Soru: 1-MVC kavramını açıklar mısınız? Neden ihtiyaç duyuluyor. Java'da nasıl kurgulanıyor. Object Oriented katmanları nelerdir?

Soru: MVC kavramını açıklar mısınız?

Cevap: MVC, Model-View-Controller kelimelerinin baş harflerinden oluşan bir yazılım mimari desenidir. MVC, uygulamanın farklı bileşenlerini mantıklı bir şekilde parçalara ayırmayı ve bu bileşenler arasındaki ilişkilerin tanımlanmasını amaçlar yani Kullanıcı etkileşimlerini, veri erişimini ve iş mantığını birbirinden ayırmak için kullanılır. Bu sayede uygulamalar daha anlaşılır, test edilebilir ve yönetilebilir hale gelmektedir.

Soru: Neden İhtiyaç Duyuluyor?

Cevap: MVC tasarım deseni büyük, karmaşık ve sürdürülemeyen uygulamalar oluşmasını önlemek için kullanılıyor. MVC modelinin bahsetmiş olduğumuz problemlere getirdiği çözümleri madde madde söyleyecek olursak;

1. Uygulamaların daha modüler ve esnek hale getirmesi (Bileşenlerin birbirinden bağımsız hale getirilmesi sayesinde bakım maliyetini düşürür. Bir bileşenin değiştirilmesi veya güncellenmesi durumunda diğer bileşenlere küçük bir etki etmesi.)
2. Geliştirme ve bakım maliyetlerini azaltması (Geliştirme işleminin parçalara bölünmesi sayesinde dallanma olmadan asıl göreve odağın rahatlıkla sağlanabilmesi ve bir problem olduğu zaman projenin modüler olması sayesinde problemin kısa sürede bulunup çözüm sağlanabilmesi sayesinde bakım maliyetlerinin azaltılması.)
3. Test edilebilirliği arttırması (Her bir katmanın ayrı ayrı test edilebilmesi.)
4. Paralel geliştirme (Farklı ekip üyelerinin uygulamanın farklı katmanlarını aynı anda geliştirebilir olması.)
5. Yeniden kullanılabilirlik (Proje içerisinde oluşturulmuş yapılardan bazıları diğer projelerde de kullanılabilir bir yapıya sahip olacak şekilde oluşturulabilir böylece yeniden kullanılabilirlik sağlanmış olur.)

Soru: Java'da nasıl kurgulanıyor?

Cevap: MVC, Java'nın nesne yönelimli bir programlama dili olması sebebiyle Java'da sıkça kullanılmaktadır. Java'da MVC mimarisini uygulamak için aşağıdaki adımları uygulamak gerekmektedir:

1. Model katmanı oluşturulmalıdır
   Model katmanı, uygulamanın veri modelini içermektedir. Bu katman verilerin alınması, işlenmesi ve depolanması ile sorumludur.

2. View katmanı oluşturulmalıdır
   View katmanı, kullanıcı arayüzünü içerir. Bu katmanla birlikte kullanıcı uygulama ile etkileşime girdiği zaman bu etkileşimleri yönetir. View katmanında web teknolojileri olan HTML, CSS ile JavaScript'te kullanılmaktadır.

3. Controller katmanı oluşturun:
   Controller katmanı, kullanıcının isteklerini işler ve model ile view katmanları arasındaki köprü işlevini görür.

Soru: OOP katmanları nelerdir?

Cevap: Cümleden anladığım kadarıyla burada kastedilen OOP katmanları, nesneye yönelik programlamadaki temel özelliklerin temsil edildiği katmanlardır. Bu katmanlar;

- Soyutlama (Abstraction): Bir nesnenin/veri yapısının karmaşıklığını gizlemek ve sadece gerekli(önemli) olan kısımlarını göstermek için kullanılır. Karmaşık sistemleri daha basit ve anlaşılır hale getirir ve gereksiz ayrıntıları gizler.
- Kapsülleme (Encapsulation): Bir nesnenin verilerini(alanlarını) ve bu verilere olan erişim yöntemlerini bütün olarak ele almayı ifade eder yani verilere doğrudan erişim kısıtlanır ve veriye metotlar aracılığıyla erişilir böylece nesnenin iç yapısı gizlenmiş olur.
- Miras Alma (Inheritance): Bir sınıfın başka bir sınıfın özelliklerini ve metotlarını devralmasını ifade eder. Kodun yeniden kullanılması ve sınıflar arasındaki hiyerarşik ilişkiler kurulmasına yardımcı olur.
- Çok Biçimlilik (Polymorphism): Aynı isme sahip olmasına rağmen farklı işlemlerim yapılabilmesini ifade eder. Bir nesnenin farklı sınıflara ait metotlarda farklı şekillerde davranabilmesine olanak tanımasına denmektedir. Böylece kodun esnekliğini arttırır ve genel kullanımına imkân tanınır.

