

T.C.
FIRAT ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
STAJ DEFTERİ

BÖLÜMÜ	BİLGİSAYAR MÜHENDİSLİĞİ		
NUMARASI	14260581		
ADI VE SOYADI	ÖMER FARUK GENÇ		
STAJ DÖNEMİ	YAZ DÖNEMİ		
STAJ BAŞ. TARİHİ	20.06.2018	TOPLAM İŞ GÜNÜ SAYISI	40
STAJ BİTİŞ TARİHİ	15.08.2018		



STAJ YAPIŞAN İŞ YERİ BİLGİLERİ

İŞ YERİ	ADI	NETAS TELEKOMÜNIKASYON A.Ş.
	ADRESİ	Yenisehir Mah. Osmancı Bulvarı No:11 Kucukörs/İST
	TELEFON-FAKS	+90 (216) 522 20 00 - 522 22 22
SCRMU MÜHENDİS	ADI VE SOYADI	Mert KARAYEL
	ÜNVANI	Yazılım Mühendisi
	GÖREVİ	Design Engineer
AMİR	ADI VE SOYADI	Hatice Elçen Taşçı
	ÜNVANI	Yazılım Mühendisi
	GÖREVİ	Yazılım Mühendisi

BÖLÜM STAJ KOMİSYONU DEĞERLENDİRME SONUCU

Yapılan pratik çalışmanın iş günü Dönem stajı olarak kabul edilmiştir /edilmemiştir.

STAJ KOMİSYONU

I

İÇİNDEKİLER

KONU

Sayfa No

Haftalık Çalışma Çizelgesi 5 - 7

Günlük Staj Notları 8 - 48

Staj ile ilgili Görüşler 49 - 55

II

ŞEKİL, ÇİZELGE VE EKLER LİSTESİ

Sekil, Çizelge veya Ek No

Sayfa No

Mikroservis Mimarisi (Şekil 1.1)	49
Postman Yapısı (Şekil 1.2)	50
JSON WEB TOKEN Yapısı (Şekil 1.3)	50
Test Otomasyon Yapısı (Şekil 1.4)	51
Sanal makine vs Docker (Şekil 1.5)	51
Proje görselleri (Şekil 1.6-2.2)	52-55

STAJIN YAPILDIĞI KURUM VEYA KURULUŞUN TANIMI

NETAS, bilgi ve iletişim teknolojileri alanındaki yenilikçi ve yaratıcı çözümlerle, ulusal ve uluslararası pozarda, Servis sağlayıcılara ve kurumlara uygun uça katkıda değerli çözümler, sistem entegrasyonu ve teknoloji hizmetleri sunuyor. 1967'den beri sahip olduğu yetkinlik, geniş bilgi ve güçlü deneyimle iş verimliliğinin artışı için geleneksel NETAS, bilgisim teknolojileri alanındaki faaliyetlerini, ArGe biriminin katkılarıyla sürdürmeye, aynı zamanda Türk Silahlı Kuvvetleri'nin ihtiyaçlarını karşılamak amacıyla savunma iletişim açısından modernizasyonunda da önemli bir rol oynuyor.

Bilisim500 araştırmasında NETAS, 2017 sırasıyla "Donanım" ve "İOT ve M2M", "Ağ Donanımı", "Hizmet ihracatı", "Veri Yedeklemesi Depolama Donanımı" kategorilerinde "Yılın sistem entegratoru" seçildi. "Bilgisim Hizmet ihracatı" alanında "Ekonominde Katkı Özel Odaklı" aldı.

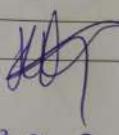
GİRİŞ

20 Haziran 2018 tarihinde NETAS'ta stajer olarak calip maya basladim. Bu haftayi tanisma ve dahil dunulon projenin ögrenilmesi, servislerin birbirlerini ile birlikte calisma kosullarini inceleyip öprendim. Sonra hattolarda ise kullanilan teknolojilerin genel yapisini, mimarilerini ve islegisini/kullonulagini ögrenip katkilar sapladim.

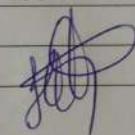
Orntayor programlarina katildim ve ekip donan egitimlere katilip kendimi her yorden gefertirmeye calistim. Test calismalarinda bulundum. Cesitli verilen gorevleri tamamladim. En son olarak ise Spring Boot projesi gelistirdim. Bu projede yeni teknolojiklerden olan Docker ve Hadoop'u kullanma imkan yakaladim. Bu sekilde gelistirmelerde devam ettim.

15 Agustos tarihinde ise 40 is gunthu doldurmus oldum. Bu surete yesodiyim tüm her seviyeye gecikme ileri sayfalarde gorebileceginiz sekilde yazdim. Bu sekilde Staj maceram noktalamiş oldum.

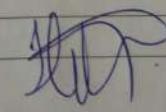
20.06.2018 tarihinden 22.06.2018 tarihine kadar bir haftalık çalışma

Gün	Yapılan İşler	Yapılan İşle İlgili Bilginin Bulunduğu Sayfa	Saat
Pazartesi			
Salı			
Çarşamba	Tanışma	8	8
Perşembe	Dahil olunan projenin ögrenilmesi.	9	9
Cuma	Servislerin çalışma ilişkisi ögrenildi.	10	9
Cumartesi			
Denetleyenin İmzası		Toplam Saat	26

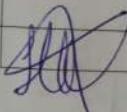
25.06.2018 tarihinden 29.06.2018 tarihine kadar bir haftalık çalışma

Gün	Yapılan İşler	Yapılan İşle İlgili Bilginin Bulunduğu Sayfa	Saat
Pazartesi	Mikroservis yapısı ögrenildi.	11	9
Salı	WebRTC yapısı ögrenildi.	12	9
Çarşamba	Redis yapısı ögrenildi.	13	9
Perşembe	RabbitMQ yapısı ögrenildi.	14	9
Cuma	Hazelcast yapısı ögrenildi.	15	9
Cumartesi		Toplam Saat	45
Denetleyenin İmzası			

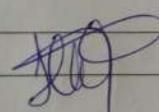
02.07.2018 tarihinden 06.07.2018 tarihine kadar bir haftalık çalışma

Gün	Yapılan İşler	Yapılan İşle İlgili Bilginin Bulunduğu Sayfa	Saat
Pazartesi	REST mimarisi ögrenildi.	16	9
Salı	PostMAN programı ögrenildi.	17	9
Çarşamba	PostMAN ile ilgili görev yapıldı.	18	9
Perşembe	Oryantasyona katıldım.	19	9
Cuma	JSON web Token yapısı ögrenildi.	20	9
Cumartesi		Toplam Saat	45
Denetleyenin İmzası			

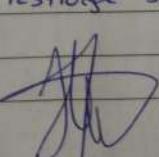
09.07.2018 tarihinden 13.07.2018 tarihine kadar bir haftalık çalışma

Gün	Yapılan İşler	Yapılan İşle İlgili Bilginin Bulunduğu Sayfa	Saat
Pazartesi	Teknik Orijantasyon programını katıldım.	21	9
Salı	API'imi test etmek için çalışmalar yaptım.	22	9
Çarşamba	Test çalışmalarına devam ettim.	23	9
Perşembe	Test çalışmalarına devam ettim.	24	9
Cuma	Test çalışmalarına devam ettim.	25	9
Cumartesi			
Denetleyenin İmzası		Toplam Saat	45

16.07.2018 tarihinden 20.07.2018 tarihine kadar bir haftalık çalışma

Gün	Yapılan İşler	Yapılan İşle İlgili Bilginin Bulunduğu Sayfa	Saat
Pazartesi	Ekip gezisine katıldım.	26	9
Salı	Test çalışmalarına devam ettim.	27	9
Çarşamba	Atanın görevi tamamlandı.	28	9
Perşembe	Yazılı fonksiyon geliştirildi.	29	9
Cuma	Yazılı fonksiyon proje içiğine edildi.	30	9
Cumartesi			
Denetleyenin İmzası		Toplam Saat	45

23.07.2018 tarihinden 27.07.2018 tarihine kadar bir haftalık çalışma

Gün	Yapılan İşler	Yapılan İşle İlgili Bilginin Bulunduğu Sayfa	Saat
Pazartesi	Postman collection testlodge'a aktarıldı.	31	9
Salı	Yeni servis için collection oluşturuldu.	32	9
Çarşamba	Test çalışmalarına devam ettim.	33	9
Perşembe	Test çalışmalarına devam ettim.	34	9
Cuma	Testlodge sisteme otomatik yapıldı.	35	9
Cumartesi			
Denetleyenin İmzası		Toplam Saat	45

30.07.2018 tarihinden 06.08.2018 tarihine kadar bir haftalık çalışma

VII

Gün	Yapılan İşler	Yapılan İşle İlgili Bilginin Bulunduğu Sayfa	Saat
Pazartesi	Docker teknolojisi öğrenildi.	36	9
Salı	Docker teknolojisi ögrenildi.	37	9
Çarşamba	Docker kurulumu gerçekleştirildi.	38	9
Perşembe	Docker ile yeni bir contaihıer oluşturuldu.	39	9
Cuma	Spring Boot projesi Docker ile oluşturuldu.	40	9
Cumartesi			
Denetleyenin İmzası		Toplam Saat	45

06.08.2018 tarihinden 10.08.2018 tarihine kadar bir haftalık çalışma

Gün	Yapılan İşler	Yapılan İşle İlgili Bilginin Bulunduğu Sayfa	Saat
Pazartesi	Spring Boot projesi için gerekli yapıldı.	41	9
Salı	Spring Boot yapısı öğrenildi.	42	9
Çarşamba	Projenin yapısı oluşturuldu.	43	9
Perşembe	Proje geliştirmeye devam edildi.	44	9
Cuma	Proje geliştirmeye devam edildi.	45	9
Cumartesi			
Denetleyenin İmzası		Toplam Saat	45

13.08.2018 tarihinden 15.08.2018 tarihine kadar bir haftalık çalışma

Gün	Yapılan İşler	Yapılan İşle İlgili Bilginin Bulunduğu Sayfa	Saat
Pazartesi	Proje de iyileştirmeler yapıldı.	46	9
Salı	HATEOAS eklendi.	47	9
Çarşamba	Performance iyileştirmeleri yapıldı.	48	9
Perşembe			
Cuma			
Cumartesi			
Denetleyenin İmzası		Toplam Saat	27

1. GÜN / 20 Haziran 2018

Bugün NETAS'taki ilk staj günüm. İlk olarak ufak bir hoşgeldiniz sunumıyla başlıyor ve ardından şirketin genel yapısını öğrenmeye çalışıyorum. Hemen ardından ekibimle tanıştım. Benimle işlenen projeler, sorunları nasıl çözüleceğini benden sorumlu mühendis ile birlikte sistemin ana yapısını inceledik.

Hangi teknolojilerin kullanıldığı, geliştirildiğimiz projelerin kapsamını, sağlayacağı faydalari ve kimlerle birlikte çalıştığımızı öğrendim.

Ekip arkadaşlarının hangi konularda uzman olduğunu ve projemize ne gibi katkılar sağladıklarını öğrendim.

Stajimin ilk günü bu şekilde geçti.

2. GÜN / 21 Haziran 2018

Dün kullanmadığım oldugumun teknolojileri bugün daha detaylı öğrenmek adına çalışmalarla başladım. Öncelikle bizim üzerinde geliştirmeler yaptığım projenin ismi "CPaaS" ingilizce "Communications Platform as a Service" cümlesinin baş harflerinden birinden gelmiş, Türkçe karşılığı ise "İletişim Platformu İnan Hizmet Platformu" olarak tanımlanabilir. CPaaS; yazılımların & gizliliklerin arka tarafta alt yapı, veya arabirimler oluşturmak gereklidirken kendi uygulamalarında, gerçek zamanlı iletişim özellikleri (ses, video ve mesajlaşma) eklemesine olanak sağlayarak bulut tabanlı bir çözüm sunar.

