
PROJE-1



KIRIKKALE
ÜNİVERSİTESİ

KIRIKKALE ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ BİLGİSAYAR MÜHENDİSLİĞİ
BÖLÜMÜ

PROJE-1 DERSİ

Java ile Veteriner Kliniği Otomasyon Projesi

Dr. Öğretim Üyesi Serkan Savaş
Doğacan Özdemir
190205097

KIRIKKALE-2023

İÇİNDEKİLER

Sayfa Numarası

| | |
|---|-----------|
| TEŞEKKÜR..... | 3 |
| 1.GİRİŞ..... | 4 |
| 1.1. Java Tarihçesi..... | 5 |
| 1.2. Java Kullanım Alanları..... | 6 |
| 1.3. Katmanlı Mimari..... | 7 |
| 1.4. IntelliJ Geliştirme Ortamı..... | 9 |
| 2. KULLANILAN TEKNOLOJİLER..... | 10 |
| 2.1. Spring Framework..... | 10 |
| 2.2.MySQL..... | 16 |
| 2.3. Bootstrap..... | 17 |
| 2.4. Thymeleaf..... | 19 |
| 2.5. Apache Tomcat Server..... | 19 |
| 2.6. Hibernate..... | 20 |
| 2.7.Lombok..... | 21 |
| 2.8 Maven..... | 21 |
| 3.JAVA İLE VETERİNER KLİNİĞİ OTOMASYONU DETAYLARI..... | 22 |
| 3.1.Dosya Yolları..... | 22 |
| 3.2. Proje Mimarisi..... | 23 |
| 3.3. Tomcat Server Üzerinde Çalışır Hali..... | 33 |
| 3.4. MySQL..... | 34 |
| 4. SONUÇLAR VE ÖNERİLER..... | 36 |
| 5.KAYNAKÇA..... | 37 |

TEŞEKKÜR

Öncelikle bitirme projesi konusu seçerken isteklerimi göz önünde bulundurup, projenin yürütülmesi sırasında bana yardımcı olan proje danışmanım Dr.Öğretim Görevlisi Serkan Savaş' teşekkürlerimi iletiyorum.