Soru: 2-Birbirinden bağımsız iki platformun birbiriyle haberleşmesi nasıl sağlanabilir. Örneğin, X platformu Java ile yazılmış olsun, Y platform u C# ile. Bu iki platformun birbiri ile iletişim halinde request-response ilişkisi kurması gerekiyor. Bu yapıyı nasıl sağlarız?

Cevap: Bu problemi çözmek için RESTful servisler oluşturabiliriz. RESTful servisler ile platformlar arası iletişimde hafif ve esnek bir yaklaşım sunmuş oluruz.

Her iki platformda da RESTful servisler oluşturmalıyız. Bunun için Java'da Spring Boot kullanabiliriz ve C#'ta da ASP.NET Core kullanabiliriz. Java ve C# platformları için aşağıdaki adımlar uygulanmalıdır:

1. Kurulum:
    a. Java: Spring Boot kurulması
    b. C#: ASP.NET Core kurulması
2. Yeni Proje Oluşturulması
3. RESTful Servisi Tanımlama
    a. Java: '@RestController' veya '@Controller' sınıflarının oluşturulması ve RESTful servisin temelinin oluşturulması.
    b. C#: Gerekli controller sınıflarının oluşturulması ve bu sınıfların 'ControllerBase' sınıfından türetilmesi.
4. Son Noktaların (Endpoint'lerin) Tanımlanması
    a. Java: RESTful servislerinde yönlendirme yapmak için '@GetMapping', '@PostMapping' gibi Spring anotasyonlarının kullanımı.
    b. C#: HTTP isteklerini yönlendirmek için 'Route' ve 'ApiController' özellikleri kullanarak yönlendirmeler yapılır.
5. Veri Döndürme ve Alma:
   RESTful servislerimiz JSON veri biçimini kullanarak alışveriş yapabilir. Bunun için kütüphaneler kullanılabilir.
6. Gerekli adımların uygulanması ve uygulamaların çalıştırılması ile platformlar arasındaki iletişimin kurulması gerçekleşir.

Soru: 3-Bir web sayfasında ekran sürekli Backend' den veya bir başka yapı tarafından güncelleniyor. Siz, web sayfasını refresh etmeden bu güncel bilgiyi anlık ekrana nasıl yansıtırsınız?

Cevap: WebSocket veya Server-Sent Events (SSE) kullanılabilir benim tercihim WebSocket tarafından olacaktır.

WebSocket'i birçok modern web tarayıcısı ve sunucu tarafından desteklenen bir iletişim protokülü olması, sürekli ve gerçek zamanlı iletişim sağlaması ile tercih edilebilir olmasını sağlaması haricinde SSE sadece sunucu tarafından istemciye veri gönderme yeteneğine sahip olması sebebiyle WebSocket'i bana göre daha tercih edilesi yapıyor.

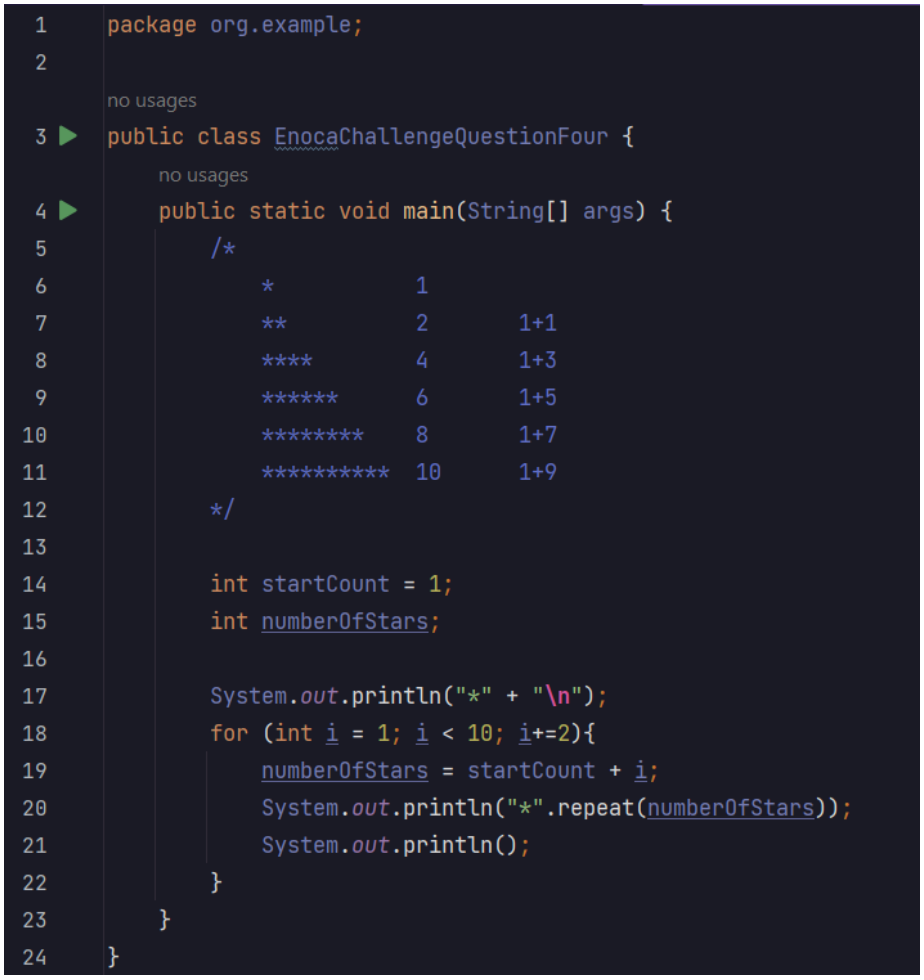
WebSocket için; Çift yönlü gerçek zamanlı iletişim kurması gereken senaryolarda kullanmak daha uygunken SSE, sunucudan istemciye tek yönlü gerçek zamanlı güncellemeler gerektiren senaryolar için daha uygundur.

