**Veritabanı işlemleri**

Projedeki hazır veritabanı bağlantısını kullanmak için MySql kurulu olması gerekmekte ve “toplanti” adında bir shema oluşturmanız gerekmektedir.

Projeyi ilk çalıştırırken application.properties dosyasındaki spring.jpa.hibernate.ddl-auto ayarını “create” olarak çalıştırın.

Mysql ayarlarını kullanıyorsanız ; application.properties’deki spring.datasource.username’e mysql root kullanıcı adını , spring.datasource.password ‘e şifrenizi yazın.

Daha sonraki çalıştırmalarınızda spring.jpa.hibernate.ddl-auto ayarını “update” veya “none” olarak çalıştırınız.  
Boş olarak oluşan veritabanınızda hazır verilerin olmasını istiyorsanız usable\_sqlScript adlı klasörün altındaki scripleri çalışırarak hazır verileri elde edebilirsiniz..

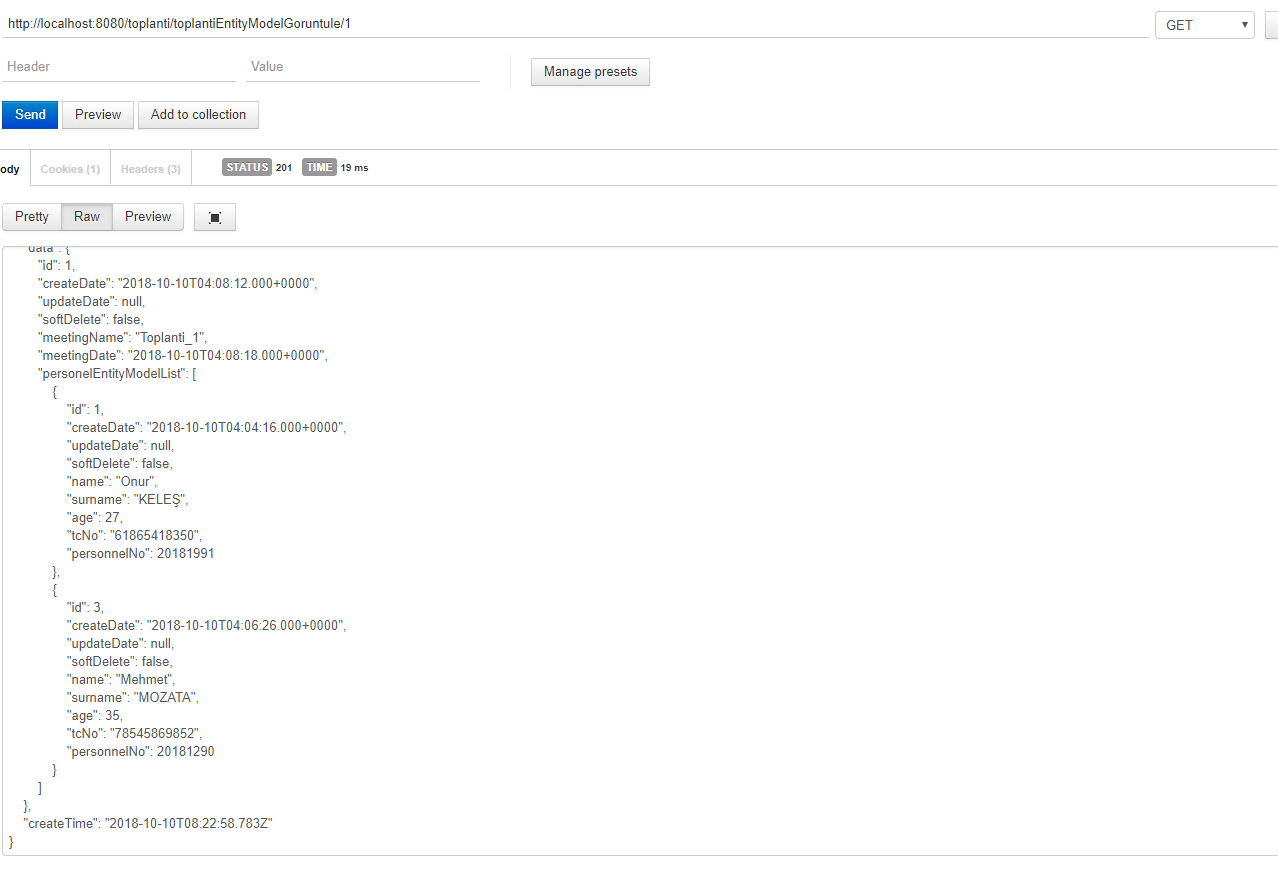
**Projede Kullanılan Özellikler**

**1-View Model**

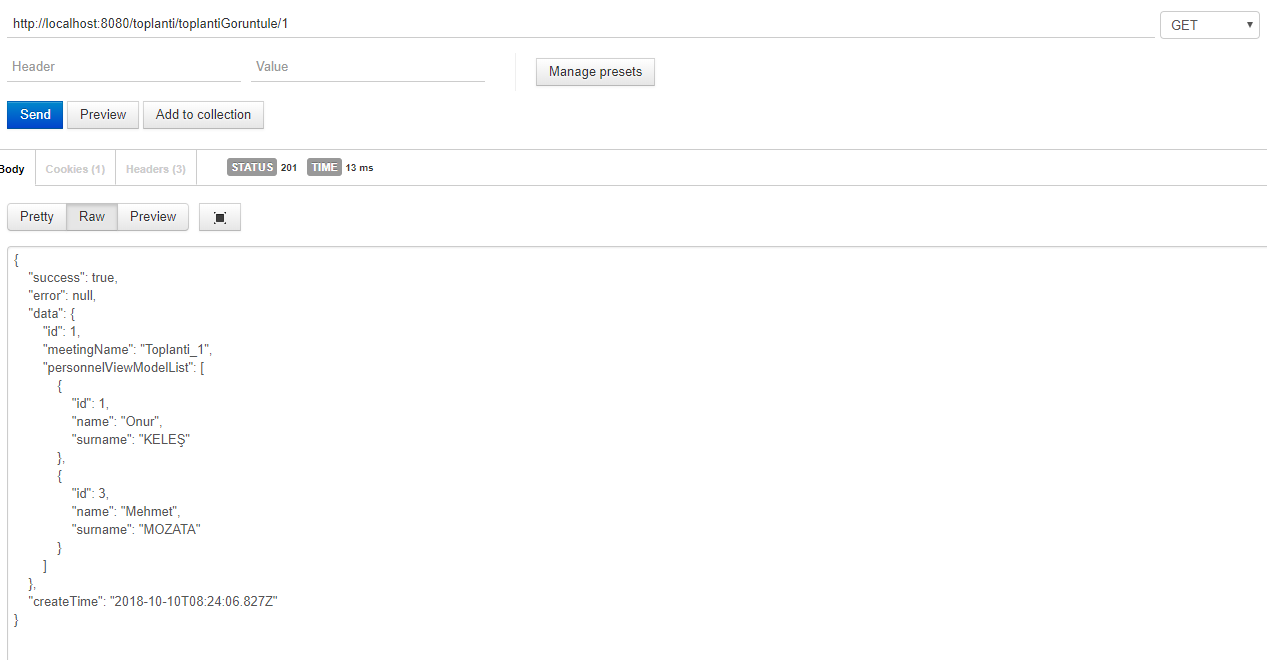
İstemciye gönderilen verilerden tasaruf etmek için, Entity modelden ön yüze gönderilmesi gereksiz olan verilerden arındırılmıs yeni bir model oluşturulması.

