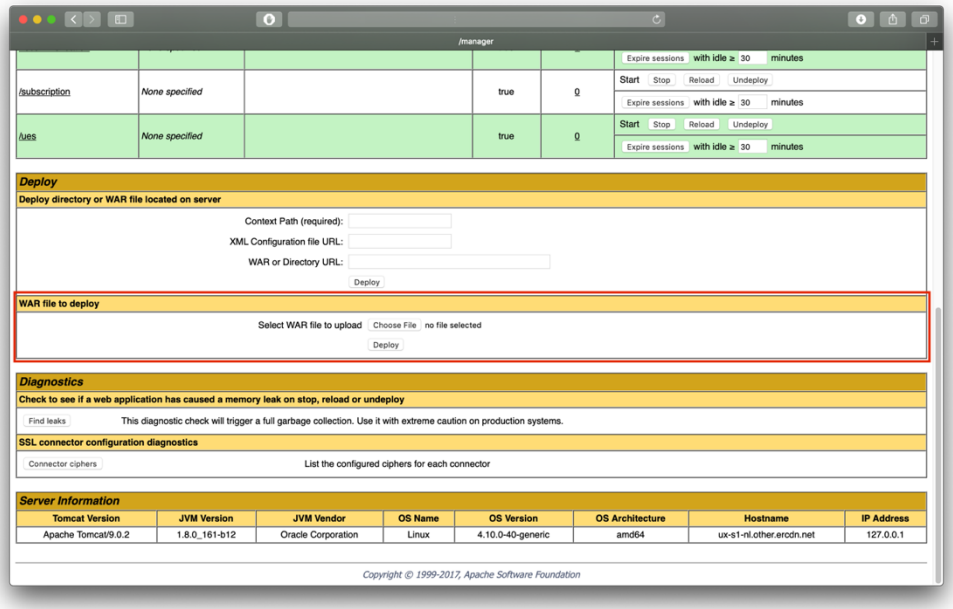
```
{
    "success": true,
    "result": [
        {
            "latencySum": 1954431.66039997,
            "documentCount": 30849,
            "distinctUser": 496,
            "value": "Australia"
        },
        {
            "latencySum": 1502.48300000009,
            "documentCount": 155,
            "distinctUser": 9,
            "value": "Turkey"
        },
        {
            "latencySum": 512.83,
            "documentCount": 185,
            "distinctUser": 91,
            "value": "Canada"
        }

    ]
}
```

3.3 Projenin Sunucuda Çalıştırılması (*Deployment*)

Son olarak proje tamamlandıktan sonra öncelikle test edilmesi, sonrasında da ürün olarak satılabilmesi için bir ya da daha fazla sunucuya kurulması gerekmektedir.

Tomcat, spring projelerini sunucuya kolayca kurmamıza izin veren çok kullanışlı bir yazılımdır. Proje derlenip *war* uzantılı dosya haline getirildikten sonra Şekil 6'daki sunucuya yüklemek mümkündür.



Şekil 6 –Projeyi *tomcat* ile sunucuya yükleme

4. SONUÇ

Yaptığım bu stajda iki farklı departmanın (yeni ürün geliştirme ve sistem departmanları) nasıl çalıştıkları hakkında detaylı bilgi edinmiş ve ürünleştirilebilir bir proje geliştirerek burda edindiğim bilgileri geliştirmiş oldum. Java programlama diline olan hakimiyetim arttı, Spring Web Framework'ünün ve RestFul API'ların nasıl çalıştığını teknik ve pratik açılardan inceleme fırsatı buldum. Servislerin test süreçlerinin nasıl yapıldığını, nasıl ürünleştirildiğini öğrenmek ise benim için çok değerliydi.

