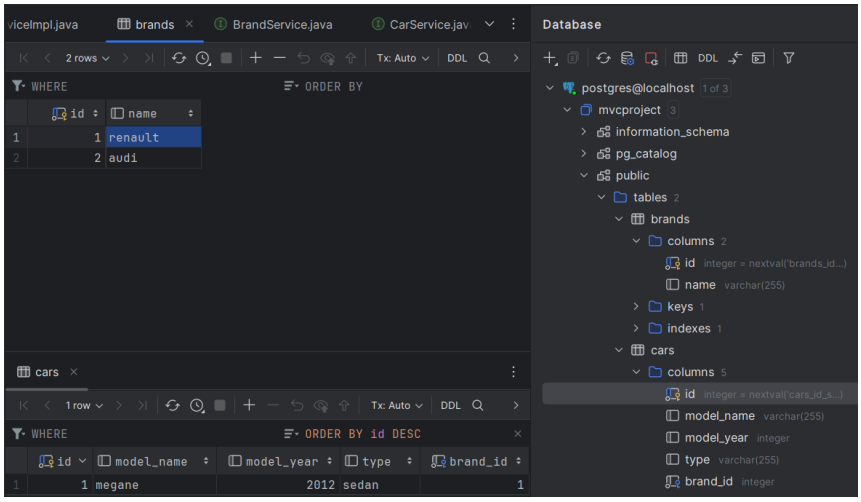
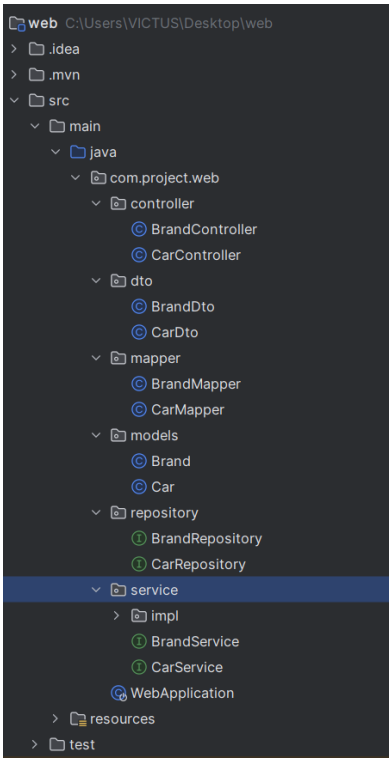
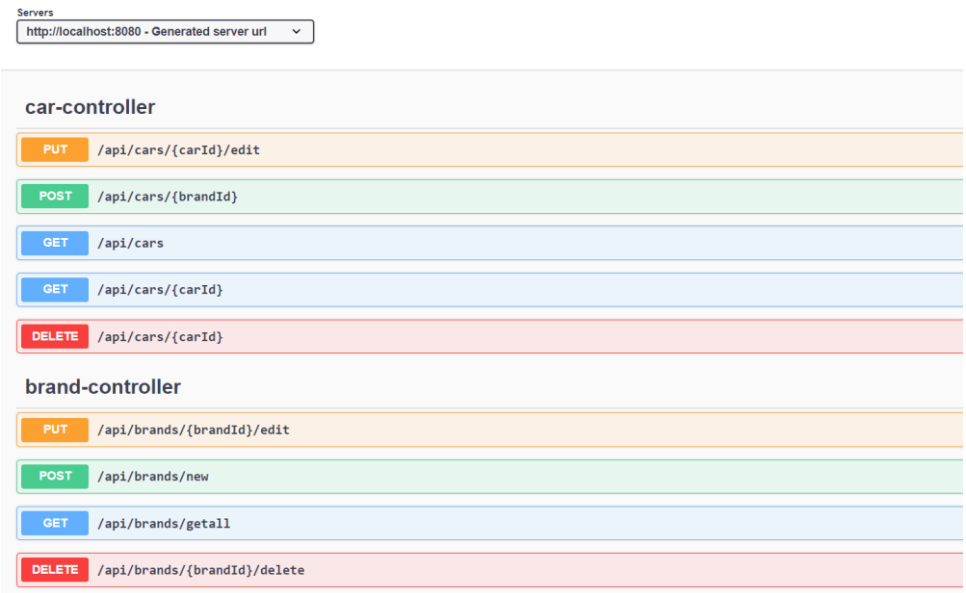
İlk olarak veritabanını idede kullanabildiğim için Intellij Idea idesini tercih ettim. Database olarak PostgreSQL tercih ettim. Gerekli kurulumları yaptıktan sonra https://start.spring.io/ web sitesinde gerekli ayarları ve kullanacağım dependency leri belirledim. Sonrasında projeyi oluşturdum gereken klasör hiyerarşisini yaptım. Controller, servis, dto gibi. Projeyi gereksinimlere uygun bir şekilde yazdım. Projede frontend kısmı olmayacağı için crud işlemlerini swagger ui üzerinden denedim ve oluşturulan istek ve yanıtların database üzerinde davranışlarını kontrol ettim. Klasör hiyerarşisi, database tabloları ve swagger-ui görüntülerini aşağıya ekledim.







KULLANILAN YAPININ AÇIKLAMASI

Öncelikle proje mvc deseni göz önünde bulundurularak sorumluluklarına göre aşağıdaki paketlere ayrıldı.

- **Model**: "Models" paketi, uygulamada veri yapısını ve veritabanı nesnelerini temsil eden Entity sınıflarını içerir. Bu sınıflar, verilerin nasıl saklandığını ve temsil edildiğini tanımlar.
- **DTO** : Bu paket, istemciden sunucuya iletilen ve sunucudan istemciye iletilen verileri temsil eder. Bu sınıflar, veri taşıma amacı güder. Bu sınıflar, projeye veri transferi işlevselliği eklerken, Model sınıflarından gelen verilerin view katmanına uygun bir şekilde taşınmasını sağlar.
- **Mapper**: Bu paket, projede Model sınıfları ile DTO sınıfları arasında dönüşümü yönetir. Model sınıflarından DTO sınıflarına ve tam tersi dönüşümleri gerçekleştirirler. Bu, veri transferi sırasında veri yapısını ve formatını uygun şekilde ayarlar.
- **Repository:** Bu paket, veritabanı işlemlerini gerçekleştirir. Entity (Model) sınıflarını veritabanına kaydeder, okur, günceller ve siler. Repository sınıfları, Model sınıflarının veritabanı işlemleri ile etkileşimini yönetir. Yani Service katmanı ve Model sınıfları arasındaki veritabanı işlemlerini soyutlar.
- **Service**: Bu paket, uygulamanın iş mantığını yönetir. Controller sınıfları tarafından çağrılır ve iş mantığı işlemlerini gerçekleştirir. Bu sınıflar, Controller sınıflarından gelen verileri işler, uygun Repository sınıflarını

kullanarak veritabanı işlemleri gerçekleştirir ve sonuçları Controller sınıflarına döner. Service paketi, iş mantığı ve işlemleri yönetirken, Controller ve Repository sınıfları arasındaki arabirimdir.

- **Controller :** Controller paketi, gelen HTTP isteklerini karşılar ve istemciden gelen veriyi alır. İş mantığı işlemlerini başlatır, uygun Service sınıflarını çağırır ve sonuçları uygun görünümlere (View) yönlendirir. Bu projede frontend tarafı olmadığı için view çalışmıyor fakat yine de addAttribute kullanarak veriyi view tarafından kullanılabilir yaptım. Özetle controller sınıfları, istemci ile etkileşimde bulunur ve sonuçları istemciye gönderir.