Örnek :

İstemciye Gönderilen ToplantıEntityModel;



Ayni verinin View modeli aşağıdaki gibidr;



**2-İstemciye tek tip Veri Gönderme**

Bu işlemi “ViewModel” klasörünün altında yer alan TransactionError ve TransactionResult classları ile yapıyoruz.

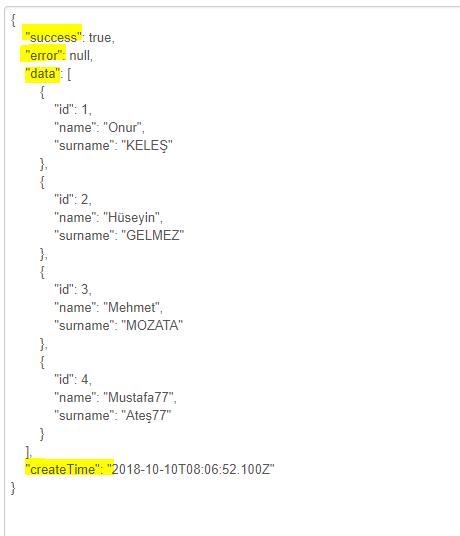
İstemciye giden veriler 4 başlık altında sabit bir şekilde gidiyor.

**sucess** : yapılan işlemin başarılı olup olmadığını belirtiyor.

**data** : Ön yüze gönderilmek istenen verileri kapsıyor.

**error**: Sunucu tarafında hata almış ise alınan hatayı istemciye ulaştırmayı sağlıyor.

**createTime**: İstemciye gönderilen cevabın ne zaman gönderildiğini belirtiyor.



**3-Model dönüştürücülerin Mapper’la yapılması**

İstemci tarafına gönderilen View modeller ,entity modellerin dönüştürücüye girmesi ile oluşturuluyorlar.

Dönüştürücülerde mapper tercih edilmesinin sebebi hem kod kalabalığını azaltıyor, isimleri isimleri aynı olan aynı tipteki verileri otomatik olarak dönüştürebiliyor.

**4-Soft delete**

Verilerin kalıcı olarak silinmesi yerine db ‘de baktığı boolean bir alana göre verilerin gösterilip gösterilmiyeceğini belirliyor.Böylece verilerimizi hard delete’ten kurtarmış ve geri dönüştürülebilir hale getiriyoruz.

**5-Base Repository ve custom query**

Repositoryler JPARepositoryden türetilerek aslında bir çok ihtiyacımızı karşılıyor.Fakat soft delete mantığına geçildiği için findAll vb metodlar soft delete’de silinip silinmediğine bakmadan tüm verileri getiriyor.Bu sebepten dolayı custom querylere ihtiyacımız oluyor.

findAllActive, countActive,findOneActive ... gibi

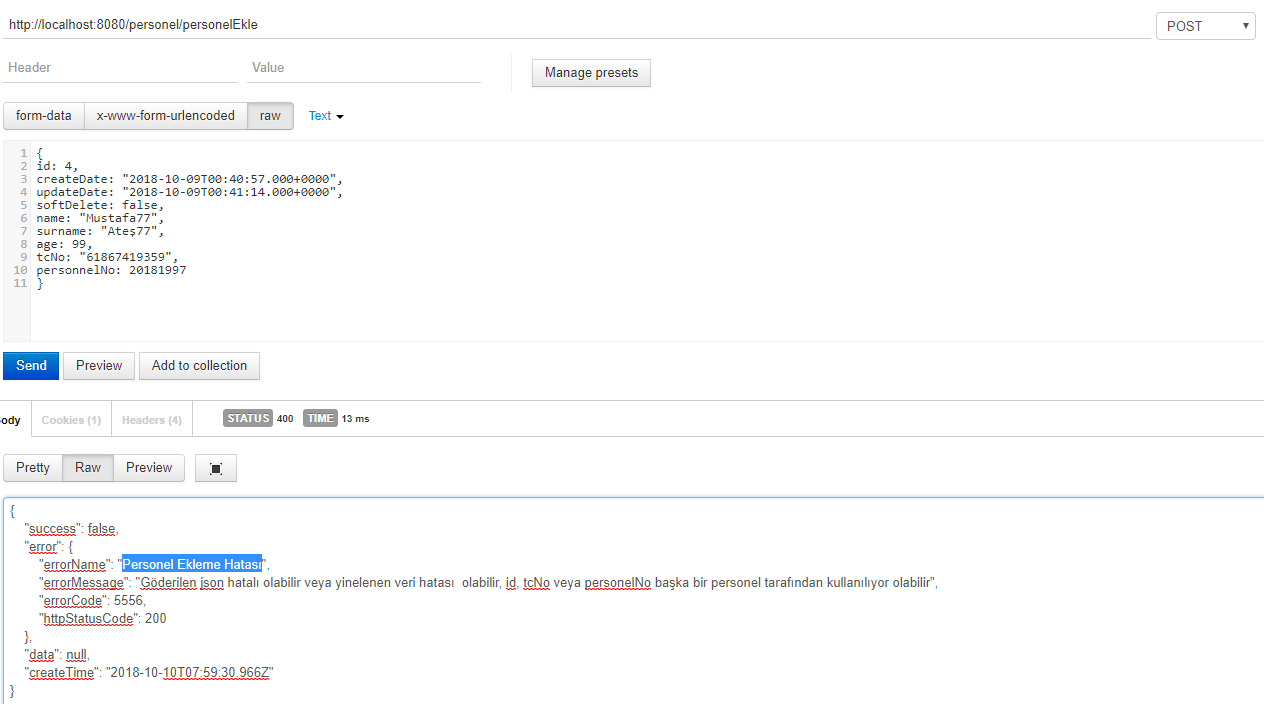
**6-CustomException**

İstemciye daha anlaşılabilir exceptionların gitmesi için hazırlanmıs exceptiondan türetilmiş bir class.