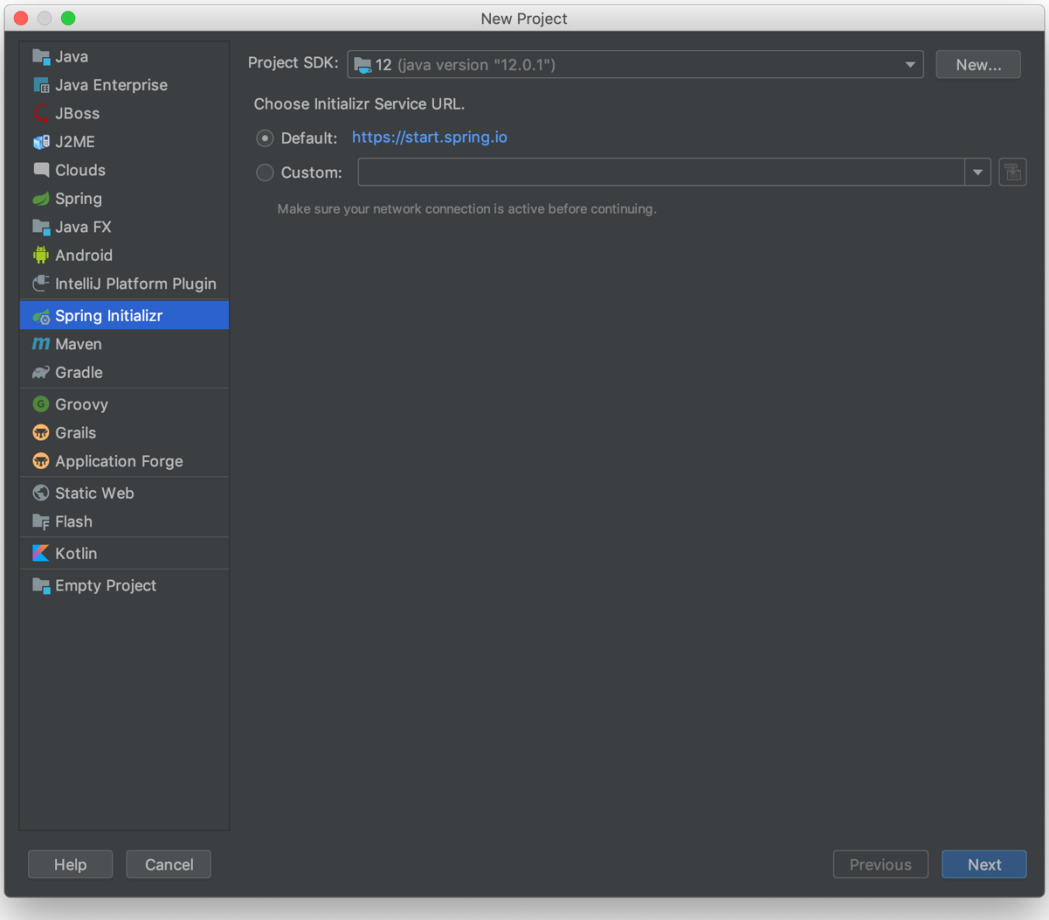
Şirket ortamının nasıl olduğunu, sektöre yönelik ürünlerin ve hizmetlerin nasıl üretildiğini gözlemleyebilmek bu alandaki bilgimi arttırdı. Alanında uzman kişilerin yardımı ile araştırma-geliştirme projelerinde nasıl bir yol izlenmesi gerektiği hakkında çok değerli fikirler edindim. Sağladığı bu staj imkanından dolayı Ersteam şirketine teşekkür ederim.