CPaaS bireye, API entegrasyonu ile genetik davranış ve yatırımlardan kaynaklanan sağırlı. En önemli ise para ve zaman sağırlı.

Sürdürülebilirlikteki kendi ürünlerini geliştirmeye yatırım yapmak gereklidir ve var olan kaynakların en iyi şekilde kullanmasına olanak sağları.

Bugünden bu şekilde yapmayı planlıyorum.

Tarih ve İşyeri Amirinin İmzası

21.06.2018


NİHAT YILDIZ
NETAS Telekomünikasyon A.Ş.
Telekomünikasyon Sistemleri - 1134912 Kırıkkale Peştepe Mahallesi
Kırıkkale - 26100 - TURKEY
V.O. Başkonsolosluğunda: 0322 000 1051
E-mail: info@netas.com.tr

3. GÜN / 22 Haziran 2018

Bugün, geliştirmekte olduğumuz API'nin iç yapısında olan servislerin birbirileyle çalışma ilişkisini, nasıl bir dizin kurulduğunu öğrendim. Projemiz, gerçek zamanlı iletişim platformunda çalışacağı için ve bu yönde maximum performans almak istediyimizden "mikroservis" ve "webrtc" gibi bilmemiştim 2 konu hakkında araştırma yaptım.

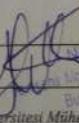
Mikroservis; bir sistemin herbinin bağımsız olarak çalışan ve aux protokoller (örneğin http) vasıtasyıyla birbiri ile iletişim kurul servislere ayrılmış, obruk tanınmayıabiliyor. Sadece bir işi yapın, gerekleştirmen ve gelişimine sürekli, bağımlılıklarını ve boyutlarını doldurabileceğini tutan etnik servisler obruk isimlendirilir.

WebRTC; Real Time Communication yani gerçek zamanlı iletişim tamamen web tabanlı obruk çalışan ve 3. parti programlara gerek kalmadan gerçek zamanlı iletişim sağlayabilecek bir özellikdir. Bu özellikleri JavaScript API'si kullanarak gerçekleştirir. Günümüzde von der cephé tarayıcı webRTC destekler. Öncelik vermek gereklisi Google Chrome, Mozilla Firefox, Opera, Android, iOS gibi.

Bugün bu konuları araştırdım.

Tarih ve İşyeri Amirinin İmzası

22.06.2018


NETAS
NETAS Telekomunikasyon A.Ş.
Bülent Ecevit Mah. 10. Sokak No: 3002 Kocatepe/Ankara
Bülent Ecevit Mah. V.D.Bakırköy 652 000 1051
Firat Üniversitesi Mühendislik Fakültesi
Sıfır İletişim İletişim Şirketi 999540004

1. GÜN / 25 HAZİRAN 2018

Bugün mikroservis yapısını daha detaylı inceledim.

Mikroservisler birbirinden bağımsız ve tek bir işe odaklanan uygulamalar olmaktadır, her bir servisi farklı bir programlama dili ile geliştirmek mümkündür. Bu da birece yani uygulamanızda tek bir dille olan bağımlılığını ortadan kaldırır. Bununla birlikte servislerin canlıya alınması (deploy edilmesi) birbirinden bağımsız oldukları için oldukça hızlı ve developer açısından oldukça büyük bir zaman kaybı sağlayacaktır. Ayrıca bir developer katıldığındaysa uygulamanın bütün yapısını öğrenmek yerine katkı vermeni bekleyen servisin yapısını öğrenip katkı vermeye başlaması oldukça doğal olacaktır.

(Şekil 1.1)

İlgili görselle bakıldığında, mikroservisler küçük, bağımsız ve bütün sistemin fonksiyonel yapısı etrafındaki giden/İASA edilmiş uygulamalardır. Birbirlerle çatışmaya ihtiyac duyduklarında açık protokoller üzerinden (HTTP, UDP, messaging) birbirlerle serbestçe iletişim kururlar. Genelde bir ya da iki kolici teknolojile (Redis, mongoDB) desteklendigidinden bu teknolojilere en uygun programlama dilinde geliştirilebilirler. Şekilde API Gateway adında bir katman görünüyor. Günümüzde birçok farklı client teknolojisi (browser, mobil cihazlar) servislerinizi kullanmak isteyorsunuz. Bu clientların servislerinize direkt ulaşması bir de probleme sebebi olabilir. Bu sebepten ötürü servisleriniz ile clientları bağlayan bu ortak katmanı API Gateway denizyor.

Tarih ve İşyeri Amirinin İmzası

25.06.2018

 NETAS
NETAS Teknoloji ve İletişim A.Ş.

Fırat Üniversitesi Mühendislik Fakültesi
Bölüm: İletişim Teknolojileri
V. D. Başkumru: Prof. Dr. İsmail Tuncer
Tc. Kimlik No: 04361401-4

5. GÜN / 26 Haziran 2018

Bugün ise webRTC konusunda detaylı bir orastuma yoptım. "Web Based Real Time Communication" yani Türkcesi "Web Tabanlı Gerçek Zamanlı İletişimi" olarak isimlendirilen Web tarayıcılarının Javascript API'leri ile perakən zamanlı ses, video, dosya, metin ve ekran paylaşımını sağlayan teknolojidir.

WebRTC ile birlikte kullanılan protokoller nelerdir?

- SIP (Session Initiation Protocol),
- RTP (Real Time Transport Protocol),
- STUN (Session Traversal Utilities for NAT),
- ICE (Interactive Connectivity Establishment) gibi protokoller bulunmaktadır.

SIP; 2 veya daha fazla katılımcı arasında logi karen Ağ protoköldür. Bu protokol RFC 3261 adınumında açıklanmış ve internet telefonu için kullanılan en yaygın protokoldür.

RTP; gerçek zamanlı ses, görüntü ya da simülsyon verilerinin iletisini ve taşınmasını sağlayan protokoldür.

STUN; istemci kendi public IP'sini öğrenmek için STUN ile iletişimde yerler ve aldığı IP bilgisine sahip olmak gereklidir.

ICE; Birbirinden farklı 2 es arasında en iyi yolu seçerek webRTC'inin kullanmasını sağlar. Eğer çiftler peer-to-peer konuşabiliyorsa ilk tercih bu olacaktır, eğer konuşamıyorsa TELN (öğütmen) seçer. Bu işi paralel yaparak

WebRTC, tarayıcıların içinde pörşenli fehriplinden kurulum gerçekleştirmez, buda kullanıcıların kolay ve hızlı şekilde VoIP kullanımına katkıda bulunur. VoIP, internet üzerinden ses verisi pörderilmektedir. (Bir yeni telefon şırasını)

P2P, 2 bilgisayar sunucu ihtiyaci olmadan doğrudan iletişim kurmasıdır.

Tarih ve İşyeri Amirinin İmzası

26.06.2018

NETAS

Fırat Üniversitesi Mühendislik Fakültesi
Bölüm: Mühendislik Mühendisliği
Tc.İd. No.: 1401240001061
Tc.İd. No.: 1401240001061

6. GÜN / 27 Haziran 2015

Bugün staj scrumteam 3 güne yarın 3 tane setinde, Redis, RabbitMQ, Hazelcast konularına bakmamı sağladı. İlk önce Redis ile başladım; En basit haliyle Redis'ın key-value şeklinde veri tabanının NoSQL veritabanıdır. Verileri HOD'a yozmadan RAM üzerinde tutmaya yorulan bir platformdur. NoSQL manşetyle çalıştığı için server kapasite dahil verilerin kaybolmasına izin vermez. SQL yapıları CPU kullanımını düşürmektedir. İstediğimiz işlem yapılıp, istenilen ciddi bir yük olur, Redis gibi yapılarla bu yükü azaltıp hız konusunda ciddi performans ortesi sağlanmış oluyoruz. Özellikle XML, JSON yapılarına ihtiyaç duyulduğunda scriptler olsa, halekçe mye fırsat veriyor. PHP, Python, Ruby, Java gibi dillerde sorunsuz çalışıyor.

Kullanım amacı; anlık verilerin tutulması, Sıfırınizasyon, tıpkı
istemeleri ve session değerlerinin konumlandırılması gibi konular doğrultusunda
diğer veritabanlarında farklı olarak konumlandırılmalıdır. Ayrıca
veriyi diske yazma操作da bulunmakta. Üzerinde tuttuğu
değerleri String, HashMap, List, Set gibi tiplerde tutabilemektedir.
Birkaç Notasyon su şekilde dir.

- ④ Cacheable → ilgili metodun sonucu ayırtlan cache kütüphanesinde逻辑上可缓存
 - ⑤ CachePut → Cacheable ile aynıdır farklı farklı metodları tekrar çağırır/call ettiğim

Bon Reatis komutator.

- SET: Anahtar değerine verilen eklementin sağıbr. (SET key "value")
 - GET: Anahtar üstündeki veriye ulaşmanın sağbr. (GET key)
 - DEL: Anahtar üstündeki veriyi silmemenin sağbr. (DEL key)
 - KEYJ: Kayıtlı anahtarlarla ulaşmanın sağları. (KEYJ *)

Bugün bu şekilde tanışmadım.

Tarih ve İsveri Amirinin İmzası

27/06/2018

NFTAS

20.18
M. M. Telekomunikasyon
Babatay, 313912 Konya, Posta Kodu: 000 1061
Baskı Tarihi: 20.01.2018

7. GÜN / 28 HAZİRAN 2018

Bugün RabbitMQ hakkında araştırmalar yaptım.

RabbitMQ; Herhangi bir kaynaktan alınan mesajın, sırası geldiğinde başka bir kaynağa iletilmesini sağlar. Amacı herhangi bir kaynaktan aldığı mesajı, sırası geldiğinde iletmesidir. Montek olmak Redi's Pub/Sub'a benzerekte örnek hizmet bir sıra söz konusudur. Yani iletimin yapıldığı kaynık sağlığa kalkan hizmet, tüm istekler kaynaktadır.

Günlük hayatı örnekle verecek olursak, kargo süreci en fazla ve en sıkıdır önek olarak görülebilir. Yani siz (producer) kargonuzu kargo şirketine (RabbitMQ quelesi) teslim edip sunucusu şirket sizin kargonuzu göndermek istediğiniz kişiye (consumer) teslim ediyor.

Neden Kullanılmalı?

Aşenkron olarak kullanılabilir yapısı vardır. Zaman ayarlı ve otomatik mail gönderme önek olarak verilebilir. Aşenkron olması bize sunucu cihazları maliyeti azaltmayı sağlar.

Çalışma Montifiş'

Producer: Kuyruğa mesajı gönderen uygulama.

Consumer: Kuyruğu dinleyen uygulama

Routing Key: Mesajını yörtenlere birebir分配 etmemi.

Exchange: Mesajı ilgili "routing key"'e göre ipelli kuyruğa yönlendiren bölüm

Queue: Mesajların son olarak döşeyen kuyruk.

Exchange Type: Gelen mesajın hangi kuyruğa nasıl gönderiliceğini belirtir.

Tipleri; Direct, Fanout, Topic ve Header'dır. Direct'te "routing key" belirlenip bu bilgi kuyruğa yazılır. Consumer buna göre işlem yapar.

Fanout: Mesajlar "exchange" de yer alan tüm kuyruğa gönderilir. Yüklendirme şartları olmamakla birlikte, bu ~~NETAS~~ Tercih etmemelidim.