Java özelinde WebSocket'i kullanabilmemiz için JavaEE WebSocket API ve Spring WebSocket bulunması ekstra bir artı olmaktadır.

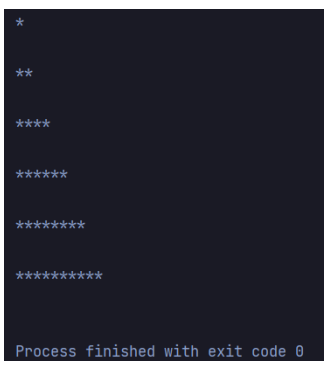
Soru: 4-Bir for döngüsü ile aşağıdaki çıktıyı yazar mısınız?

```
*

**

****

******

********

**********
```

Cevap: Kod ve kod çıktısının görselleri eklenmiştir.



```
package org.example;

no usages
public class EnocaChallengeQuestionFour {
    no usages
    public static void main(String[] args) {
        /*
            *          1
            **         2       1+1
            ****       4       1+3
            ******     6       1+5
            ********   8       1+7
            **********  10      1+9
        */

        int startCount = 1;
        int numberOfStars;

        System.out.println("*" + "\n");
        for (int i = 1; i < 10; i+=2){
            numberOfStars = startCount + i;
            System.out.println("*".repeat(numberOfStars));
            System.out.println();
        }
    }
}
```



```
*

**

****

******

********

**********

Process finished with exit code 0
```

Soru: 5- Firmada çalışman için sana remote bir linux server verildi. Elinde ip adresi port bilgisi kullanıcı adi ve şifren var. Server a erişimi nasıl test edersin, Server a nasıl erişirsin, Server a nasıl dosya atarsın, Server'dan nasıl dosya çekersin?

Soru: Server'a erişimi nasıl test edersin?

Cevap: Server'a ping atabiliriz:

Terminali açtıktan sonra local makinemizden server'a erişim sağlanabiliyor mu diye ping atabiliriz. Örnek kod:

```
ping <server_ip>
```

Server'a ulaşım sağlanılıp sağlanamadığını gelen çıktıdan yorumlanabilir. Server cevap vermiyorsa server'a erişim sağlanamıyor veya ICMP'den istekler alınamıyor olabilir.

Soru: Server'a nasıl erişirsin?

Cevap: SSH Bağlantısı Yapılabilir

SSH bağlantısını yaparken port numarasını da bildiğimi için port numarası bilgisini de girebiliriz.

```
ssh <user_name>@<server_ip> -p <port_number>
```

Sunucuya bağlanma girişiminde güvenli bağlantı sorusunu geçtikten sonra sahip olduğumuz kullanıcı adı ve şifreyi girerek giriş yapılabilir.

Soru: Server'a nasıl dosya atarsın?

Cevap: 'scp' komutu ile dosya transferi yaparım.

'scp' (Secure Copy Protocol), dosyaları güvenli bir şekilde kopyalamak için kullanabileceğimiz bir komuttur ve SSH üzerinden çalışır.

```
scp <file_path> <user_name>@<server_ip>:<remote_path>
```

Komutunu kullanarak yerel makinemizdeki dosyayı sunucuda istediğimiz dosya yoluna kopyalayabiliriz. Bu yöntem dışında SFTP de kullanılabilir.