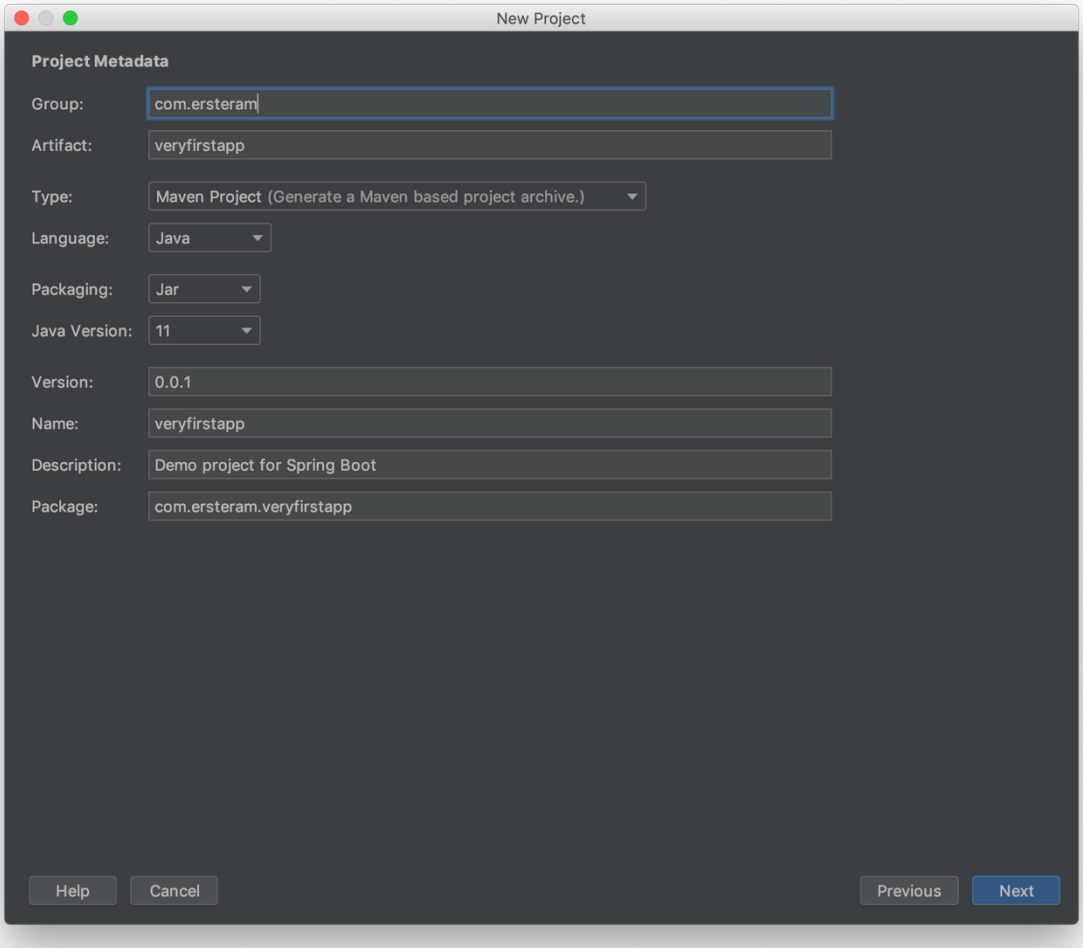
5. REFERANSLAR

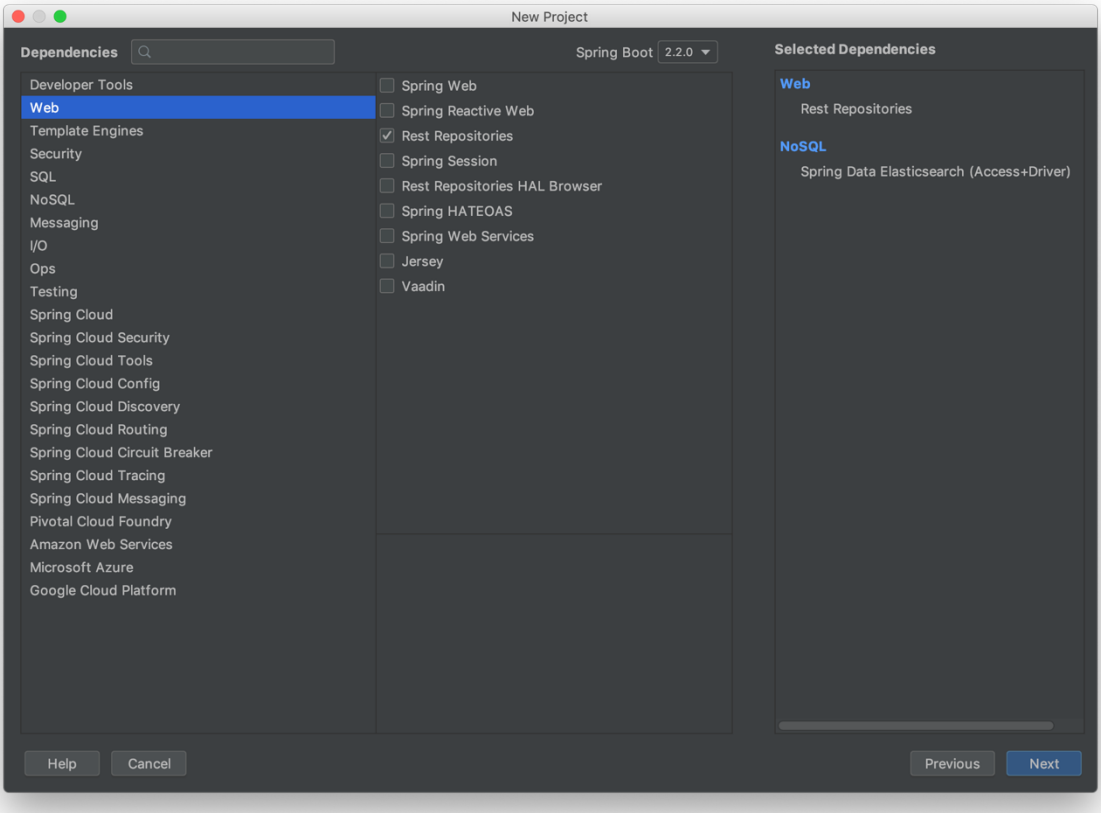
1. *Elastic Search.* http://www.elasticsearch.co/what-is/elasticsearch
2. *Web frameworks Benchmark Tests.* https://www.techempower.com/benchmarks
3. *Maxmind*. https://www.maxmind.com