Tarih ve İşyeri Amirinin İmzası

28/06/2018

Netas Telekomunikasyon A.S.
Yerel İstihdam Çalışma Programı No: 11.2012.0001061
Fırat Üniversitesi Mühendislik Fakültesi, Mühendislik Fakültesi
Baskası: Prof. Dr. İsmail Tuncer
T.C. 11.2012.0001061.0001061

8. GÜN (29 HAZİRAN 2018)

Bugün Hazelcast İde'nin obklemortasyonunu ne iş yaptığını, neler sağladığını öğrendim.

Hazelcast, açık kaynak kodlu Java tabanlı ve kompatible, dağıtılmış bir yapısız veritabanı sunucusu olacak hafızada tutar. JVM'ler sunuslu verileri eşit şekilde kolayca etrafında yerleştirebilir. Özellikle pojekti, hizli, güvenli ve en fazla veri kaybı için kolay bir yapısız JVM'ler ile sunucular arasında her veri yeriin dağılımını belirleyebiliriz. Aynı zamanda C# da destekler. Maven dostudur. Yönetimi hem console, hem de web arayüzü üzerinden yapılabilir.

Hazelcast, verileri hafızada Map şeklinde tutar. Sadece yinelemekle kalmayıp, verileri kitterme, kuyruğa alma gibi işlevleri yelpazeye yerleştirmektedir. Veriler RAM'de tutulurken bir o kordon'da backup şeklinde yine bellekler arasında paylaşılırken veri kaybı en az indirimini sağlıyor. Sadece Map değil, aynı zamanda Veriler: Queue, List, Set tipindeki tutulabilir. Var olan sunucular ve Hazelcast sisteme yeni sunucu ekleme JVM ekleneninde veriler heren paylaşılmasına başlıyor.

Proje de kullanılmış;

```
Config config = new Config();
HazelcastInstance instance = Hazelcast.newHazelcastInstance(config);
Map<Integer, String> mapKisi = instance.getMap("kisi");
mapKisi.put(1, "Faruk");
mapKisi.put(2, "Ömer"); // öteyerek mapimizi oluşturuyoruz
özet kolon ise Hazelcast Config ayarları ve isteyeceğimiz adresi belirtmek kolaydır. Bu içinde bu şekilde bir NETAS.
```

9. GÜN / 2 Temmuz 2018

Bugün geliştirmekte olduğumuz API, REST mimarisine sahip olmaktadır. İlk ise anında başladım. Once API'ın tanımını yaparak başlayalım. Bir uygulamanın istedigimiz bölümünü farklı bir uygulama ile komutlarken iain oluşturulan module "Application Programming Interface" yani API deniyor.

SOAP nedir? Farklı uygulamalarda web servislerin haberleşmesi için tasarlanan, RPC modelini kullanır. İstemeç / Sunucu mimarisine dayalı bir mimardır. Tüm mesajlar XML ile iletilir. Bize proxy ve WSDL kullanmışız olur. Bu konuda esnek değildir.

REST nedir? Bu yaklaşım SOAP'in alternatifidir. Öncelik ortaya çıkarılmıştır. Bir veri iletimi yoludur. Avantajları; RPC modeli yerine HTTP protokolü üzerinden iletişim sağlar. Esnekdir, kestin standartları yottur. Veri ister JSON, ister XML isteyse TEXT olarak veri alımı imkanı sunar.

REST hangi gerekliliklerimi karşılamıyor? Versiyonlama, Sayfalar, Kullanıcı doğrulama, Yetkilendirme, Silinen / Filtrleme, Döndürme, Caching, Temel Metodlar.

- GET: Veri listeleye / görüntüleme için kullanılır.
 - POST: Veri eklemek için kullanılır.
 - PUT: Veriyi güncelleme isteği olarak kullanılır.
 - DELETE: Veriyi silmek için kullanılır.
 - PATCH: Verinin sadece 1 parçasını güncellenet için kullanılır.
 - OPTIONS: Bir API URL'ine istek yapıldığında o URL hizmetleri katıldığını HTTP status kodları
- 1XX → Başarıyandırma, 3XX → Yönlendirme, 4XX → Kullanıcı hatası
 2XX → Başarılı işlem, 5XX → Server tarafı hatası

NETAS

Tarih ve İşyeri Amirinin İmzası

02.07.2018

Fırat Üniversitesi Mühendislik Fakültesi

Netas Telekomünikasyon A.Ş.
 1. Ünited Kütahya Pazarı
 16120 Kütahya / TURKEY
 Tel: +90 222 230 00 00
 Fax: +90 222 230 00 01

10. GÜN / 3 Temmuz 2018

Bugün geliştirmekte olduğumuz API'ı test edebilmek adına kullanabileceğimiz programı öğrenmem istendi. Programın ismi POSTMAN. Bu program ile akıllı dokümantasyon oluşturuyor.

(Şekil 1.2)

Öncelikle Postman üzerindeki "Environments" yapısını bilmek lazımdır. Bu tür environments, birde fast deployment, olan (development, staging, production) API'lar için oluşturulmuş tanımlı değişkenler tutabildiğimiz yapılardır. Hemen bir environment tanımlayarak başlayalım. Environment içinde tanımladığımız değişkenleri request atarken requestin cresitli yerlerinde kullanmakta有用. Örneğim bir uclu environment olarak tutmak istiyorum. Bu şekilde sadece request atacağım pathler değişeceler. {{url}} şeklinde erişilebilir. {{url}}/v1/app.json adresine request attığımında url kısmını direkt tanımladığımın kısımı da altı yerle koymamız gerekmektedir.

Postman Pre-Request Script adımlarımız kısım bir istek gerektilmeden önce çalışacak işlevleri bu adımda tanımlıyoruz. Örneğim; bir adrese istek yapabilmek için yetkili giriş yapmanız gerekiyor. Bu kısımda önce tokenimizi alıp o şekilde istek yapabaktır.

Postman Request Script kısımında ise istek attığımız adresten dönen cevabı yani response'u kontrol edebiliyoruz.

Collection Runner kısımında ise, bir collection'daki endpointları sizin verdığınız değişkenler ile n tane çalıştırmağa yarıyor. Bu sayede tek tek environment'ı değiştirmenize gerek yok.

Tarih ve İşyeri Amirinin İmzası

03.07.2018

Fırat Üniversitesi Mühendislik Fakültesi

Netas Teknikomunikasyon A.Ş.
T.C. 11.3401.000.000-1001
Büyükdere Mah. Çankaya Cd. No: 652/100/1001
Büyükdere Mah. Çankaya Cd. No: 652/100/1001

NETAS

11. GÜN / 4 Temmuz 2018

Bugün benden sorumlu mühendis bana bir görev verdi. Dün öğrenmiş olduğum progradan yani POSTMAN'de Request Script aracılığı ile istek attığımın adresinden cevapları kontrol edebiliyorduk. Bunu biraz daha açıkayım. Örneğin dönen cevapta aradığımız bir String var mı? Attemp'ın isteği lorsilik dönen response kaçımsı da dindü? Dönen cevabın status kodu nedir? Dönen cevabın header kısmında Content-type var mı? Dönden belirtmiş olduğumuz semajen wifun bir data dindü mü?

Bana bir excel dosyası verildi. Bu dosyada yazdıklarla göre istek yapacağımın path, POST-PUT gibi isimlerde gönderileceğim body ve dönmescini beklediğimi response'lar vardı.

İlk olarak Excel dosyasındaki belirtilen Test Case'leri öğrenmekle olduğum Postman'ı gevirdim. Bu case'lerin isimlerini ve path bilgilerini girdim ve hedef metod ile (GET, POST, PUT, DELETE) istek yapınası isteniyorsa onu girdim. Dönmescini beklediğimi cevapları da Postman'de bulunan "Examples" kısmına ekledim.

Bugünü bu şekilde tamamladım.



12. GÜN / 15 Temmuz 2018

Bugün Organizasyon günüydi. Yaklaşık 100 stagjer ile birlikte Düzenlenen organizasyona katıldık. İlk olarak birbirimizle tanıştık. Daha sonra NETAS kurumsal videosunu izledik. Ne gibi projelerin Olduğu bir sunumla devam ettilik.

Ardından NETAS'lı olmanın avantajlarının anlatıldığı ve çalışanların pazarda NETAS nedir? videosunu izledik.

Sorularımızı sorup yemepe indik. Yemeğten sonra kürk bir ayın aynadak ile iletişimle dolanlı bir sunum da gösterildi.

Ardından laboratuvar perisi yapıldı. Savunma Departmanını gezdikten sonra günün tamamlanarak

13. GÜN / 6 Temmuz 2018

Gorsamba günde kaldığım yerden hızlı bir şekilde devam etmek üzere çalışmamıza başladım. Sistemimizde güvenlik adına kullanacağımız JSON Web Token yapısını öğrendim.

Kısaca yapısı 3 parçadan oluşur "base64" formatında kodlanmış bir blokten oluşuyor. Bu 3 parça sırasıyla ; Header, Payload, Signature. Header kısmında bloğun tipi ve şifreleneceğine ait algoritması vardır. Payload kısmında claim olarak nitelenenler "Email, UniqueId, Username vb." gibi bilgiler tutulur. Son parça yani Signature kısmında ise header ve payload şifrelenecek kullanılmaktır. Jeldi ve şifreleneceğine ait algoritmanın kodlarıyla ürettilir.

En temel kullanım senaryosu "Kimliklendirme" dir. Kullanıcı überası genetik adrese direkt senaryoda username ve parola bilgisini girerek hangi kaynaklara/web sayfalarına erişebileceği bu token bilgisile kontrol edilir. Korunmuş yani yetkili kişi tarafından bir adrese istenilen tarafından Authorization: Bearer <token> şeklinde iletilmelidir.

Farklı nömler arasında token değişim-tokanı sahibi kişi tek seferlik giriş senaryolarında yapan şekilde kullanır.

Aventajları ; JSON kullanması, URL üzerinden trasnfer edilebilmesi, web browserlerde (cookies) kullanımı sorunlu olmaması, CSRF ataklarına karşı otomatik olarak koruma yapabilmesi, token okutturulma engellemesi, web uygulamaları arasında HTTP session paylaşımı, stateless kullanım uygun olması ve very bölgelerde paylaşılması.

(Şekil 1.3)

NETAS

Tarih ve İşyeri Amirinin İmzası

06.07.2018

Pınar Üniversitesi Mühendislik Fakültesi

Netas Telekomunikasyon A.Ş.
Yenibosna Mah. 1. Blok No: 10
06700 Ankara - TURKEY
+90 312 300 00 00
+90 312 300 00 01

14. GÜN / 9 Temmuz 2014

Bugün teknik oryantasyon günüydü. Caddeye istinatlı online test yapılabilen sistem tanıtılıdı. Ardından bilimde çeşitli firmalar GEHBANB firması hakkında bilgi verildi. Multimedya uygulamaları anlatıldı. Ardından yeni ürünler ve teknolojiler, IoT ile bilimde sunuldu.

Kahve makasının ardından SPİDR & KANDY ürünleri birlikte anlatıldı. Sonrasında C2D ve NFU konuları sunuldu.

Hemen ardından Siber güvenlik departmanının şefi yetkili ufak bir tanıtım sunumu gerçekleştirdi.

ICT çözümleri konulu bir sunum da yapıldı. Ardından Test Attornottom hukuki mülkiyet yetkili şefi bilgiyi yaygınlaştırmak üzere şirketlerin ürünlerini nasıl test ettiklerini, neler yapıldığını, hangi işlerde uygulanıyorlarını anlattı. Bugünde bu şekilde tanışmadık.

İFTAS

15. GÜN / 10 Temmuz 2018

Gecen hafta bize verilen görevde API'ımızı test etmek için ilk挑战mamızı tamamlamıştık. Bu gün bize verilmiş olan Excel dosyasına https://[https://](#) dğinden Postman aracılığı ile Directory servisi için challenge'ını devama ettirdim.