Soru: Server’dan nasıl dosya çekersin?

Cevap: ‘scp’ komutu ile dosya çekerim.

‘scp’, komutunu bu sefer de dosya çekmek için kullanabiliriz. (Sunucudan yerel makineye)

```
scp <user_name>@<server_ip>:<remote_path/file> <local_path>
```

Komutunu kullanarak sunucuda bulunan ve çekmek istediğimiz dosyayı yerel makinemize çekebiliriz. Aynı şekilde bu yöntem dışında SFTP de kullanılabilir.

Soru: **6-** Local database kurulumu (mysql, postgresql veya herhangi bir database)

- Java spring uygulaması ayağa kaldırılması,
- İki adet tablo yer almalı ve bu tabloların birbirleriyle bağı olmalıdır. (Ör: şirket ve çalışan gibi),
- Java spring uygulamasında ekleme, silme, güncelleme, listeleme gibi servisler yer almalıdır ve
responseda yapılan işlem detayı return edilmelidir.
- Ekleme, silme, güncelleme, listeleme işlemlerini postman vb ile işlem yapılabilmelidir.

Bu adımlar sırasıyla izlenip java uygulaması üzerinden database' e kayıt atılmalı (Herhangi bir kayıt olabilir fark etmez. Database'de bir tablo açılıp o tabloya değer girilmesi java isteği üzerinden). Daha sonra aynı istek atılan uygulama ile (örnek postman ...) get ve list java spring endpointleri çağırılarak, database e atılan kayıt response olarak dönülmeli.

MVC deki model ve kontroller akışının güzel kurgulanması ve uygulama ayağı nasıl kaldırılıp servislerin nasıl kullanıldığına dair bir döküman hazırlanmalıdır. Bu proje için kaynak kodu ile iletilmesi gerekmektedir.

Cevap:

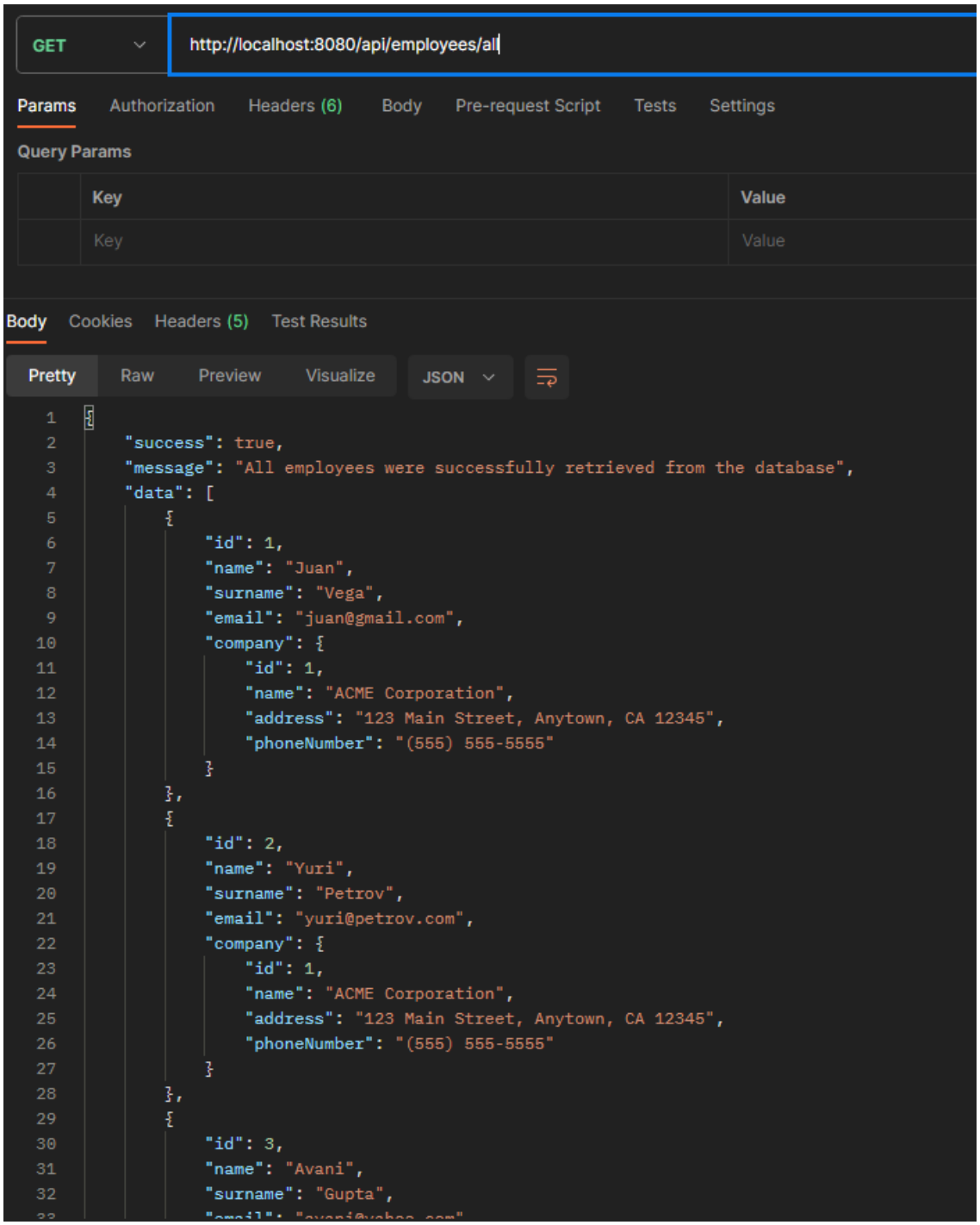
Postman Çıktıları:

Employee için:

Get Methodu:

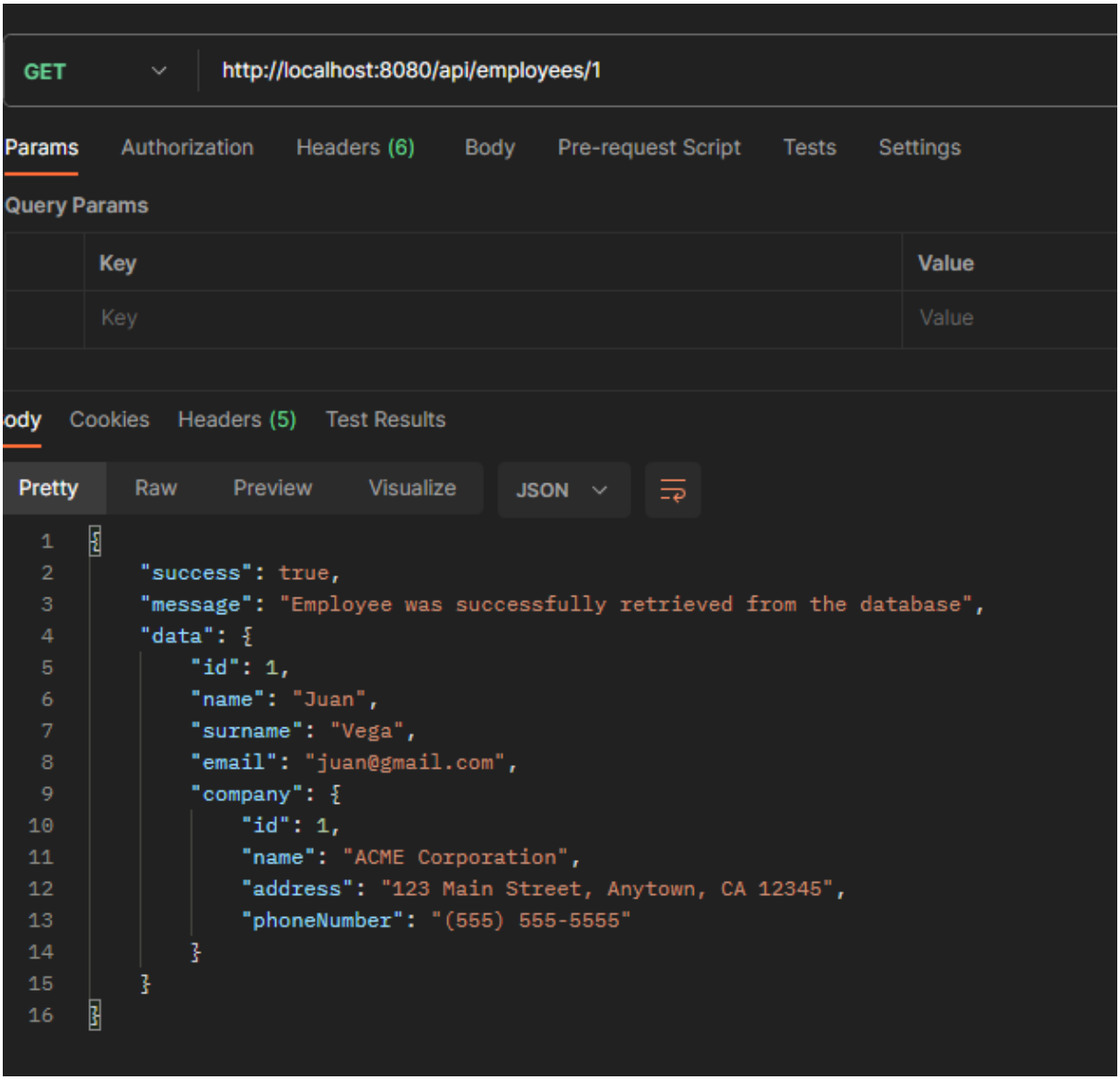
Get all employees:

Request-response sonucunda “message” bilgisi response’ta görünmektedir.