**Örnek:**

CustomException customException = new CustomException("Toplanti Sorgulama Hatası","Göderilen id hatalı olabilir veya böyle bir toplanti kayıtlarımızda bulunmamaktadır.",6555,200);

**Hatanın ekran görüntüsü:**

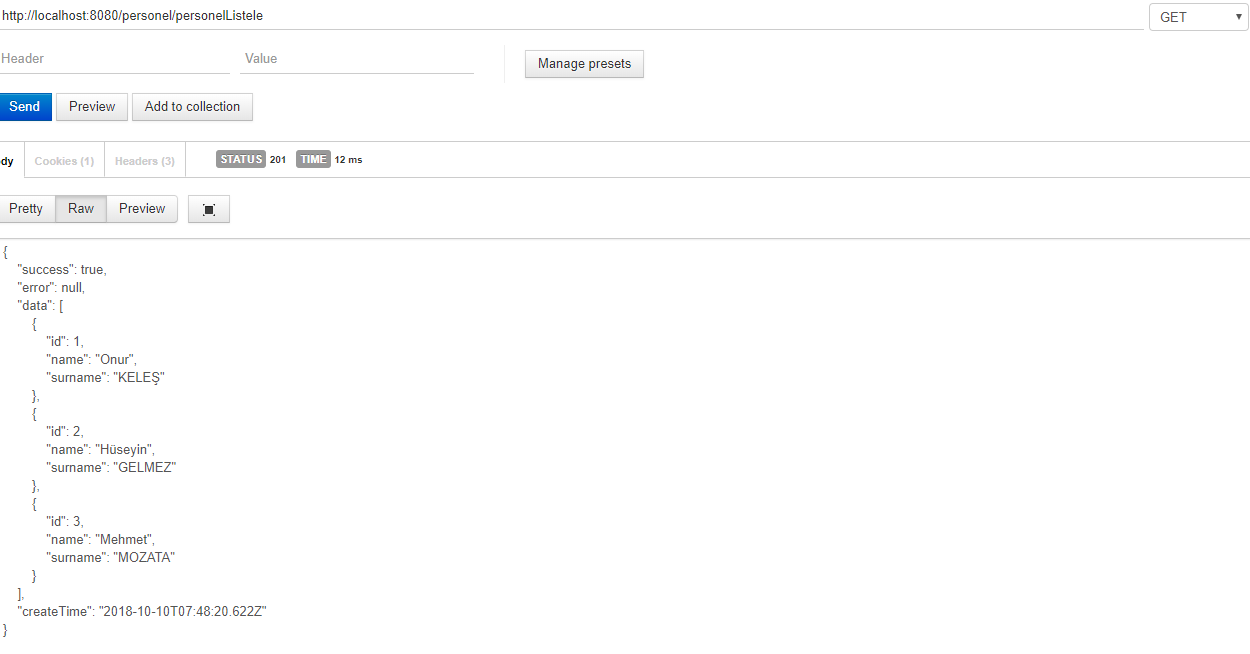


**REST CONTROLLER KULLANMA TALİMATLARI**

**Personel İşlemleri**

**1-Personel Listeleme**

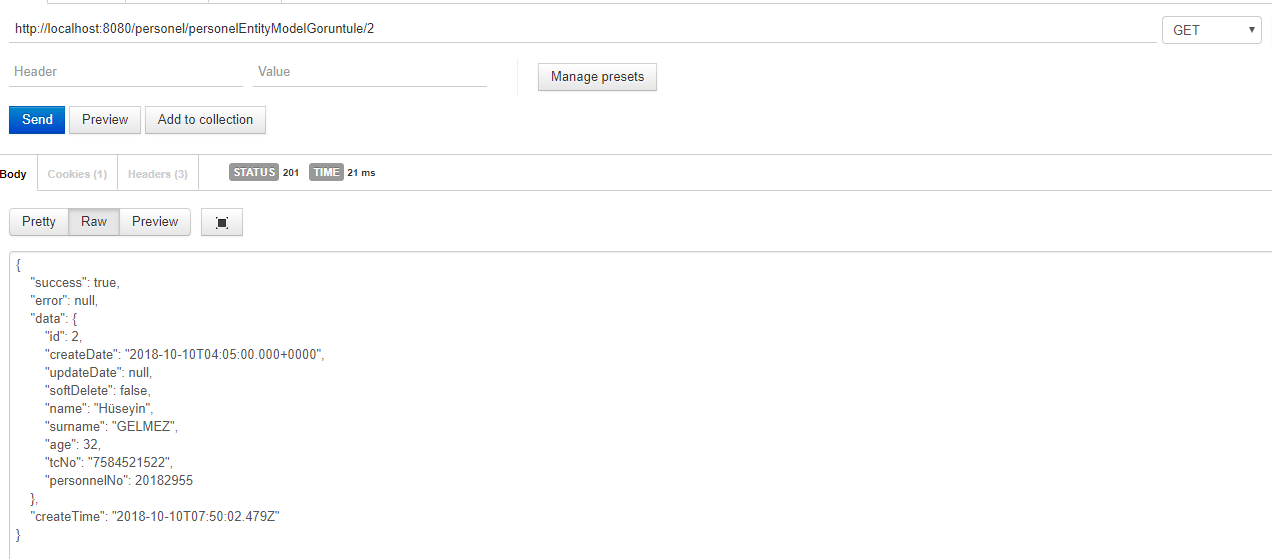
@GetMapping(value = "/personelListele")