Token birim sisteminde de var olsa bir yapı olduğunu istek yapacağım path'e önceden tokenimi verip atman gerekiyor. Bu nedenle Pre-Request Script kumunu kullanıyorum. Bu Javascript ile token almamızı yapacak bir fonksiyon yazdım. Bu fonksiyon şu şekilde;

```
var getTokenRequest = {
    URL: getTokenPath,
    method: "POST",
    header: "Content-Type:application/x-www-form-urlencoded",
    body: {
        mode: "urlencoded",
        urlencoded: [
            { key: "username", value: UTILS.testUser },
            { key: "password", value: UTILS.testUserPassword },
            { key: "grant_type", value: "password" },
            { key: "client_id", value: pm.environment.get('clientId') },
            { key: "client-secret", value: pm.environment.get("clientSecret") },
            { key: "scope", value: "openid" } ] } };
```

Bu method bize belirlediğimiz username ve password'a göre sunucuya istek atıp token üretiyor. Bu şekilde kullanıcıların tokenini alıyor.

NETIAS

16. GÜN / 11 Temmuz 2018

Dün istekte buluştığımız adreslere tıkla adıock devam edip isteği başarıyla attık. Bunu sonucunda lütfen beklediğimiz bir cevap doldurdu. Ancak bir henu kontrol etmek istiyorum. Geriye kalan içinde ne olduğunu söyleyebilir mi?

Bunu şu şekilde yapıyoruz. Request Script Test kısmında önce elimize düşen datayı alınması gerekiyor.

Var jsonData = pm.response.json();

Bu kod ile artık jsonData değişkenimiz içinde responsumuz var. pm.response ifadeleri postman'e özgü bir yazım dili.

Örnek vermek gerekirse;

```
pm.test ("response status code is 200", function () {
    pm.response.to.have.status(200);});
```

Şeklinde istek attığımız adresden dönen status kodu 200 mü buna kontrolünü gerçekleştiren test. Bir başka örnek;

```
pm.test("Content-Type is present", function () {
    pm.response.to.have.header("Content-Type");});
```

Burada da attığımız isteğe dönen cevabın header kısmında Content-Type var mı bunu kontrolün gerçekleştirdik.

Buüstü de bu seferde tamamladım.

NETAS

17. GÜN / 12 Temmuz 2018

Bugün istek atacağınız adresten ögren cevapların hepsi'ni kontrol ettim. İlk olarak dönmesci beklediğimiz ve istekte bulunduğundan adres su şekilde.

Path → GET -{{url}}/cpaas/directory/v1/{{testUser}}/default
 Response; {
 "requestError": {
 "policyException": {
 "messageId": "POL1009",
 "text": "User has not been provisioned for %1",
 "variables": [
 "directory"] } } }

Bu dönerisi genelken cevabı su şekilde kontrol ettim.

```
var jsonData = pm.response.json();
var messageId = jsonData.requestError.policyException.messageId;
var text = jsonData.requestError.policyException.text;
var variables = jsonData.requestError.policyException.variables;

pm.test("Check for messageId", function() {
    pm.expect(messageId).to.eq("POL1009");
});

pm.test("Check for text", function() {
    pm.expect(text).to.eq("User has not been provisioned for %1");
});

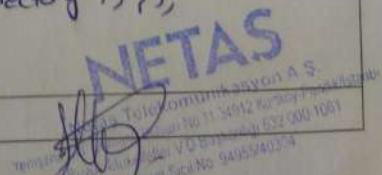
pm.test("Check for variables", function() {
    pm.expect(variables[0]).to.eq("directory");
});

pm.test("Status code is 401", function() {
    pm.response.to.have.status(401);
});
```

Tarih ve İşyeri Amirinin İmzası

12.07.2018

Fırat Üniversitesi Mühendislik Fakültesi



18. BÜN / 13 Temmuz 2018

Cırsamba günü basıldığım attığımız odresten dönen cevapları kontrol etme işlemini sürdürdüm.

Postman'de collection haline getirdiğim tüm test caseler için bu işlemler devam edecekti.

Dünde yaptığı gibi farklı caseler için kontrol tertibatı yazdım. Farklı olarak dönenin beklediğimiz liste objesini tek bir teste yazdım.

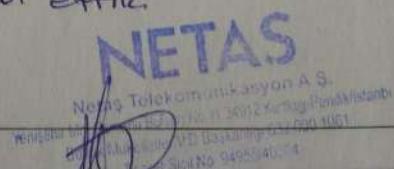
Dönenin beklediğimiz liste → "variables": [

- "path",
- "kodlu",
- "Unknown directory identifier"]

Bu liste için yazmış olduğum metod şu şekilde;

```
pm.test ("Check for variables", function () {
    pm.expect (variables [0]). to.eq ("path");
    pm.expect (variables [1]). to.eq ("kodlu");
    pm.expect (variables [2]). to.eq ("Unknown directory identifier");
});
```

Şebekide yazanız response'daki dönen listesi; tele bir metod içinde tüm indexlerini kontrol ettili.



Tarih ve İşyeri Amirinin İmzası

:13..07.2018

19. GÜN / 16 Temmuz 2018

Bu gün ekibe biraz kafa dopitnak adına, hattası
gözel bozukluğunuyle, parantesi sendromunu üstünden
oturmak adına Sile/İnektumla geziye gitti.

Burada ekibim ile olağan çok kaynaktık ve birlikte
gözel bir ukit peñirdeki Bırıtlıkları olağanlığı tondıktı.

Bu gün bu şekilde bitti.

NETAS
NETAS Telekomünikasyon A.Ş.
T.C. 4432 Nömreli Posta İstasyonu
34340 İSTANBUL - TURKEY

20. GÜN / 17 Temmuz 2018

Bu gün geçen hafta kaldığımı yeden yani Directory Servisimizden belirli pathlere gönderilen isteklerin dönen response'larını, status kodlarını kontrol etmeye devam ettim.

Farklı pathler aranakasitli filtreleme seçenekleri yani örneğin {{url}}/cpaas/directoy/ut/{{user}}/{{directory}}/{d33}/{tartış} gibi bir pathe query param adımsız şekilde istek atılmış caselerini zwróci. Buna göre responselarında bu filtrelemelerin dönmesi şerekiyordu.

Bunun kontrolünü yapmakla ugrastım. Buğunda de bu şekilde sonlandırdım.

NETAS
Telekomünikasyon A.S.
İzmir İletişim İhale İdaresi
0232 362 1651

21. GÜN / 18 Temmuz 2018

Bu gün dönen cevaplarda objelerimizin içindeki değişkenlerin konsantolarının cevaplarının null mi? undefined yani tanımlanmadı mı? yoksa boş mu birdaklınır bunun nasıl kontrolünü yaparız tarzında bir problem değildi. Bana da bu problemi çözmem için yol gösterildi.

İlk olarak öncelik objenizle başlıyorum.

var person = {

"name": "Furuk",
"surname": "GENÇ",
"age": 5 }

Bunu gibi bir objeyi kontrol etmek oda yazdığını fonksiyon su şekilde;

```
function checkData (person) {
    for (var key in person) {
        if (person[key] == null || person[key] == "" || person[key] == undefined)
            return false;
    }
    return true;
}
```

Fonksiyon çağırıldığında → checkData(person);

person içinde bulunan tüm keyleri yani (name,surname,age) genep konsantolarının person[key] şeklinde for Each döngüsünde kontrol edecektir, eğer dolayla true değilse false denecektir.

Tarih ve İşyeri Amirinin İmzası

18.07.2018

NÜSAYİ İŞYERİ AMİRİ
NETAS
Fırat Üniversitesi Mühendislik Fakültesi
Bülent Ecevit Mah. 3212 Kırıkkale
0462 223 000 1067

Fırat Üniversitesi Mühendislik Fakültesi

22. GÜN / 19 Temmuz 2018

Bu gün den kaldığım yerden, objenizin içini kontrol eden fonksiyon geliştirmeye çalıştım. Çünkü bize o kadar basit bir obje dönmeyecekti. Objeniz dizî şeklinde olduğunu varsayıp ona göre bir fonksiyon yazdım.

Örnek data ;

```
var persons = [
  { "name": "Feruk",
    "surname": "Genç" },
  { "name": "Dincer",
    "surname": "Genç" }
]
```

Bunu kontrol eden fonksiyon su şekilde yazdım;

```
function checkDataArray (persons) {
  for (i=0; i<data.length(); i++) {
    for (var key in persons[i]) {
      console.log ("index :" + i + " lain " + key);
      if (persons[i][key] == null || persons[i][key] == "" || persons[i][key] == undefined) {
        console.log ("Hata olor → index :" + i + " key " + key);
        return false;
      }
    }
  }
  return true;
}
```

checkDataArray (persons); fonksiyonunu çağırduğumda bize true döndürür çünkü tüm değerler dolu.

Bu anıda bu şekilde tamamladım.

NETAS

Tarih ve İşyeri Amirinin İmzası

19.07.2018

Fırat Üniversitesi Mühendislik Fakültesi

Nefis Teknoloji İmalat A.Ş.
T.C. OSB İmza No: 11-30124-000-00000000
Buyuk Marmara V.U Başvuru No: 032 000 1001
Ticaret Sicil No: 9905-4034

23. GÜN / 20 TEMMUZ 2018

Bugün 2 gündür üzerinde çalıştığımız objemizin key bölmelerine karşılık gelen kısımları kontrol eden fonksiyon birim testimizde data boyut bir yapısı için data boyutu for döngülerinin olduğu bir metodu entegre ettim ve başarılı bir şekilde çalıştı.

Bu metodu buraya yazmadığımız önceki konuların zaten içindeydi. Bu metodu geliştirdikten if kontrolü bölümünde keyfeder içinde hâldeki ve operatora (II) kullanımları data nesneleri olduğunu kör bildik. Çünkü bu şekilde data hâlini bir sonus alocat performans koruyacak şekilde Kodu refactor ettik. Son olarak testini şu şekilde yazdım

```
pm. test ("Check Array", function () {
    pm.expect (checkArray (directoryItem)).to.eql(true);
});
```

Eğer objemizin tüm keyleri' istedigimiz gibiye sonus true alocetur birde test kısmında cevap true mu? Onu kontrol ettik.

Bugünden bu şekilde devam edeceğiz.

NETAS
Netas Communication A.S.
Telekomünikasyon ve İletişim Pazarlama

24. GÜN / 23 Temmuz 2018

Bugün geçen hafta üzerinde çalışmalar yürüttüğüm Directory Servisimın Postman Collectionları artık hazır hale geldi. Bu collection'ı export edip staf sorumluma teslim ettim. Sadece bu collection'ı değil aynı zamanda environment olarak tanımladığımız geliştiricilerimizin tutulduğu kırımda teslim ettim.

Ardından bu yapıyı online test yönetimi isimli "testlodger" adresine tedarim. Buradaki gerekli ayarları konfigürasyonları yaptırmıştım.

Üzerimdeki bu şırdan sonra baska bir servis için bu şırdan de yapıtlarımın hepsi hepsi aynı formatta genetikini öğrendim. Bu servisin ismi "Address Book".

Excel dosyası tarafından şırdanıldı, çalışmaların içinde itibaren başlayacaktır.

Bugünüm bu şekilde geçti.

NETAS
Takasyon A.S.
www.netas.com.tr

25. SUN 124 Temmuz 2018

Dün benimle paylaşılan yeni Excel dosyasında Address Book servisimize dobr tipki Directory servisimizde olduğu gibi pathler, body'ler, dönenimiz beklediğimiz response'lar bulunuyorlar.