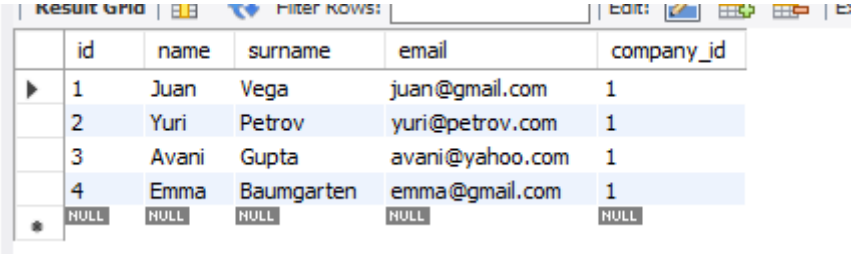


Get 1 employee:

Request-response sonucunda “message” bilgisi response’ta görünmektedir.



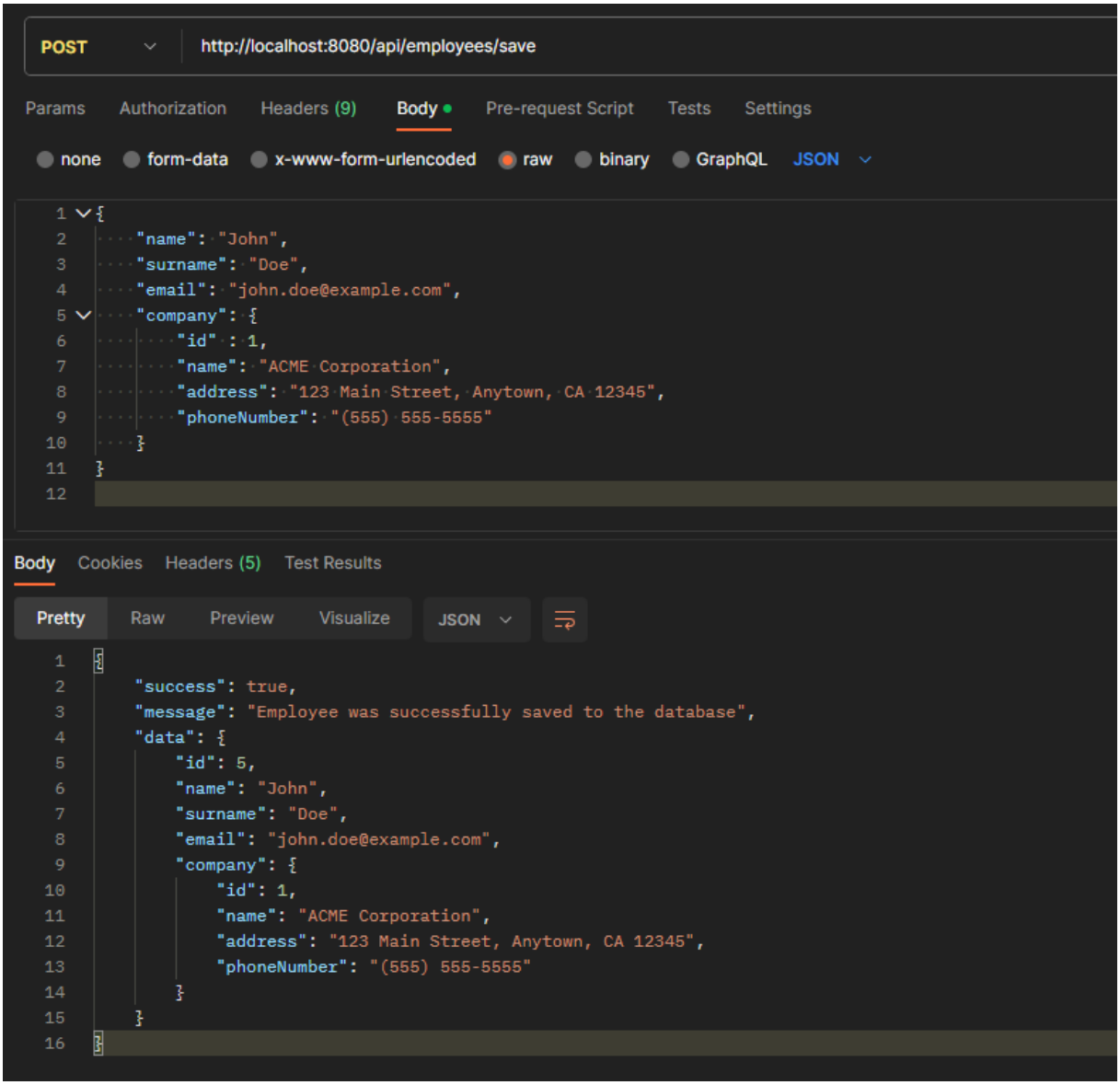
Post metodu öncesi mysql verileri:



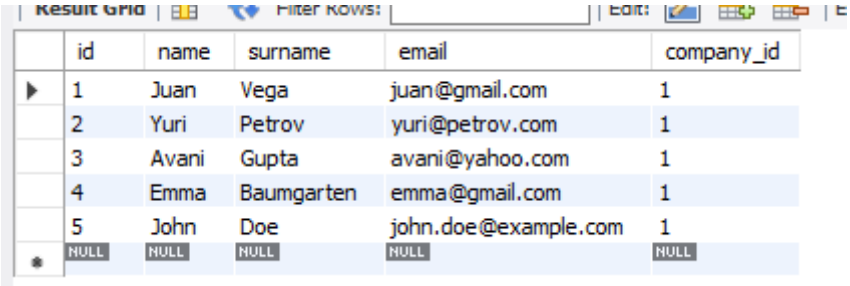
id	name	surname	email	company_id
1	Juan	Vega	juan@gmail.com	1
2	Yuri	Petrov	yuri@petrov.com	1
3	Avani	Gupta	avani@yahoo.com	1
4	Emma	Baumgarten	emma@gmail.com	1
NULL	NULL	NULL	NULL	NULL

Post metodu:

Request sonucunda response’ta “message” bilgisi altında çalışanın veri tabanına eklendiği bilgisi gösterilmektedir.



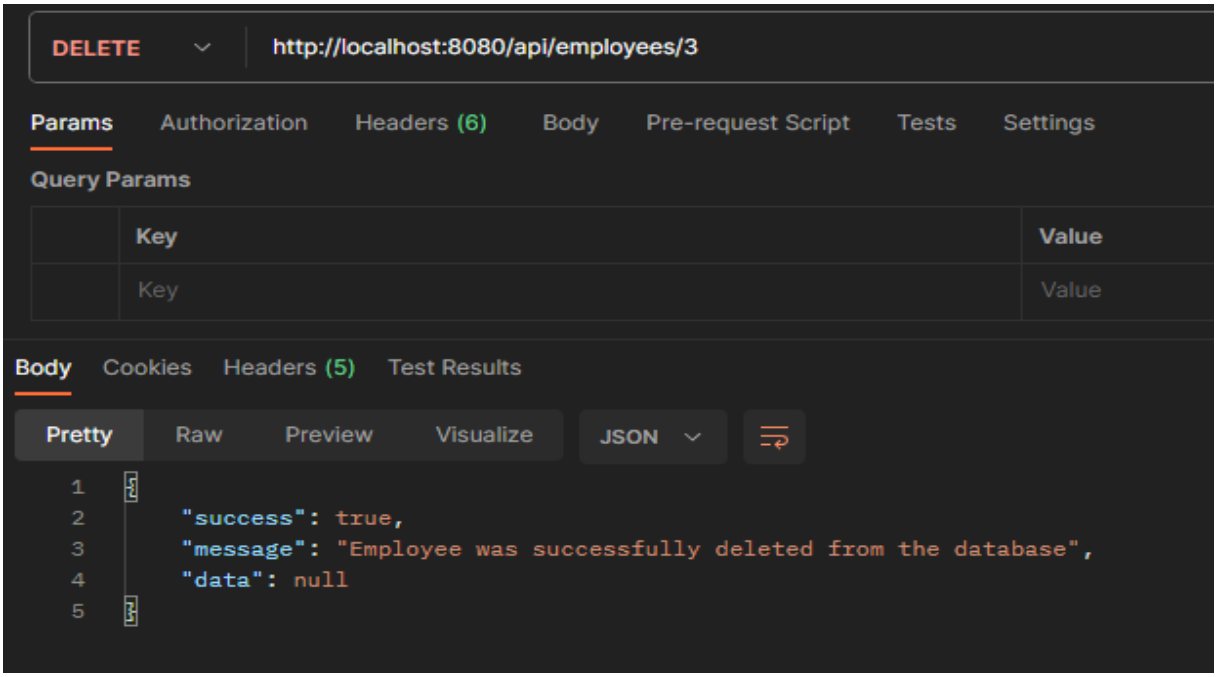
İşlem sonucu veri tabanındaki employee tablosu:



id	name	surname	email	company_id
1	Juan	Vega	juan@gmail.com	1
2	Yuri	Petrov	yuri@petrov.com	1
3	Avani	Gupta	avani@yahoo.com	1
4	Emma	Baumgarten	emma@gmail.com	1
5	John	Doe	john.doe@example.com	1
NULL	NULL	NULL	NULL	NULL

Delete metodu:

Request sonucunda response'ta "message" bilgisi altında çalışanın veri tabanından silindiği bilgisi gösterilmektedir.