**2-Personel Görüntüleme**

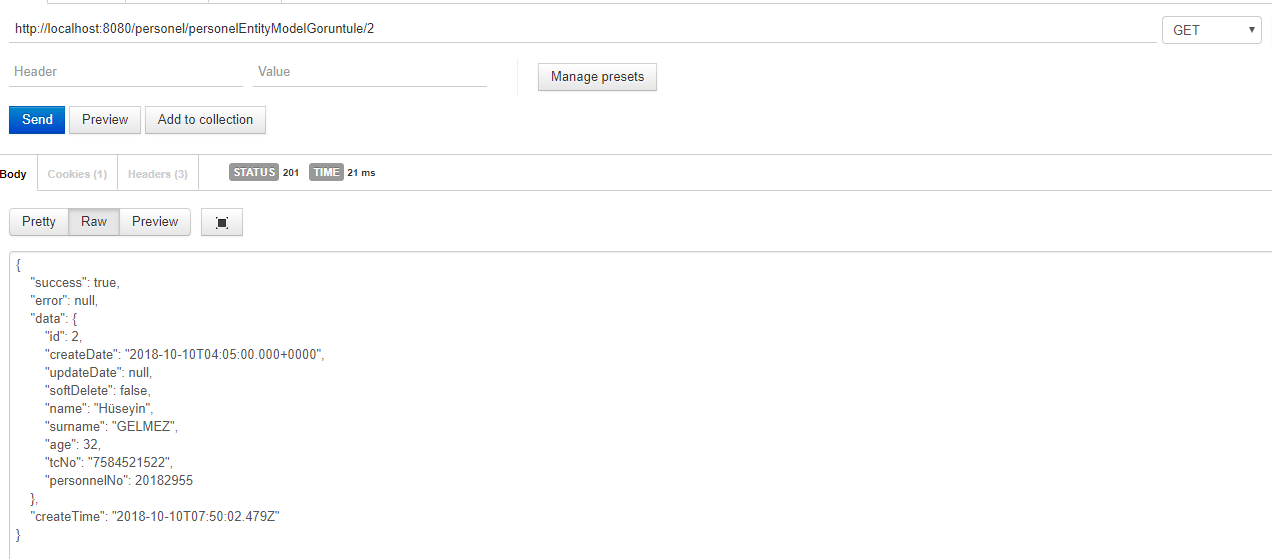
@GetMapping(value = "/personelGoruntule/{id}")

@GetMapping(value = "/personelEntityModelGoruntule/{id}")



**3-Personel Entity Model Görüntüleme**

/personelEntityModelGoruntule/{id}



**4-Personel Ekleme**

@PostMapping(value = "/personelEkle")

**Örnek Json:**

{

id: null,

createDate: "2018-10-09T00:40:57.000+0000",

updateDate: "2018-10-09T00:41:14.000+0000",

softDelete: false,

name: "Onur",

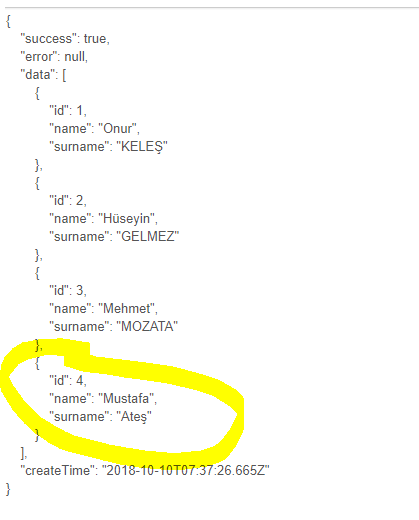
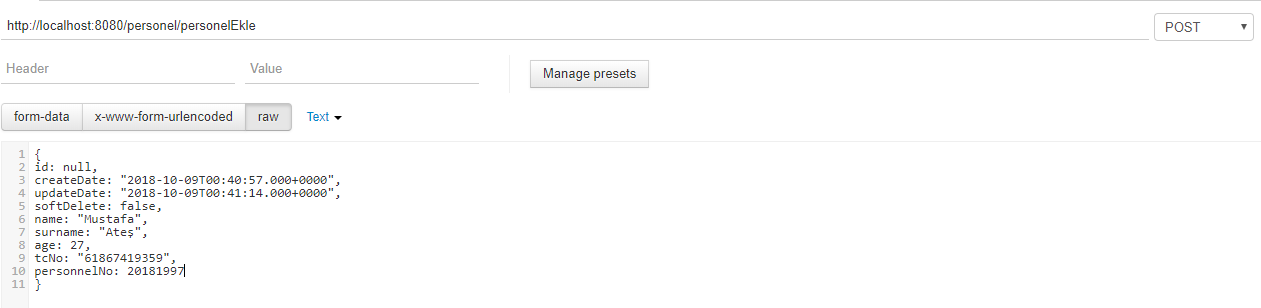
surname: "Keleþ",

age: 27,

tcNo: "61867419350",

personnelNo: 20181991

}



**5-Personel Güncelleme**

/personelGuncelle@PutMapping(value = "/personelGuncelle")