Hemen ilk olank ise bu yopiların hepsini POSTMAN collectionları haline getirmek oldu. Tüm test caselerin içeriğlerini gevurduktan sonra ilgili request atacığınız adreslerin pathlerin adreslerini "Directory" denisinde yoptığım gibi {{url}} yopilâne ekstarenek data katoly konfiye edilme yopisina gâne ayarladık.

Ardından excelde dögnük gelen yepiyi "Success" ve
"Fail" gibi bir klasörleştirmeye yapısına gidenek daha
kolay onasılır olmasını sağladım.

Address Book servisimizde 3 adet olmak üzere (userAccessToken, NoServiceUserAccessToken, invalidAuthUserAccessToken) token olma yeri var. Buntanın sonucunda userAccessToken deşifreleninde herhangi erişebildiğimiz token var. Bu 3 model içinde request test script yeri → Testse kısımı şereflisi environment olarak koydettmenin adına gerekli bir kodlu.

Spannungsabfälle gest.ⁿ

Tarih ve İşyeri Amirinin İmzası

24/07/2018

NETAS
Necatip Teleskomün Kasyon A.S.
Yenizade Mah. 13. Sok. 12. No: 100/1
35121 Konya / TURKEY
Phone: +90 362 300 1651
Fax: +90 362 300 1652
E-mail: info@netas.com.tr

26. GÜN / 25 Temmuz 2018

Bugün kaldığım yerden hizla devam etmek adına çalışma basladım. Token oturum yapınızı hazırladıktan sonra birim Address Book servisimizde tüm contactları grebildiğimiz bir yapının var. Birim istek oturumun odasına ~~den~~ cevapta belti bir yapı döndürmek bekliyoruz ve bunun obratıza edilmeli istiyoruz. Önek olması onaçla döndürülerek belirtilmiş liste/giri zu getirildi;

var contactCollectorCheckers = [

```
{ "attribute": { "buddy": "true",
  "primary Contact": "omer@fontpenc.com",
  "none": "font" } }
```

```
{ "attribute": { "buddy": "true",
  "primary Contact": "font@penc.com",
  "none": "omer" } } ]
```

Bir bu yapıya benzer bir yapının response olarak döndürülüyor. Buran içinde bir fontkayıp yapısının. Bugün ise temel kılın oynamadım Test bölümme su kurmamı ekledim;

```
var user = UTILS.user;
var addressBookId = UTILS.addressBookId;
var authToken = UTILS.QAuthToken;
var response = pm.response.json();
let UTILS = eval(pm.variables.get("loadUTILS"));
```

Tarih ve İşyeri Amirinin İmzası

25.07.2018

Nostaljik Teknolojiler Ltd. Şti. Sayı No: 14912 Konya-Pendik/İzmit
T.C. Ticaret ve İstatistik Kurumu Sayı No: 14912 Konya-Pendik/İzmit
T.C. İdari Mahkeme Sayı No: 14912 Konya-Pendik/İzmit

NETAS

27. GÜN / 26 Temmuz 2018

Dün borsettigim gibi boyan contact'ının kontrol etmek adına function yazdım. Bu fonksiyon esasında aşağıdaki

UTILS.CheckContactCollection (user, addressBookId, response, contactCollectionMocks)

Bu yorum UTILS classı içinde yer almış checkContactCollection'ı göstermek.

```
UTILS.CheckContactCollection = function (coll, ContactColl(user, ad, resp, ccc)) {
    pm.test("Check contactCollection response", function () {
        var length = response.ContactCollection.contact.length;
        pm.test("Check response size", function () {
            pm.expect(length).to.eq(ccc.length);
        });
        for (i=0; i<length; i++) {
            if (ccc[i].list != null && ccc[i].list[0] != undefined) {
                UTILS.CheckContact(user, ad, ccc[i].list[0].list[0]);
            } else {
                UTILS.CheckContact(user, ad, null, response.ContactCollection[i]);
            }
        }
    });
}
```

UTILS.CheckContactCollection ResourceURL (user, resourceURL)

});

Bu şekilde tabii olarak bir yapıyı hazırladık.

Tarih ve İşyeri Amirinin İmzası:

26.07.2018

Fırat Üniversitesi Mühendislik Fakültesi

Netas İletişim Teknolojileri A.Ş.

Genel İletişim ve Satış Ofisi: No: 11-36932 Kocatepe-İstanbul

Buyuk Mekaneller YD Mahallesi 9112 000 1051

Tel: 0312 442 5147, 4

NETAS

28. GÜN / 27 TEMMUZ 2018

Bugün, gün boyunca olduğum reporablesi: funkisyon
sayı sorumlum ile birlikte daha uygun ve enesibillir.
hale getirdik. O funkisyonun içerisinde oprice bir kağız
tane daha farklı vardı. Onları da toplayadık.

Bütün bu istemeler bittiğinden sonra tüm sistemde
bu yapıları uyguladık.

Adress book sentrimiz bir kumur, sayı sorumlum
bir kumur ve ben yaptım. Sonra toplayanın bu
collectionlarını hepsini online test yapan portalının
den "testloft" adını aldık.

(Şekil 1.4)
Bu haftaya bu şekilde gelsende toplayadı. Artık
haftaya süpriz bir konuya başlayacağım.

NETAS
NETAS Elektronik A.Ş.

29. GÜN / 30 TEMMUZ 2018

Bugün yeni bir teknoloji: Öğrenmeye başladım kendilerine kılıncağımız bir yapısı olduğundan benim de öğrenmem iyi olacaktı. Bu teknolojinin ismi "Docker".

Docker, dünyada en çok kullanılan yazılım konteynırlaştırma platformudur. Linux Containers (LXC) yapısı üzerine kurulmuş bir teknolojidir. Containerler içerisinde aynı işletim sistemi tarafından çalıştırılan processlere LXC tarafının da işletim sisteminde sadece kendisi çalıştırılmış gibi düşünülmektedir. Sağlayıcı sorumluluğu ortası kurulmuştur.

Tüm Container'lar aynı işletim sistemi üzerinde çalışmasına rağmen birbirlerinden tamamen isolate edilmişlerdir ve gerekli işlemler yapılmadığı sürece birbirleriyle iletişim kuramazlar. Günümüzde veri merkezleri büyük oranda hypervisor'lar tarafından sunulmaktadır. Cloud (bulut) olanda adlandırılan kavramı alt yapısını oluşturan ve merkezlerin tarama ile hypervisor tarafından yönetilmektedir. Bu sayede ihtiyac olgasunda farklı işletim sistemleri aynı sunucuda kurulabilen Docker ise tek bir işletim sistemi üzerinde birbirinden bağımsız uygulamaları çalıştırabilen Linux'in hypervisorle羁ne ilişkisi en büyük avantaj aynı işletim sistemi üzerinde sunulmasıdır. Hypervisor bu sunuculara kendine alt Guest işletim sistemi bulunuş genetikidir. Linux'de ise container'lar host'un yanı one makinanın işletim sistemini kullanırlar.

(Set 11 1.5) NETAS

30. GÜN / 31. Temmuz 2018

Bu gün, dün kaldığım yerden devam ettim Container teknolojisi hizmet杆菌 ile tasarım sağlar. Her bir işletim sistemi için ayrı ayrı disk alanı, RAM, core gerektikten LXc'de bu gerekmeyecektir. Baktır da kafayı yapmakta Doker ise LXc'ın üzerinde gelişmiş ve herkesin kullanabileceği bir forma dönüştür halleden En önemli özellikle container'in yapısını bir tek bir metin dosyaları "Image" formatı ile tanımlayabilmesidir. Bu sayede yaratılmışının container'da bir resme kaynak (doker yapıyı) bireklerinin kullanımmasını vesile oluşturur.

Docker-ide kan ha keyword'ün temmamı uppnač isterim.

Docker Engine: Sanal lastiklerin yopildiği, VMware/HyperV'ye benzer yapılarla korsanlık gelen kısımdır. (Docker daemon docker mesajlarını)

run C: Dacterin Lxider sonra pekti yoptigi, unu gevreklerine oraci.

Docker CLI: Docker engine üzerinde komutları konstruktörünü sehpası içinde.

Dockerfile: Bir docker uygulaması oluşturmak için gerekli dosya.
Bu dosya içinde uygulamanın neler yaptığını görebiliriz

Images : Bir imgz dosyasi dir. Dockerfile'in build edilmesiyle olusur.

Container : Bir İmge için calzor örneğidir

Docker Registry / Hub : Docker registry / hub sayesinde farklı anahtarların oluşturulması, image'ler incelenebilir, kullanılabılır. Burası maven reposu / github reposu gibi bir kendi oluşturduğumuz image'lerde güncellenebiliriz. Bunu bu şekilde ~~tanıyaladık~~ NETAS.

Tarih ve İşyeri Amirinin İmzası

31.03.2018

[Signature]
Netas Telok Intan Sdn Bhd
No. 10, Jalan 15/100, Bandar Baru Intan, 31400, Perak Darul
Sultana, Malaysia. Tel: +60 52 400 1001

31. GÜN / 1 Ağustos 2018

Dün yaptığım arastırmaların ardından Image ve Container'ın biraz daha detaylı inceledim.

Docker Images: Bir kontenat ya da tomurcuk, calisan bir sistem degildir deyebiliriz. Herhangi bir Image'in yapısına bakarak bilgi sahibi olabiliyoruz. Docker Image'ı baska bir Image'den türedebilir. Her docker Image'ının tek bir sorumluluğu olması. Bu sayede bireysel, scale edilebilen, ihtiyaca göre boyutunu büyütülebilir sistemler oluşturabilir.

Containers: Container'lar Image'lerin calışan kopileridir. Image'ler tek başına bir ortam ifade etmeler. Her docker Image'ı docker engine ile başlatır ve ortak container gibi ve calısan bir sistem ifade eder.

Docker kullanımları nasıl?

- Mikroservisler yapmak çok uygun olduğundan dolayı, dolaylı kullanılır.
- Geliştirme ortamında ait olan paceti degistirerek yapmakta testte ya da gerçek ortamda kullanırız.

Docker'ı kimler kullanıyor?

Google her hafta 2 milyar container başlatıyor. Bunu dışında Netflix, Spotify, Yandex, Amazon gibi şirketlerde aktif olarak kullanıyor.

Docker teknolojisini kullanmak ve önderlik yapmak içine bulut servisi olan Digital Ocean üzerinden bir sunucu alıyorum. Bu sunuya bağlanıp su komutlu docker'ı kurdum.

→ \$ sudo apt-get install docker-ce

Docker kurulumu bu kadar basittir, yarında işte de catkocap12

32. GÜN / 2 Ağustos 2018

Bugün, din üzerinde Docker'ın sunucusu docker'i kuruyorum.

Bugünde öncelik olmasa onaçıyla sunucuma nginx web servisi'ni Docker'ıne.

→ docker run -d -p 8080:80 nginx

Bu komutla eger nginx docker önceden indirilmemişse indirir ve bu docker'ı run komutıyla container haline getirip çalıştırır. -d parametresiyle oturum planında çalışması sağlanır. -p parametresiyle hostun 8080 portuna gelen istekler container'ın 80 portuna yönlendirilir.

→ docker exec -it <container-id> /bin/bash

Bu komut ile nginx'in container'ın id'sini verecek şekilde kullanılır.

→ docker build -t fofuclu/nginx .

Komutu ise "docker file"ının olduğu dizinde çalıştırılıp işe yeterlidir.

Docker file'ının yapısına bakalım.

RUN: build sırasında kurulması istenilen komut birabir iliştilir.

CMD: Image'in çalışarak default komutunu belirtmenize yarar.

ENTRYPOINT: Çalıştırılabilir bir dosya gibi kullanımmasına, parametre izin verir.