İşlem sonucu veri tabanındaki employee tablosu:

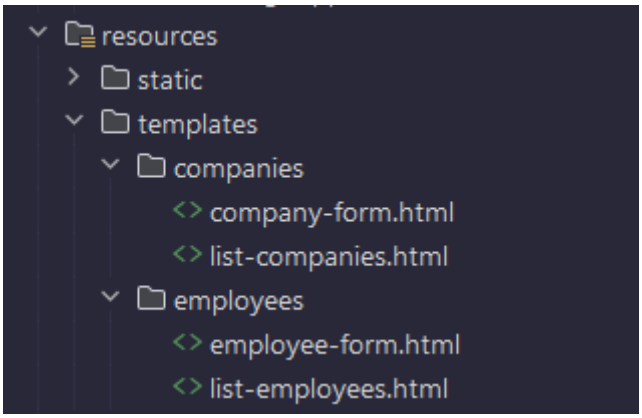
id	name	surname	email	company_id
1	Juan	Vega	juan@gmail.com	1
2	Yuri	Petrov	yuri@petrov.com	1
4	Emma	Baumgarten	emma@gmail.com	1
5	John	Doe	john.doe@example.com	1
NULL	NULL	NULL	NULL	NULL

Son maddede istenen MVC'ki akışın nasıl kurgulandığı ve uygulandığını anlatacak olursak:

1. Adım:
   İlk olarak gerekli modellerimiz oluşturuldu (MySql de bulunan tablo içeriklerimize uygun olacak şekilde).
2. Adım:
   Core.utilities.result şeklinde bir core katmanı hazırlandı bunun sebebi api controller'ımızdan döndürülecek sonuçlarda işlem durumunun başarılı olup olmadığına dair bilgilendirme mesajlarının gösterilmesi.
3. Adım:
   Gerekli olan DAO sınıflarımız oluşturuldu böylece MVC'deki veri erişim objemiz tanımlandı.
4. Adım:

Service sınıflarımız oluşturuldu örneğin Interface olarak EmployeeService, Class olarak EmployeeServiceImpl. Impl sınıflarımız içerisinde veri erişim objelerimiz olan DAO'lar kullanılarak veri tabanına erişim sağlanarak istenilen işlemi gerçekleştirecek olan metotlar yazılmıştır.

5. Adım:
   Controller katmanını ise iki parçaya ayırdım. Controller katmanında her bir modelin kendisine ait api ve template sınıfları oluşturuldu. Böylece Postman işlemlerimi yapmak istediğimiz zaman api sınıflarının controller'ları çalışırken (örnek url: http://localhost:8080/api/employees/all) MVC'nin View kısmı için hazırlanmış olan template yapılarımız olan template controller'larımız çalışacak. Template'leri main/resources/templates altında model türüne göre klasörlere ayırdım.



Sonuç olarak tüm proje MVC mimarisine uygun şekilde Model katmanına, View katmanına, Controller katmanına sahip ve kendilerine uygun klasörleme yapıları altında gerçekleştirilmiştir. İsterde belirtilen Postman işlemleri için Api'lar oluşturulmuş ve Postman işlemleri yerel makinede gerçekleştirilebilir halde test edilmiştir.

Kaynak Kod Linki: https://github.com/RafetJinx/challenge

Soru: 7- Apache Solr servisine yazılacak bir query örneği Apache Solr kullanılan sql programlarından daha farklı runtime bir database. Solr a hali hazırda kayıtlı bir alan olduğunu düşünelim. Alanın ismi "updatedAt" long tipinde tutulan bir alan. Ben 2020 Ocak ayından sonraki verileri getir dediğimde solr a nasıl bir query yazılmalı. http://example?query= kısmını nasıl doldurmalıyım?

Cevap: Apache Solr üzerinde yazılması gereken query:

```
http://example?query=updatedAt:[2020-01-01T00:00:00Z TO *]
```

Bu query ile 2020 Ocak ayından sonraki veriler getirilecektir.

```
http://example?query=updatedAt:[2020-01-01T00:00:00Z TO *] AND
type:product
```

Query'si ise 2020 Ocak ayından sonra güncellenmiş olan product verilerini getirecektir.