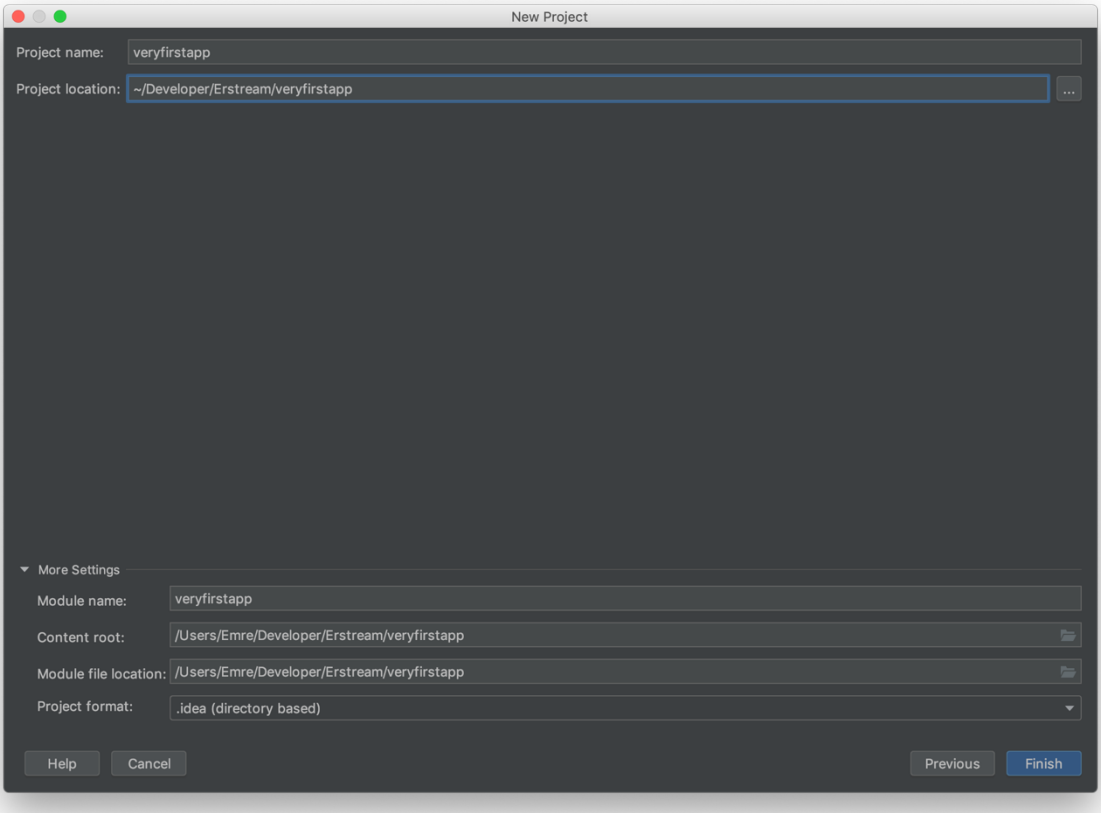
6. EKLER

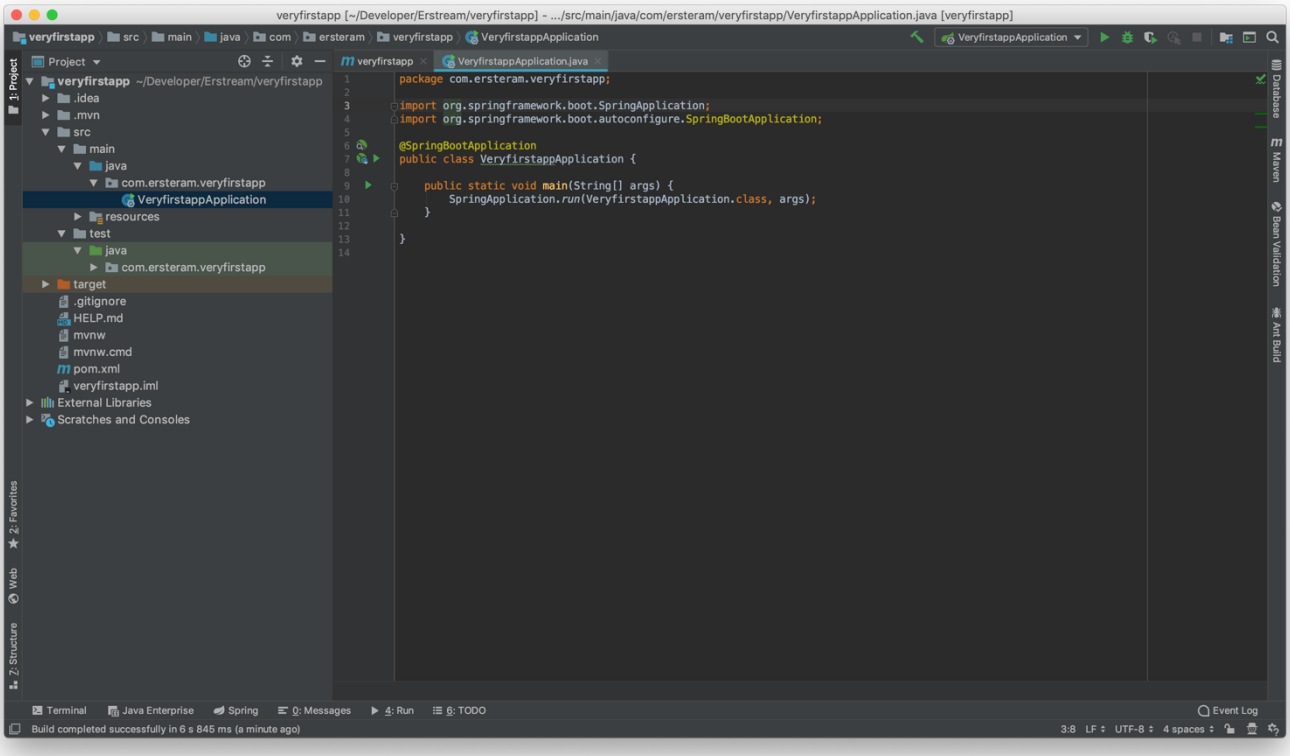
Ek 1: Yeni *spring* projesi oluşturmak için sırayla izlenmesi gereken adımlar, (5 resim)




New Project
Project Metadata
Group: com.ersteram
Artifact: veryfirstapp
Type: Maven Project (Generate a Maven based project archive.)
Language: Java
Packaging: Jar
Java Version: 11
Version: 0.0.1
Name: veryfirstapp
Description: Demo project for Spring Boot
Package: com.ersteram.veryfirstapp
Help
Cancel
Previous
Next


New Project
Dependencies
Spring Boot 2.2.0
Developer Tools
Web
Template Engines
Security
SQL
NoSQL
Messaging
I/O
Ops
Testing
Spring Cloud
Spring Cloud Security
Spring Cloud Tools
Spring Cloud Config
Spring Cloud Discovery
Spring Cloud Routing
Spring Cloud Circuit Breaker
Spring Cloud Tracing
Spring Cloud Messaging
Pivotal Cloud Foundry
Amazon Web Services
Microsoft Azure
Google Cloud Platform
Spring Web
Spring Reactive Web
Rest Repositories
Spring Session
Rest Repositories HAL Browser
Spring HATEOAS
Spring Web Services
Jersey
Vaadin
Selected Dependencies
Web
Rest Repositories
NoSQL
Spring Data Elasticsearch (Access+Driver)
Help
Cancel