NOT: CMD ile ENTRYPOINT birbirine konfliktliliklerden birlikte kullanılabilir. Default ve parametrel olarak kullanılabilir.

EXPOSE: Bu komut uygulamanın hangi portları servis edileceğini belirler.

ADD: Image'mizdeki dosyaların ya da internetten bir dosya eklenmesi yarar.

WORKDIR: Bu komut kendinden sonra gelen CMD, ADD gibi komutları ettirir.

Bu komutların başına otomatik eklenir.

Rüyamı bu şekilde tamamladım.

Tarih ve İyeni Amirinin İmzası:

02.08.2018

NETAS

Fırat Üniversitesi Mühendislik Fakültesi

Netas Tel: 0324 230 00 00
Faks: 0324 230 00 01
E-mail: netas@firat.edu.tr
Ticaret Selc No: 74895/04/274

33. GÜN / 3 Agustos 2018

Biyün baştt bir Spring Boot ile "Hello World" uygulaması
yazonda bunu "mun clean install" komutuyle build edip
derleyip "jar" dosyası haline getirdim. Daha sonra bu
uygulamayı sunucumda göstermek üzere Dockerfile dosyası
hazırladım dockerize ettim. Hazırladığım Dockerfile aşağıdaki;

```
From openjdk:8
ADD target/app.jar app.jar
EXPOSE 8080
```

```
ENTRYPOINT ["java", "-jar", "app.jar"]
```

Dockerfile'ımızı bu şekilde hazırladıktan sonra bulunduğu dizine
giderek → "docker build -t front/app . " komutu ile
bu dockerfile'i okuyup yeni image oluşturdu. Sonra run
komutu ile bunu sunucu üzerinde çalıştırıyorum.

Tabii ki uygulamalar tek silinden okunmaz. Birbütigle ilişkili
olar (veritabanı sunucusu, cache sunucusu, web sunucusu) servisler
yani containerlar "docker-compose.yml" adlı dosyada tanımlıysa
tek bir komutla (up, down vs) ile ayaga kaldırma, durdurma,
birbütigle başlatır ve silerler. Öncelik compose file;

version: '2'

services:

web1:

image: nginx:latest

ports:

- "8001:80"

web2:

image: nginx

ports:

- "8002:80"

Tarih ve İşyeri Amirinin imzası

03.08.2018

Netas Teknolojileri İŞLETİMİS

Yerel Yönetim Kuruluşu İD: 17 NO 02 600 000 1063

Büyük İhracatçı İd: 17 NO 02 600 000 1063

Mühendislik Fakültesi

T.C. 17 NO 02 600 000 1063

34. GÜN / 6 Ağustos 2018

Bugün, Spring Boot ile basit bir crud (create, read, update, delete) istekleri yapabileceğim, REST API geliştirmek adına çalışmaya başladım. Öncelikle temel den Spring nedir? ile başladım. Spring, Java uygulamaları geliştirmek için kullandığımız bir framework'dür. Varsayılan olarak nesneler sadece 1 kez oluşturulur. (Singleton) Temel olarak "IOC" (Inversion of Control) yani bağımlılıkların oluşumu ve yönetiminin developerin üzerinden obr yatkınlığı üzerine kurulmuştur. Bu teknik tıpkı Loose-coupling (zayıf bağımlılık) kavramının (esneklik ve modülerlik petirisidir) olur. Bu teknikin uygulanması için geliştiriciler bir teknik ise DI (Dependency Injection) teknigidir. Temel prensibi, bileşenleri birbirinden bağımsız hale getirip, bağımlılıklarını dışarıdan çalışma zamanında (runtime) enjekte etmektir.

Avantajları: Yaptırılmış kod ortamı, uygulama konfigasyonunu basitleştirerek bağımlılıkları tek bir reposadan yönetmek, test edilebilirliği geliştirmeye yardımcı, uygulamayı esnek, düzgün ve kaliteli bir yapıya dönüştürmek.

IOC teknigini Java'da uygulandı için 2 teknik vardır. 1 → BeanFactory: Factory Design Pattern ile dílusturulmuştur ve en temel DI desteği sağlayarak interface 2. → Application Context: BeanFactory'nin tüm özellikleri barındırır ve ekstra olarak (Transaction, AOP, I18n gibi) özellikleri sağlar. Bean oluşturulmasında iin herhangi metod çağrımları beklerken, objelerin oluşturulmasından ve birbirine bağılmasına da sorumludur. Bz classlarının Spring Bean olarak tanımlanın, Spring Container ise bu surec yöneterek DI görevlerini yapar.

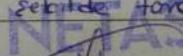
Az da olsa önceden Spring bilgiim vardı. Bugün bu seferde tanımlandı.

Tarih ve İşyeri Amirinin İmzası:

06.08.2018

Fırat Üniversitesi Mühendislik Fakültesi

Notas: Tercih Edilen İmza / A.Ş.
Bölgeve Mh. Onuraltı Bulvarı, 23230, Elazığ
Rıza M. M. 0532 2602 000 1051



35. GÜN / 7 Ağustos 2018

Dün Spring Framework'ü biraz detaylı inceledikten sonra bugun Spring Boot'ıa başlıyım. Spring Boot: Spring tabanlı uygulamaların hızlı ve hızlı şekilde geliştirilmesini sağlayan bir framework'dır. Spring Cloud ve diğer teknolojiler (Redis, Hazelcast, cassandra vb.) gibi yapılandırma ayarlarını kendi içinde bulundurarak geliştiricilerin sadece kod yazmak kadar. Belli başlı bir konu Annotasyon var, onları inceledim.

- ① Spring Boot Application → @Configuration + @EnableAutoConfiguration & @Component
- ② RestController → HTTP isteklerini yakalayan DispatcherServlet bu sınıfın bir denetleyici olduğuunu bildirir.
- ③ RequestMapping → Gelen request'in hangi HTTP methodu (GET, POST ..) ile gönderileceği, hangi path ile gönderileceğinin söyledigiinin Anotasyonudur. Bunu yerine @GetMapping, @PostMapping, @PutMapping, @DeleteMapping kullanabiliriz.
- ④ RequestParam → Request ile gelen parametre bilgisini, hangi değişkende geleceğinin bilgisini verdigini Anotasyondur.
- ⑤ Path Variable → Bu Anotasyon bire URL'deki değişkenin sağlanmasını sağlar. En temel özellikleri → Kendi başına çalışabilen Spring uygulamaları oluşturabiliriz. Jar dosyaları için deprived Tomcat desteği vardır. Yapılandırmaları otomatik halleder. XML yapılandırmasına yerle kalmaz.
- Spring Boot Yapılandırma Ayarları → application.properties ya da application.yml dosyalarında uygulanamamış yapılandırma ayarları yapılandırılabilir. Buna Tomcat port, RabbitMQ, kafka gibi ayarlar olabiliyor.
- Spring Boot web sevkoları için Freemarker, Mustache, Thymeleaf gibi sevkolarını da destekler.

Bugündü bu şekilde təmənlədim.

Tarih ve İşyeri Amirinin İmzası

07.08.2018

NETAS

Netas İşyeri Amirinin İmzası
Universitə Mühəndislik Fakültesi
Engemu 15, Dənizli, 22051, Turkiyə
Birlik, İşsizlik, VD Başkanlığı: 632 000 1951
Şirkət İmzalama №: 945544024

36. GÜN / 8 Ağustos 2018

Bu gün Proje'ye başlamak için her şey hazır. Sadece genel kodlar ve temel yapısı nasıl olacak konusu belirlenmesi kaldı. Kullanıcılarını teknajiler şu şekilde: Spring Boot + Spring Data JPA + HATEOAS + MySQL olacak. Projenin temel tanımı şu şekilde. Biz istek yaparak veritabanına hem Person hemde Department ekleme sonrasında da bu Personları belirli departmentlere atamak istiyoruz. Bu iki model arasında da @OneToOne ve @ManyToOne ilişkisi olsun istiyoruz. Ayrıca Person, departmenti ve atadığımız departmeni silme, güncelleme gibi özellikleri de sorumlu olmak istiyoruz. Projenin temel yapısında domain, exception, repository, service, controller ve resource katmanları olacak. Bu sayede daha düzenli ve kapsamlı bir sistem inşa edeceğiz.

(Şekil 1.6)

Şekildeki gibi bir yapı oluşturduk. Ve ilk olarak Model yani domain katmanından başladım. Burada Person ve Department adında 2 adet modelimiz olacak. Buna, bir Entity haline getirip veritabanımızda tablo oluşturduğum uygun hale getirdim. Person modelimizde Department modeli ile ManyToOne ilişkisi kurдум. 1'den fazla Person aynı department olabilir. Department modelinde ise Person objemize OneToMany ilişkisi kurдум. Bu sayede 1 departmentde 1'den fazla Person olabiliyor管理体制 kurдум.

(Şekil 1.7)

Tarih ve İşyeri Amirinin İmzası

08.08.2018

NETAS

Netas Teknolojileri A.Ş.

Yenigün Mah. 11. Sok. 11/11 D:1134012 Kırıkkale/Pendik/İstanbul

37. GÜN / 9 Ağustos 2018

Bugün dün kaldığım yerden devam ettim. Modellemimi değiştirdim. Buna göre veritabanı boyutu ikişin SQL yorumu yerine Spring Data JPA kullanarak metodlar aracılığı ile crud işlemleri yapabileceğimiz repository'yi değiştirdim.

(Şekil 1.8)

Daha sonra bu metodları kullanacağımız olsun istemiz yani Business Logic katmanımıza Service bölümünde geçtim. Person Service, Department Service, Assignment Service classları yarattım. Person Service'de bir Person'i ekleme, silme, güncelleme, görüntüleme gibi bir iş oluşturduk. Anna bir veritabanında bulunan Person modelimizde tüm bilgilerini geri döndürmek istenediğinden resource katmanında geri döndürmeni istedigimiz bilgileri vererek kurduk. Bu işlemi hem Person hem Department hende Assignment servisimiz içten gerçekleştirdik.

(Şekil 1.9, Şekil 2.0)

Resimlerde de gördüğünüz üzere böyle bir yapı oluşturduk ve exception kisimlarını kurup buna bu sayede hatalı bir işlem yapıldığında bize onları bir mesaj dönerek ve daha iyi bir sisteme sahip olacaktık.

(Şekil 2.1)

Bugünü bu şekilde tamamladım.

Tarih ve İşyeri Amirinin İmzası

09.08.2018

NETAS

Netas İletişim Teknolojileri A.Ş.

Venüs Mah. 12. Sok. 12/1 D: 1001/1002/1003/1004

Evinin Mah. 12. Sok. 12/1 D: 1001/1002/1003/1004

Ticaret Sıra No: 94955/1002/4

Fırat Üniversitesi Mühendislik Fakültesi

38. GÜN / 10 Ağustos 2018

Bu gün projemin controller kılmasını yarınla bekliyor. Projemin kurulumosunda ve döde doğru bir yapı oluşturmak adına staj sorumlum ve ekibimde bir arkadaşım da yardım alıyor. Bu controller bölümünde Person Controller, Department Controller ve Assignment Controller adında classlar oluşturuldu. Person Controller sınıfında bir endpoint'e yapılırı istek doğrultusunda ne yapacağımı nasıl karsılayacağını belirtmem gerekiyordu. Bu kısımları yazdık. Toplam 5 metod'dan oluşan (add, get, delete, getAll, update) gibi metodlar vardı. Bu kısımları Department Controller içinde yaptım. Geriye kalan Assignment Controller sınıfında ise sadece 1 metod olacak ve bu metod alacağı personlara göre bu personları 1 department'e atayacak. Bunu da yazdık.