{

id: 1,

createDate: "2018-10-09T00:40:57.000+0000",

updateDate: "2018-10-09T00:41:14.000+0000",

softDelete: false,

name: "Onur",

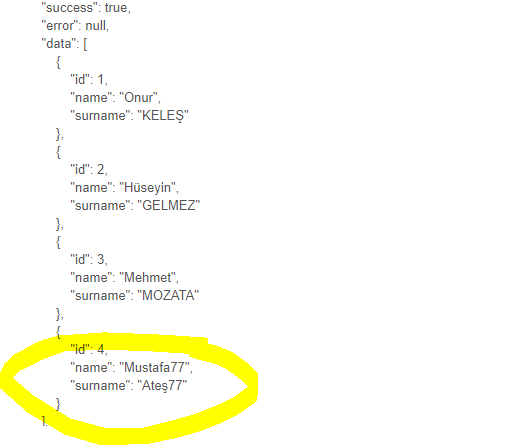
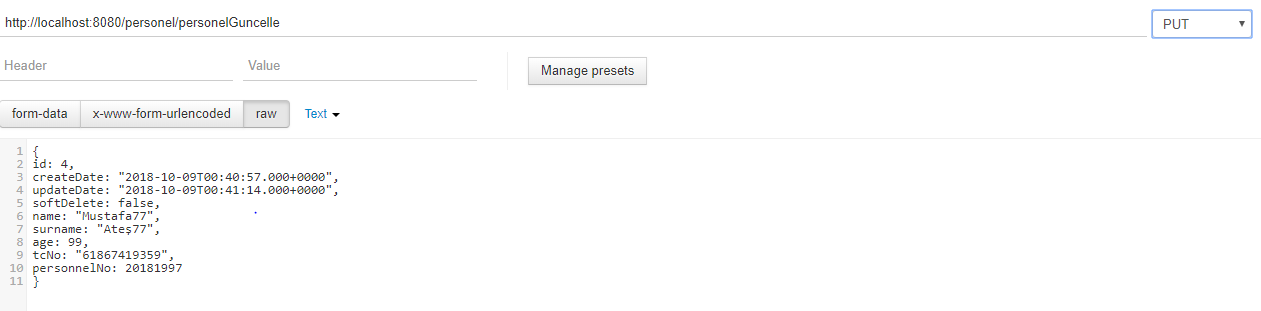
surname: "Keleþ",

age: 27,

tcNo: "61867419350",

personnelNo: 20181991

}



**6-Personel Silme**

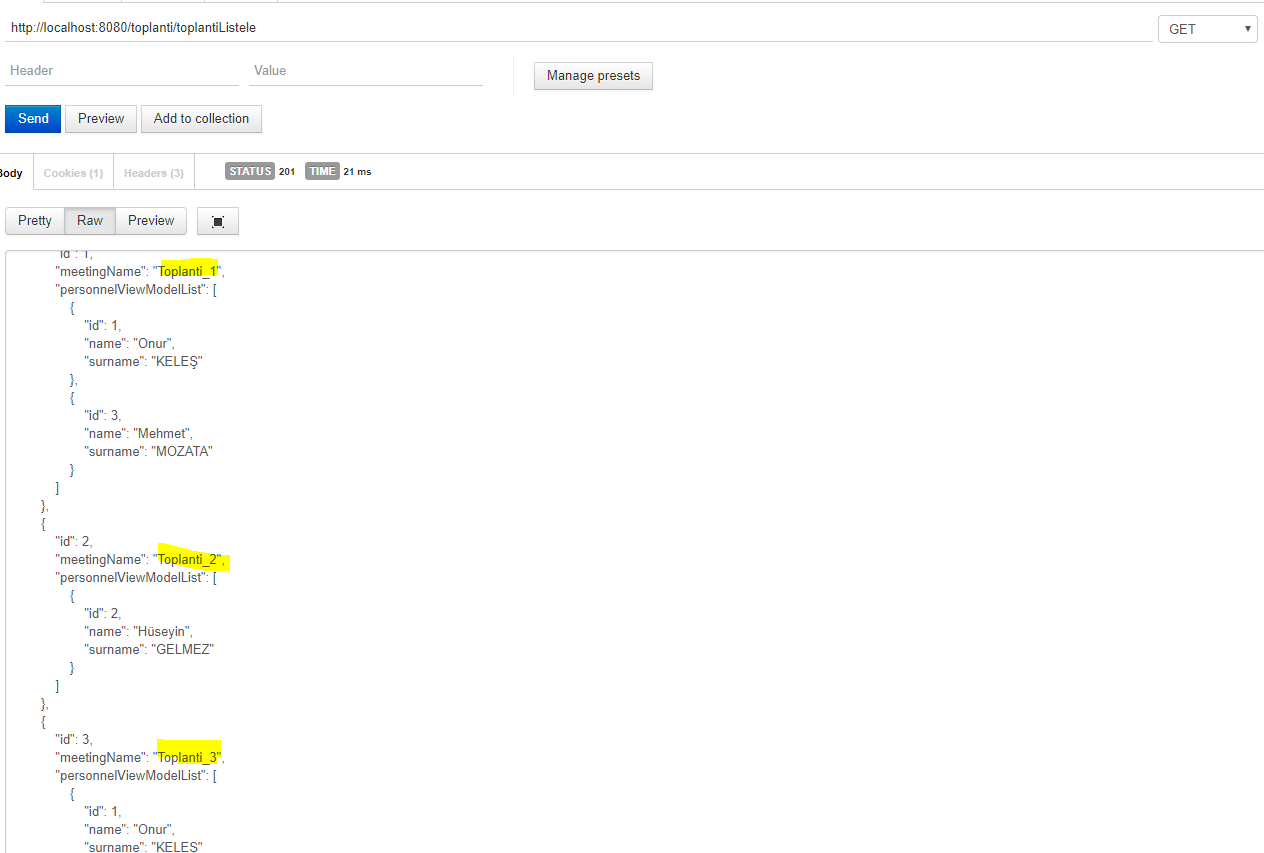
@DeleteMapping(value = "/personelSil/{id}")



