**27.03.2023**

İlgili günde HTTP protokolüne ait GET,POST,PUT,DELETE metotlarının nasıl çalıştığı ve pratikte nasıl kullanılması gerektiği hakkında araştırma yapılmıştır. Araştırmalar sonucunda H2 Database ve bir Rest API servisi üzerinde HTTP metotlarının işlevsel olarak nasıl kullanıldığının anlaşılması için bir vatandaşlık sistemi örneğinin geliştirilmesine başlanmıştır.

Rest (Representational State Transfer); Temsili Durum Transferi anlamına gelmektedir. Rest, bir Web uygulamasının dış dünyaya açıldığı kapıdır. Kullanıcılar bu kapılardan veri talep edebilir veya veri gönderebilir. Rest’ten hariç olarak SOAP, WSDL, GraphQL, gRPC gibi API mimarileri de vardır. Tüm bu API mimarileri Client-Server iletişimini HTTP protokolü üzerinden yapabilmektedir. Bazı mimariler TCP/IP, JMS gibi protokolleri de kullanabilmektedir.

HTTP protokolünde farklı işlem metotları bulunmaktadır. Bunlardan başlıcaları GET, POST, PUT, DELETE fonksiyonlarıdır. Bu metotları ayrı ayrı açıklayacak olursak;

**GET**: Bu metot sunucudan bir veri talep etmek için kullanılır. URL üzerinden gönderilen parametreler gözükür durumdadır.

**POST**: Bu metot client tarafından sunucu tarafına bir veriyi göndermek için kullanılır. URL üzerinde gözükmesi sakıncalı olan bilgiler bu yöntemle gönderilmelidir. Örneğin kullanıcı adı ve parola bilgisi.

**PUT**: Sunucu tarafındaki bir verinin güncellenmesi gerekiyorsa bu metot üzerinden ilgili veriye ait ID bilgisi gönderilmelidir.

**DELETE**: Sunucu tarafındaki bir verinin silinmesi gerekiyorsa bu metot kullanılmalıdır. Spesifik bir verinin silinmesi gerekiyorsa ID bilgisi gönderilmelidir.

Rest API ve HTTP protokolüne ait metotlarının H2 Database ile kullanımının kavranması için geliştirilmesi hedeflenen vatandaşlık sistemi örneği için Spring Boot üzerinde Entity katmanı oluşturulmuştur. Proje çalıştırıldığı zaman RAM üzerinde çalışması planlanan veritabanında citizen isminde bir tablo oluşturulacaktır. Bu tabloda bir vatandaşa ait isim, soyisim, kimlik numarası bilgileri saklanacaktır. ORM yapısı için Entity katmanında Citizen isminde bir sınıf oluşturulmuştur. Bu sınıfa Entity katmanında olduğu için @Entity anotasyonu verilip bir tabloyu temsil ettiği için de @Table(name=”citizens”) anotasyonu verilmiştir. Her vatandaşa ait bilgiler field olarak tanımlanıp @Column(name=”field\_ismi”) anotasyonu verilmiştir. Her field için getter-setter metotlar tanımlanmıştır. Bunun yanında önceki yapılan çalışmalardan farklı olarak @Entity anotasyonuna sahip bu sınıfa Constructor(yapıcı metot) fonksiyon tanımlanmıştır. Bu fonksiyon, kullanıcı veritabanına POST yöntemi ile bir veriyi kaydetmeye çalıştığı zaman nesne oluşturup setter metotları çalıştırmak yerine doğrudan yapıcı metotlar tarafından kaydın oluşturulmasını sağlamıştır. Bunun yanında bir kullanıcıya ait id, isim, soyisim, kimlik numarası bilgisinin bir String formatında bütünleşik olarak elde edilmesi için aşağıdaki gibi toString() metodu tanımlanmıştır.

@Override

public String toString() {

return "Citizen [id=" + id + ", name=" + name + ", surname=" + surname + ", citizen\_number=" + citizen\_number + "]";

}

Entity katmanına erişilebilmesi için Repository katmanı oluşturulmuştur. Bu katman içerisinde Citizen nesnelerine erişilebilmesi için CitizenRepository isminde bir interface oluşturulmuştur. Bu interface önceki günlerde yapılan çalışmalardan farklı olarak JpaRepository katmanından kalıtım almaktadır. JpaRepository, CrudRepository’e göre daha geniş işlevlere sahip olan fonksiyonları barındırmaktadır. Örneğin bir tablodaki verilerin tümünü etkileyecek işlemleri yapmak gibi. Bu interface içerisine aşağıdaki gövdesiz fonksiyon tanımlanmıştır. Bu fonksiyon veritabanı üzerinde isim ile arama yapmak için kullanılacaktır.

List<Citizen> findByName(String name);