(Şekil 2.2)

Şekil de görüldüğü gibi tim kısımları oluşturmuş ve artic Person ekleyip, güncelleştirip, güncelleştirip, silip bilgi isteri Department içinde yapabilen ve Personları belki bir department atayıp veritabanına kaydeden bir REST API'si var.

Proje kodlarının bureau yazmasının yerine github'a oturak göstermesi döde mantıklı bildim.

Tarih ve İşyeri Amirinin İmzası

10.08.2018

NETAS
Netas İkamet Konut İmar ve Turizm Pazarlama A.Ş.
İşyeri İkamet Konut İmar ve Turizm Pazarlama A.Ş.
Büyüklük İkamet Konut İmar ve Turizm Pazarlama A.Ş.
Buyuklik İkamet Konut İmar ve Turizm Pazarlama A.Ş.
Ticaret Sayı: 5115540234

Fırat Üniversitesi Mühendislik Fakültesi

39. GÜN / 13 Ağustos 2018

Bugün, geçen hafta çöpu kısmını tanımlamış olduğumuz projemin bazı kısımlarını ekip arkadaşımın yardımıyla refactor ettik. "AssignmentService" adlı classda tek bir person icin department belirliyorken bu classı yeniden düzenleyerek verdığımız person id'lerden oluşan bir list de tüm personları çapırıp bunbu belli etmemiz doğmuştur departmenta kolayca tek seferde ekleyebilcektik. Bunun eklenmesinin yanında bir de kılavuya göstericeğimiz response düzenlememiz gerekiyordu. Bizim tasarımımız Departmanlar içerisinde farklı Personlar olabiliyor, buna imkan sağlıyor olmuyordu. Response da bunun göstergesi için geringe "Department Resource" dönmeliydiç ancak istek yaparken person id'leri list olarak alacağımızdan göstergimde de departmentin içinde bulunan PersonList için yeni bir obje yarattık. Daha doğrusu var olan nesnemizi List şeklinde perjeye dönüştürmemizi sağlayarak bir bölüm yazdık.

Projenin belli kısımlarını refactor ederken, onlara istenilenleri düzelttik ve penekiz sınıfları sildik.

Bugün bu şekilde tanımladım.

Tarih ve İşyeri Amirinin İmzası

13.08.2018



40. GÜN / 14 Ağustos 2018

Dün ki refactoring isleminden sonra yorma oldigumuz REST API'lin bütük bir ismi seçmemiz oldu. Projemizin çoğu kisim tamamda önceki bir REST API'ni dahi kapsamli bir seviye kullanmak istiyorduk. Bu uygulamamız onaigile "Richardson Maturing Model" adinda bir modelde batmam gerekti. Bu model 4 seviyeden oluşmaktadır.

Level 0 → Genelde POST metodu ile istekler gerçekleştirtilir.

Level 1 → Belirli bir context path'e sahiptir. Örneğin "person/1" gibi bir yol izleyipse ve genelde POST metodu kullanılyorsa REST'i Level 1'de kullanıyoruz denekir.

Level 2 → Metod yapımlarında PUT, DELETE, GET gibi metodlar kullanılıyorsa REST'i seviye 2'de kullanıyoruz.

Level 3 → Bu seviye de herhangi bir URL'ye istek yaptığımızda örenceki bu resource ile yapabileceğimiz islemleri yapabileceğimiz URL içinde istenilen dönsüzlüğü söylekadar. Bu sayede kullanici bir önceki, bir sonraki ya da elindeki bu path ile hangi islemleri yapabileceğine dobr admını yapabiliyor.

Level 2'de olan projemi level 3'e çıkarttım. Burada bir REST mimarisini olan "HATEOAS" kullanılmıştır. Projemde entegre edip "Person Resource Assembler" ve "Department Resource Assembler" classlarını kullanıcımıza tanıttırınım adına "Link" kisimlarını ekledim. Bu sayede projem artık REST mimarisini Level 3'te kullanır bir proje dönsümüş oldu.

Tarih ve İşyeri Amirinin İmzası

14.08.2018

Fırat Üniversitesi Mühendislik Fakültesi

NETAS
Netas Teknolojileri A.S.
T.C. Ticaret Bakanlığı No: 11.32912 Konya Peştemal Mah.
Konak Mah. 10. Sokak No: 10/1051
T.C. İdari Sayı No: 945540324

41. GÜN / 15 Ağustos 2018

Bugün stajimin son günüydü. Sekiz ilk olarak, projemde bir sql sorgusu yazmak üzere Spring Data JPA ve Hibernate dokümantasyonunu okudum. JOIN FETCH ve LEFT JOIN hakkında bazı örnekler yaptım. JPA'de kendiniz elle bir query yazmak istiyorsak, JpaRepository'i kullanıp (extends) class'ta @Query注释ini kullanarak query'yi oluşturabiliriz. Her seyde önce Spring Boot console ekranında hibernate loglarını görmek istiyorum. Bu için "application.properties" dosyasına bir koy tanımları yapıyorum.

Ardından NETAS'in hizmet sunduğu Optus'a gittiğimiz bir gürüm dan "Optus Loop Line" isimli trünen sunumuna katıldık.

Projemde bire peri dönen JSON objesinin içinde null değerlerin genenmesini istemediğimden dolayı "application.properties" dosyasına "spring.jackson.default-property-inclusion=NON_NULL" tanımlaması gerekiyor. Loglama levelini ve projem ayakte iken gelen istekler doğrultusunda akan loglarımı ayrı bir metin bittişine yordurmak adına properties dosyasına bir koy tanımlama daha yapıyorum.

Stajimin son gününde bu şekilde tanımladım

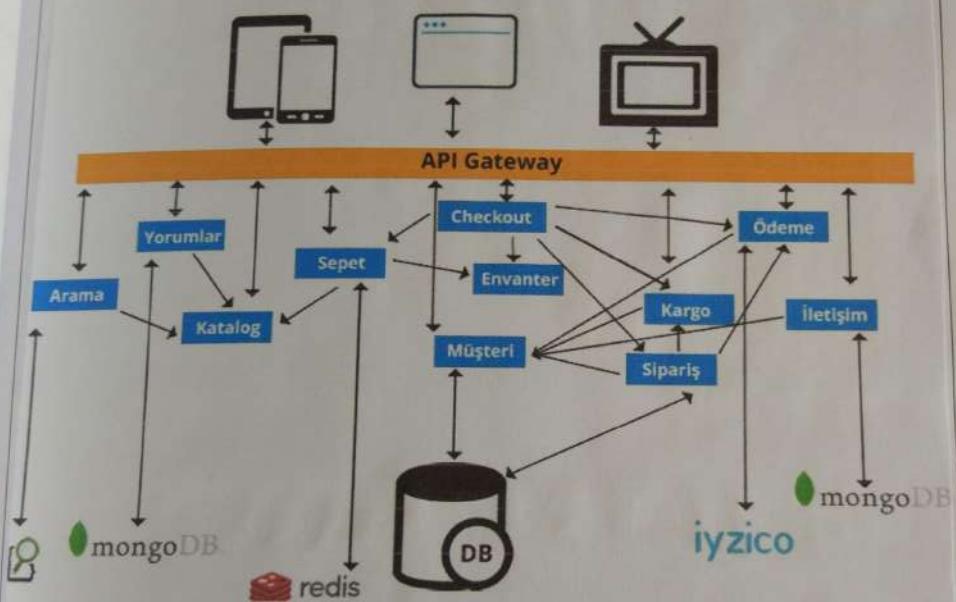
NETAS

Netas Telekomünikasyon A.S.
İzmir İletişim İdaresi İletişim Piyasaları
Bölge Müdürlüğü VD Başkanlığı 332 000 1061
www.netas.com.tr

Tarih ve İşyeri Amirinin İmzası

15.08.2018

Mikroservis Mimarisi



(Şekil 1.1)

NETAS

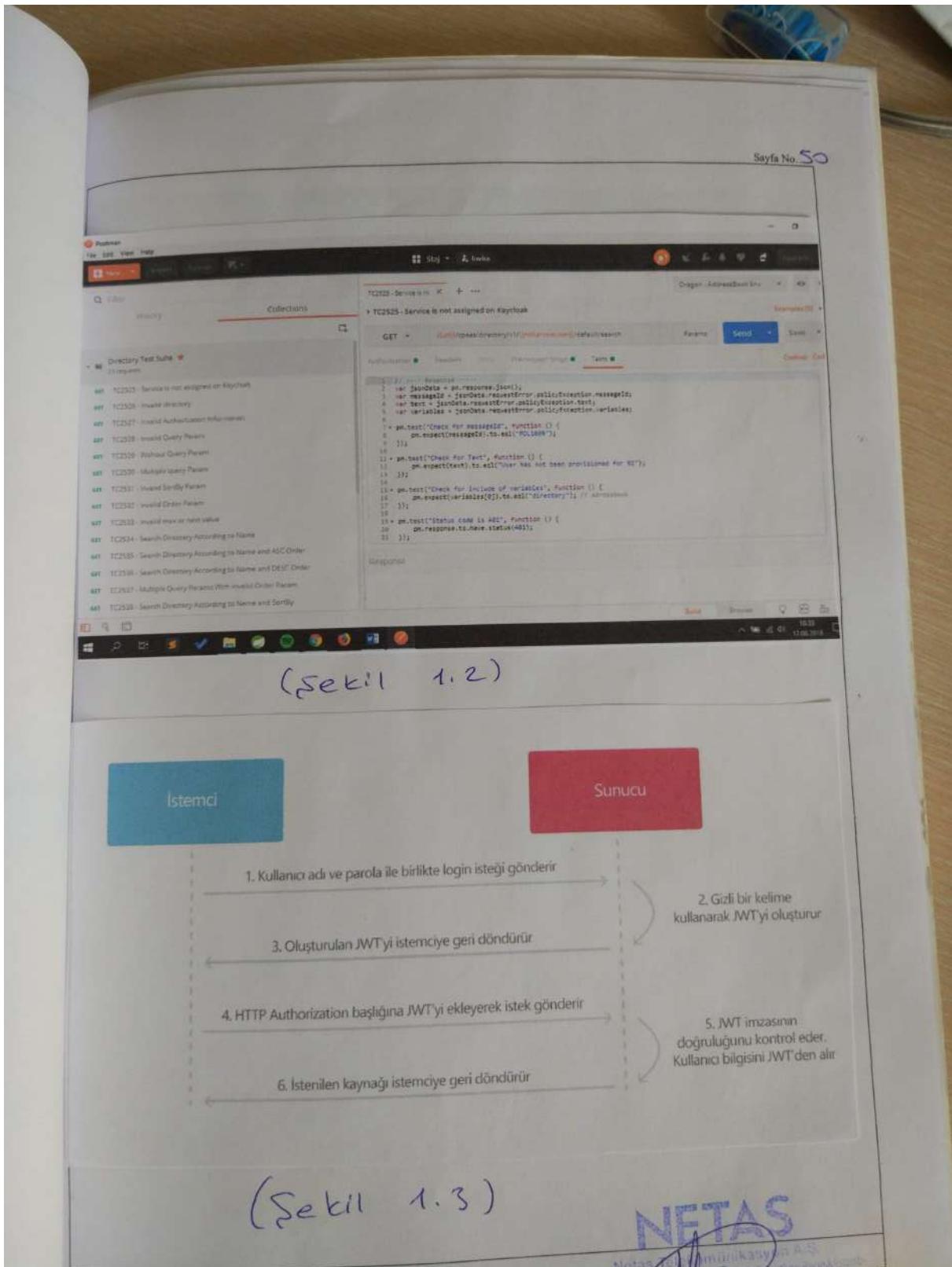
Tarih ve İşyeri Amirinin İmzası

...../...../20...