**Toplantı İşlemleri**

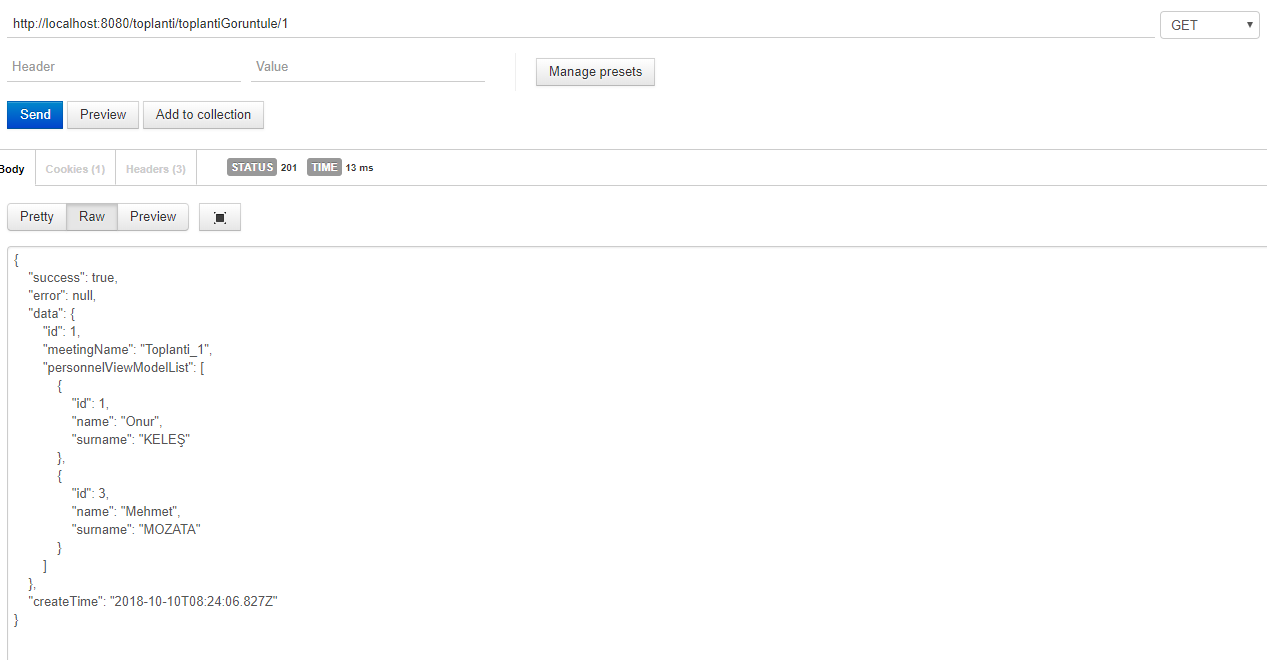
**1-Topalntı Listeleme**

@GetMapping(value = "/toplantiListele")



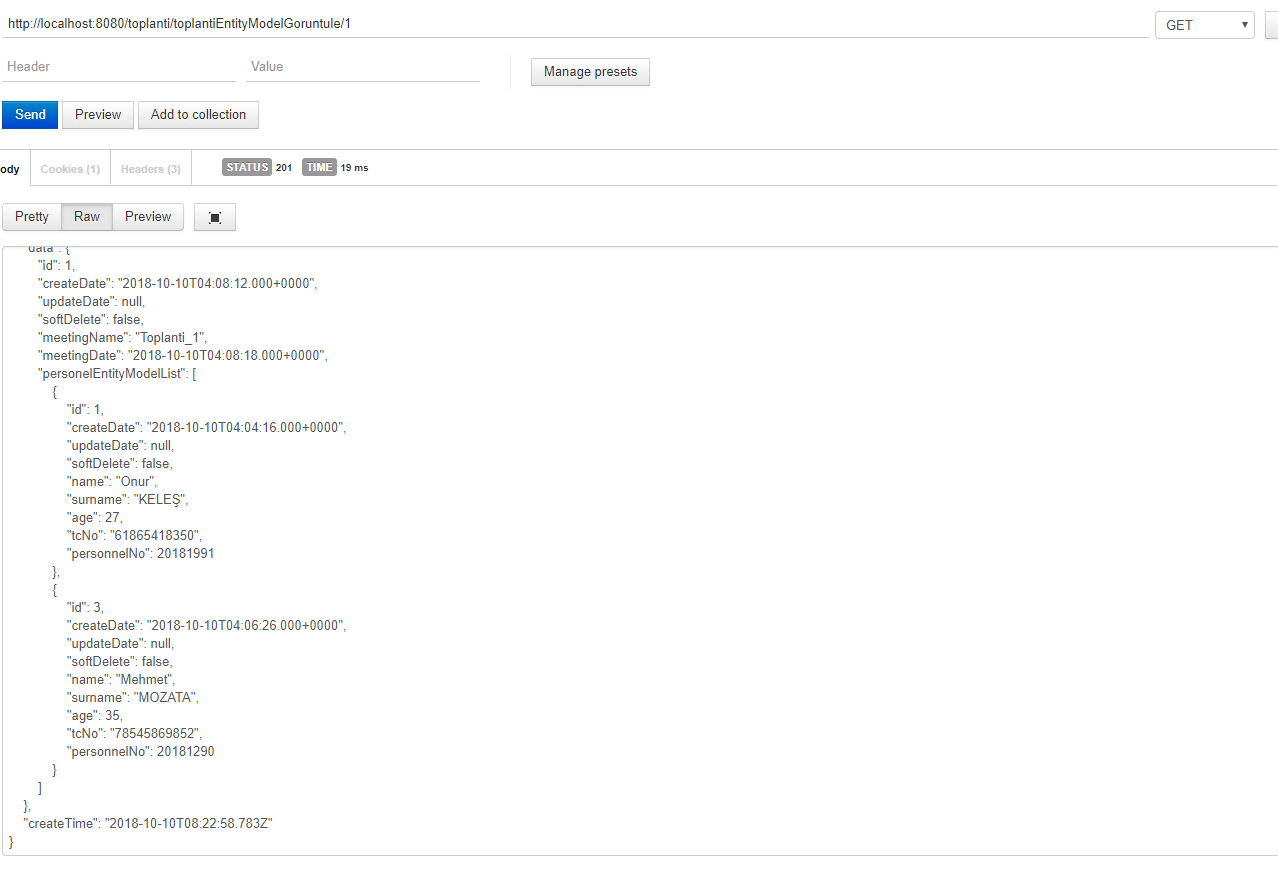
**2-Toplantı Görüntüleme**

@GetMapping (value = "/toplantiGoruntule/{id}")



**3-Toplantı Entity Model Görüntüleme**

@GetMapping (value = "/toplantiEntityModelGoruntule/{id}")



**4-Toplantı Bilgileri Ekleme**

@PostMapping(value = "/toplantiEkle")

Örnek Json:

{

"id": null,

"createDate": "2018-10-10T01:13:54.000+0000",

"updateDate": "2018-10-10T01:14:02.000+0000",

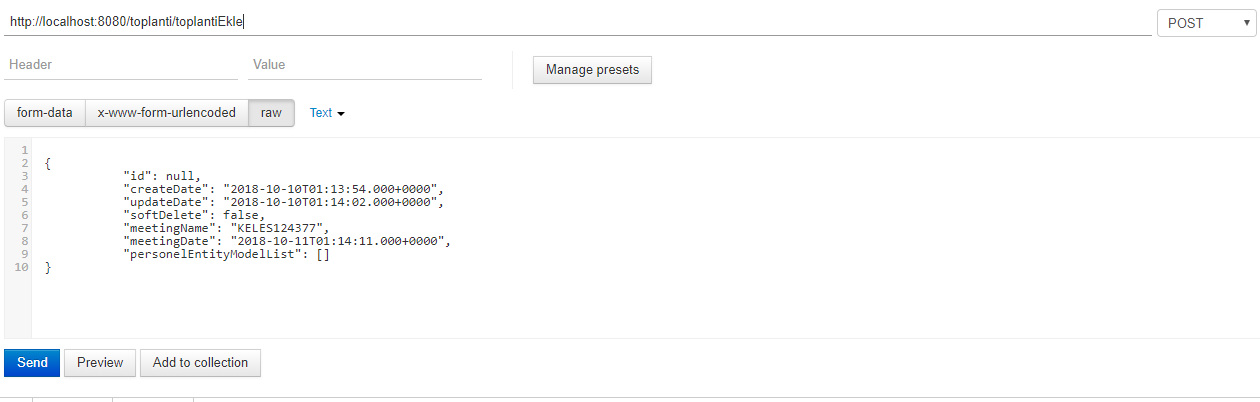
"softDelete": false,

"meetingName": "KELES1111111",

"meetingDate": "2018-10-11T01:14:11.000+0000",

"personelEntityModelList": []

}



**5-Toplantı Bilgileri Güncelleme**

@PutMapping(value = "/toplantiBilgileriniGuncelle")

ÖrnekJson

{

"id": 4,

"createDate": "2018-10-10T01:13:54.000+0000",

"updateDate": "2018-10-10T01:14:02.000+0000",

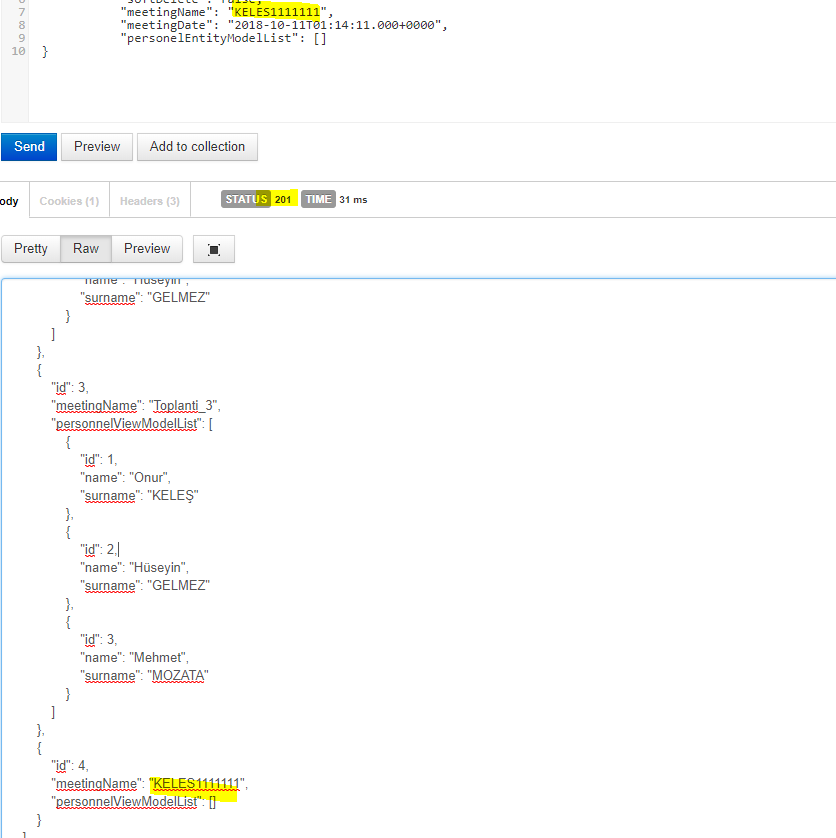
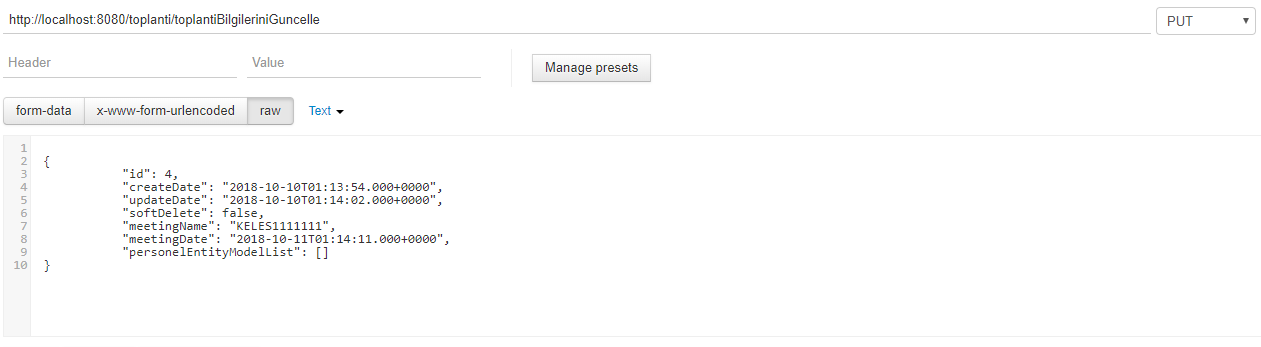
"softDelete": false,

"meetingName": "KELES1111111",

"meetingDate": "2018-10-11T01:14:11.000+0000",

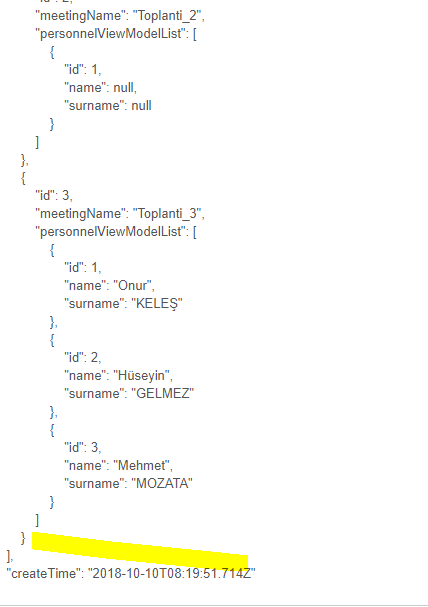
"personelEntityModelList": []