Previous
Next


New Project
Project name:
veryfirstapp
Project location:
~/Developer/Erstream/veryfirstapp
More Settings
Module name:
veryfirstapp
Content root:
/Users/Emre/Developer/Erstream/veryfirstapp
Module file location:
/Users/Emre/Developer/Erstream/veryfirstapp
Project format:
.idea (directory based)
Help
Cancel
Previous
Finish


veryfirstapp [~/Developer/Erstream/veryfirstapp] - .../src/main/java/com/ersteram/veryfirstapp/VeryfirstappApplication.java [veryfirstapp]
Project
veryfirstapp ~/Developer/Erstream/veryfirstapp
.idea
.mvn
src
main
java
com.ersteram.veryfirstapp
VeryfirstappApplication
resources
test
java
com.ersteram.veryfirstapp
target
.gitignore
HELP.md
mvnw
mvnw.cmd
pom.xml
veryfirstapp.iml
External Libraries
Scratches and Consoles
VeryfirstappApplication.java
package com.ersteram.veryfirstapp;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class VeryfirstappApplication {
public static void main(String[] args) {
SpringApplication.run(VeryfirstappApplication.class, args);
}
}
Terminal
Java Enterprise
Spring
0: Messages
4: Run
6: TODO
Event Log
Build completed successfully in 6 s 845 ms (a minute ago)