Netas Telekomünikasyon A.S.

İnterjet İletişim Hizmetleri Ltd. Şti. 113917 Küçükçekmece/İstanbul

Fırat Üniversitesi Mühendislik Fakültesi / İletişim Mühendisliği Bölümü / Başkaanlığı 632 000 1061



Keyclock

TC2755 - Verify that user cannot use addressbook due to addressbook service is not assigned
Last updated on: 2016-Jul-2012, Last tested by: Metin KARAYIL, View test run stats

Test steps
Expected result:

```

    {
        "requestError": {
            "policyException": {
                "messageID": "POL1009",
                "text": "User has not been provisioned for X1",
                "variables": {
                    "addressbook"
                }
            }
        }
    }
  
```

Automation Status: Completed
Type: Positive
Status Introduced: OpenS2.0-Blended-10.17
Module: Service
Component: AddressBook

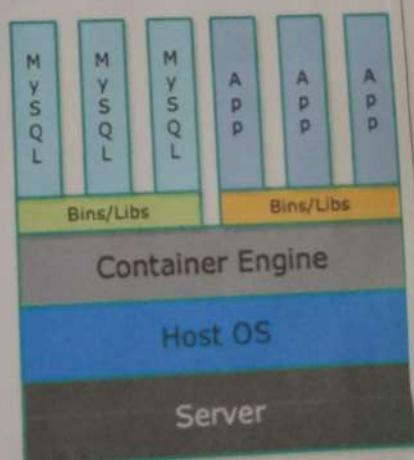
TC2718 - Verify that user cannot reach unknown addressbook
Last updated on: 2016-Jul-2012, Last tested by: Metin KARAYIL, View test run stats

(Şekil 1.4)

Virtual Machines



Containers



(Şekil 1.5)

NETAS
NETAS İNFORMATİON A.Ş.
TURKISH COMPUTER SYSTEMS A.Ş.
www.netas.com.tr
+90 216 555 00 00

```
✓ 5 > crud-person-example [boot] [crud-person-example master]
  ✓ 5 > src/main/java
    ✓ 5 > com.farukgenc
      > 5 CrudPersonExampleApplication.java
    ✓ 5 > com.farukgenc.person.domain
      > 5 Department.java
      > 5 Person.java
    ✓ 5 > com.farukgenc.person.exception
      > 5 DepartmentNotFoundException.java
      > 5 GlobalExceptionHandler.java
      > 5 PersonNotFoundException.java
    ✓ 5 > com.farukgenc.person.repository
      > 5 DepartmentRepository.java
      > 5 PersonRepository.java
    ✓ 5 > com.farukgenc.person.service
      > 5 > AssignmentService.java
      > 5 DepartmentService.java
      > 5 PersonService.java
    ✓ 5 > com.farukgenc.person.web.controller
      > 5 AssignmentController.java
      > 5 DepartmentController.java
      > 5 PersonController.java
    ✓ 5 > com.farukgenc.person.web.resources
      > 5 AssignmentResource.java
      > 5 AssignmentResourceAssembler.java
      > 5 > DepartmentResource.java
      > 5 DepartmentResourceAssembler.java
      > 5 ErrorResource.java
      > 5 PersonResource.java
      > 5 PersonResourceAssembler.java
      > 5 SuccessAssignmentResource.java
      > 5 SuccessResource.java
  > 5 > src/main/resources
```

(Şekil 1.6)

İFTAS

```
1 package com.furkugent.person.domain;
2
3 import java.util.List;
4
5 import javax.persistence.*;
6
7 @Entity(name = "DEPARTMENT")
8 public class Department {
9
10     @ManyToOne(mappedBy = "department", fetch = FetchType.LAZY, cascade = CascadeType.PERSIST)
11     private List<Person> person;
12
13     @Id
14     @GeneratedValue(strategy = GenerationType.IDENTITY)
15     private Long id;
16
17     @Column(name = "DEPARTMENT_ID", unique = true)
18     private String name;
19
20     public Long getId() {
21         return id;
22     }
23
24     public void setId(Long id) {
25         this.id = id;
26     }
27
28     public String getName() {
29         return name;
30     }
31
32     public void setName(String name) {
33         this.name = name;
34     }
35
36     public List<Person> getPerson() {
37         return person;
38     }
39 }
40
41
42 package com.furkugent.person.domain;
43
44 import java.util.List;
45
46 import javax.persistence.*;
47
48 @Entity(name = "PERSON")
49 public class Person {
50
51     @Id
52     @GeneratedValue(strategy = GenerationType.IDENTITY)
53     private Long id;
54
55     @OneToOne(fetch = FetchType.LAZY, referencedColumnName = "ID")
56     @PrimaryKeyJoinColumn(strategy = GenerationType.IDENTITY)
57     private Department department;
58
59     @Column(name = "PERSON_NAME")
60     private String name;
61
62     @Column(name = "PERSON_SURNAME")
63     private String surname;
64
65     public Long getId() {
66         return id;
67     }
68
69     public void setId(Long id) {
70         this.id = id;
71     }
72
73     public String getName() {
74         return name;
75     }
76
77     public void setName(String name) {
78         this.name = name;
79     }
80 }
```

(Sekil 1-7)

(Sekil 1.8)

en Amirinin İmzası: 20.

(Setil 1. 3)

```
PersonResource.java 10
1 package com.farukgenc.person.web.resources;
2
3 import org.springframework.hateoas.ResourceSupport;
4
5 public class PersonResource extends ResourceSupport {
6
7     private Long personId;
8
9     private String name;
10
11    private String surname;
12
13    private DepartmentResource departmentResource;
14
15    public Long getPersonId() {
16        return personId;
17    }
18
19    public void setPersonId(Long personId) {
20        this.personId = personId;
21    }
22
23    public String getName() {
24        return name;
25    }
26
27    public void setName(String name) {
28        this.name = name;
29    }
30
31    public String getSurname() {
32        return surname;
33    }
34
35    public void setSurname(String surname) {
36        this.surname = surname;
37    }
}
DepartmentResource.java 11
1 package com.farukgenc.person.web.resources;
2
3 import org.springframework.hateoas.ResourceSupport;
4
5 public class DepartmentResource extends ResourceSupport {
6
7     private Long departmentId;
8
9     private String departmentName;
10
11    public Long getDepartmentId() {
12        return departmentId;
13    }
14
15    public void setDepartmentId(Long departmentId) {
16        this.departmentId = departmentId;
17    }
18
19    public String getDepartmentName() {
20        return departmentName;
21    }
22
23    public void setDepartmentName(String departmentName) {
24        this.departmentName = departmentName;
25    }
26
27 }

```

(Şekil 2.0)

```
1 PersonNotFoundException.java 22
2 package com.farukgenc.person.exception;
3
4 public class PersonNotFoundException extends Exception {
5
6     private static final long serialVersionUID = -1L;
7
8     private String errorCode;
9     private String description;
10
11     public PersonNotFoundException(String errorCode) {
12         this.errorCode = errorCode;
13         this.description = description;
14     }
15
16     public String getErrorCode() {
17         return errorCode;
18     }
19     public void setErrorCode(String errorCode) {
20         this.errorCode = errorCode;
21     }
22
23     public String getDescription() {
24         return description;
25     }
26     public void setDescription(String description) {
27         this.description = description;
28     }
29
30 }
31
32
33 DepartmentNotFoundException.java 22
34 package com.farukgenc.person.exception;
35
36 public class DepartmentNotFoundException extends Exception {
37
38     private static final long serialVersionUID = 6L;
39
40     private String errorCode;
41     private String description;
42
43     public DepartmentNotFoundException(String errorCode) {
44         this.errorCode = errorCode;
45         this.description = description;
46     }
47
48     public String getErrorCode() {
49         return errorCode;
50     }
51
52     public void setErrorCode(String errorCode) {
53         this.errorCode = errorCode;
54     }
55
56     public String getDescription() {
57         return description;
58     }
59     public void setDescription(String description) {
60         this.description = description;
61     }
62
63 }
64
65
66 GlobalExceptionHandler.java 22
67 package com.farukgenc.person.exception;
68
69 import java.util.List;
70 import javax.servlet.http.HttpServletRequest;
71
72 @ControllerAdvice
73 public class GlobalExceptionHandler {
74
75     @ExceptionHandler(value = PersonNotFoundException.class)
76     public ResponseEntity<ErrorResource> handlePersonNotFoundException(PersonNotFoundException e) throws Exception {
77         return new ResponseEntity<ErrorResource>(e.getErrorResponse(), HttpStatus.NOT_FOUND);
78     }
79
80     @ExceptionHandler(value = DepartmentNotFoundException.class)
81     public ResponseEntity<ErrorResource> handleDepartmentNotFoundException(DepartmentNotFoundException e) throws Exception {
82         return new ResponseEntity<ErrorResource>(e.getErrorResponse(), HttpStatus.NOT_FOUND);
83     }
84
85 }
86
```

(Sekil 2.1)

```
partmentController.java 11
package com.farukgenc.person.web.controller;
import java.util.List;

@RestController
@RequestMapping("/department")
public class DepartmentController {
    private DepartmentService departmentService;
    private DepartmentResourceAssembler departmentResourceAssembler;
    @Autowired
    public DepartmentController(DepartmentService departmentService,
        DepartmentResourceAssembler departmentResource) {
        this.departmentService = departmentService;
        this.departmentResourceAssembler = departmentResourceAssembler;
    }
    @GetMapping
    public ResponseEntity<List<DepartmentResource>> getAll() {
        return ResponseEntity.ok().body(departmentResourceAssembler.list());
    }
    @PostMapping
    public ResponseEntity<DepartmentResource> addDepartment() {
        return ResponseEntity.ok()
            .body(departmentResourceAssembler.toResource(
                departmentResourceAssembler.create()));
    }
    @PutMapping
    public ResponseEntity<DepartmentResource> updateDepartment() {
        return ResponseEntity.ok().body(departmentResourceAssembler.update());
    }
    @DeleteMapping(value = "/{departmentId}")
    public ResponseEntity<DepartmentResource> getDepartment() {
        return ResponseEntity.ok().body(departmentResourceAssembler.get());
    }
}

PersonController.java 22
package com.farukgenc.person.web.controller;
import java.util.List;

@RestController
@RequestMapping("/person")
public class PersonController {
    private PersonService personService;
    private PersonResourceAssembler personResourceAssembler;
    @Autowired
    public PersonController(PersonService personService) {
        this.personService = personService;
        this.personResourceAssembler = personResourceAssembler;
    }
    @GetMapping
    public ResponseEntity<List<PersonResource>> getAll() {
        return ResponseEntity.ok().body(personResourceAssembler.list());
    }
    @GetMapping(value = "/{personId}")
    public ResponseEntity<PersonResource> getPerson() {
        return ResponseEntity.ok().body(personResourceAssembler.get());
    }
    @PostMapping
    public ResponseEntity<PersonResource> addPerson() {
        return ResponseEntity.ok().body(personResourceAssembler.create());
    }
    @PutMapping
    public ResponseEntity<PersonResource> updatePerson() {
        throw new PersonNotFoundException();
        return ResponseEntity.ok().body(personResourceAssembler.update());
    }
}

AssignmentController.java 33
package com.farukgenc.person.web.controller;
import org.springframework.beans.factory.annotation.Autowired;
@RestController
@RequestMapping("/person/addDepartment")
public class AssignmentController {
    private AssignmentService assignments;
    @Autowired
    public AssignmentController(AssignmentService assignments) {
        this.assignments = assignments;
    }
    @PostMapping
    public ResponseEntity<AssignmentResource> successAssignment() {
        return ResponseEntity.ok().body(assignments.create());
    }
}
```

(Sekil 2.2)