}



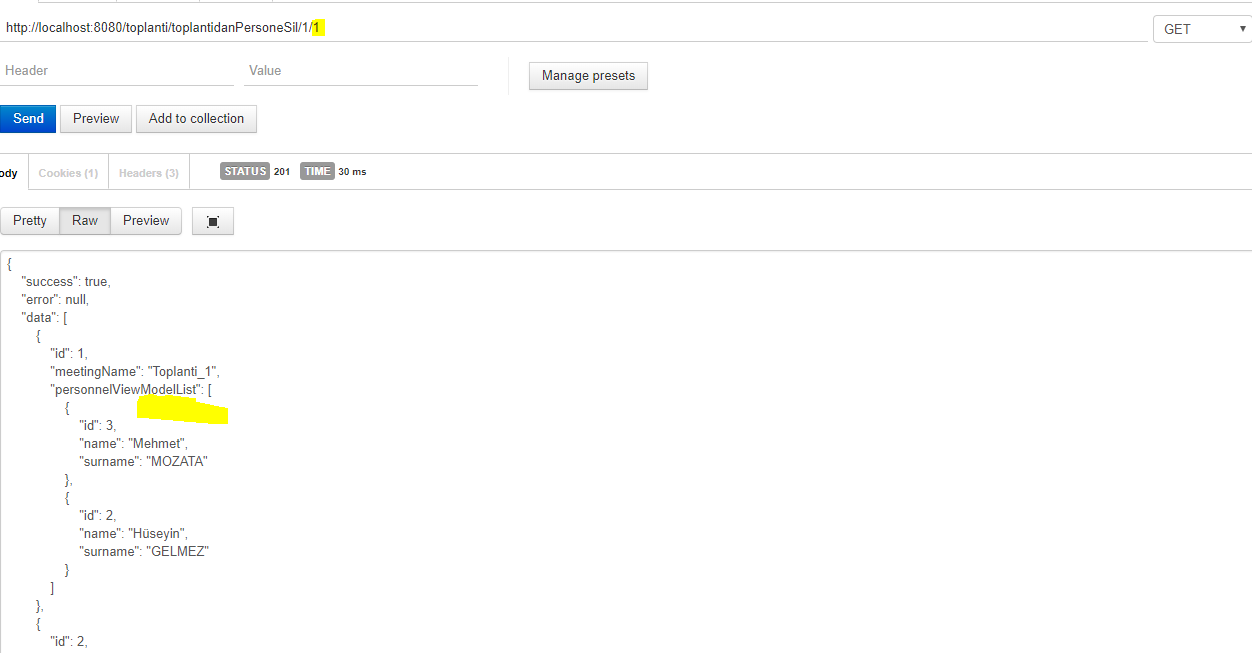
**6-Toplantı Silme**

@DeleteMapping (value = "/toplantiSil/{id}")



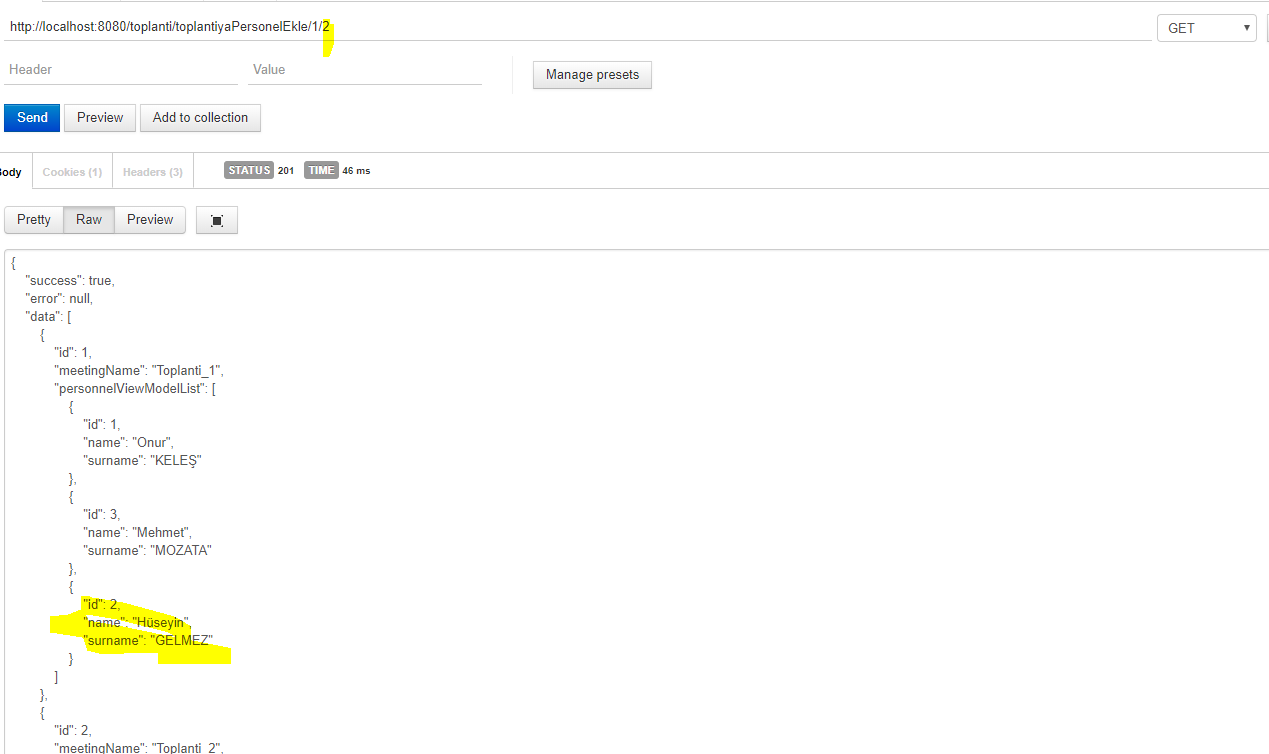
**7-Toplantıya Personel Ekleme**

/toplantiyaPersonelEkle/{toplantiId}/{personelId}"



**8-Toplantıdan Personel Silme**

/toplantidanPersoneSil/{toplantiId}/{personelId}

****