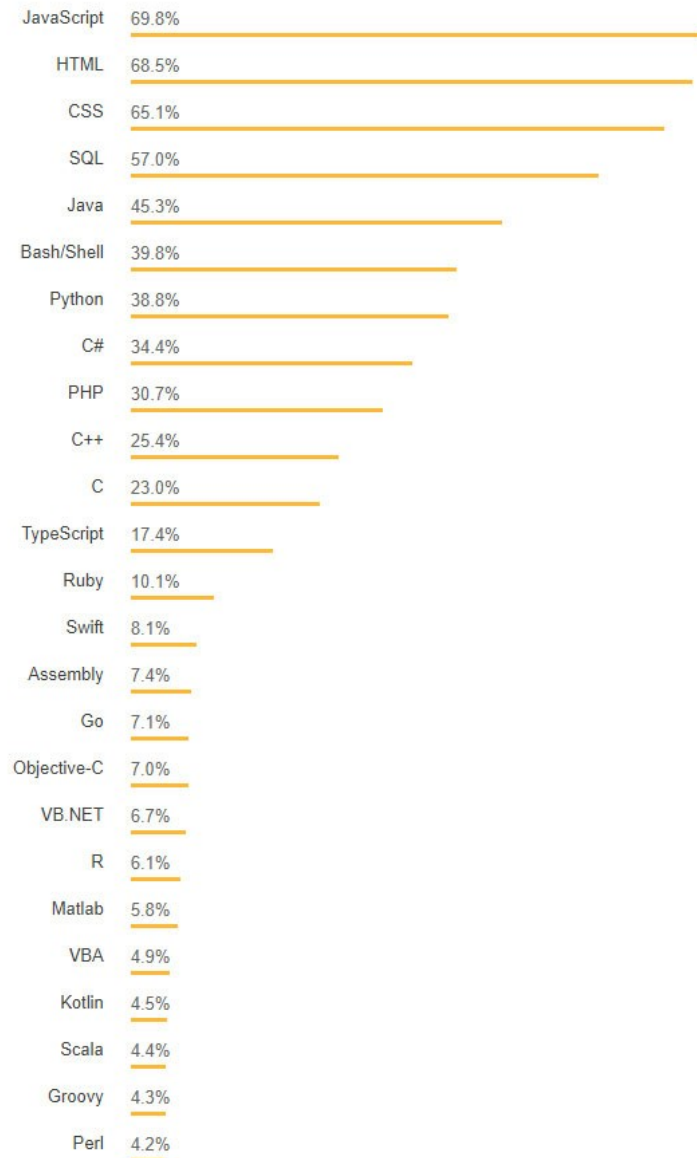
Doğacan Özdemir 2023

1. Giriş

Bilgisayarların hayatımıza girmesi ile birçok problem bilgisayarlar sayesinde çözülmeye başlamıştır. Bu duruma karşılık olarak ortaya birçok programlama dili çıkmıştır.

Programlama dili, yazılımcının bir algoritmayı ifade etmek amacıyla, bir bilgisayara ne yapmasını istediğini anlatmasının tektipleştirilmiş yoludur. Programlama dilleri, yazılımcının bilgisayara hangi veri üzerinde işlem yapacağını, verinin nasıl depolanıp iletileceğini, hangi koşullarda hangi işlemlerin yapılacağını tam olarak anlatmasını sağlar.

Şu ana kadar 250’den fazla programlama dili ortaya çıkmıştır. Bunlardan bazıları C, C#, C++, Java, JavaScript, Python, Swift, Kotlin gibi dillerdir.



Tablo-1 – Dünyadaki popüler programlama dilleri

1.1 Java Tarihçesi

Java, Sun Microsystems mühendislerinden James Gosling tarafından geliştirilmeye başlanmış açık kaynak kodlu, nesneye yönelik, platform bağımsız, yüksek verimli, çok işlevli, yüksek seviye, hem yorumlanan hem de derlenen bir dildir.

Java, Sun Microsystems'den James Gosling tarafından geliştirilen bir programlama dilidir ve 1995 yılında Sun Microsystems'in çekirdek bileşeni olarak piyasaya sürülmüştür. Bu dil C ve C++'dan birçok sözdizim türetmesine rağmen bu türevler daha basit nesne modeli ve daha az düşük seviye olanaklar içerir. Java uygulamaları bilgisayar mimarisine bağlı olmadan herhangi bir Java Sanal Makinesi (Java Virtual Machine - *JVM*) üzerinde çalışabilen tipik bytecode'dur (sınıf dosyası).

| Versiyon | Yayın Tarihi |
|------------------|--------------------|
| JDK Beta | 1995 |
| JDK1.0 | 23 Ocak 1996 |
| JDK 1.1 | 19 Şubat 1997 |
| J2SE 1.2 | 8 Aralık 1998 |
| J2SE 1.3 | 8 Mayıs 2000 |
| J2SE 1.4 | 6 Şubat 2002 |
| J2SE 5.0 | 30 Eylül 2004 |
| Java SE 6 | 11 Aralık 2006 |
| Java SE 7 | 28 Temmuz 2011 |
| Java SE 8 (LTS) | 18 Mart 2014 |
| Java SE 9 | 21 Eylül 2017 |
| Java SE 10 | 20 Mart 2018 |
| Java SE 11 (LTS) | 25 Eylül 2018 |
| Java SE 12 | 19 Mart 2019 |
| Java SE 13 | 17 Eylül 2019 |
| Java SE 14 | 17 Mart 2020 |
| Java SE 15 | 15 Eylül 2020 |
| Java SE 16 | 16 Mart 2021 |
| Java SE 17 (LTS) | 14 Eylül 2021 |
| Java SE 18 | 22 Mart 2022 |
| Java SE 19 | 13 Eylül 2022 |
| Java SE 20 | 07 Haziran 2022 ** |
| Java SE 21 (LTS) | 07 Aralık 2022 ** |

James Gosling ve Patrick Naughton Java projesini Haziran 1991'de başlattı. Java ilk olarak interaktif televizyonlar için tasarlandı ancak dijital kablo televizyon endüstrisi için o zamanlar çok gelişmişti. Java'nın ilk hali Oak ismini taşıyordu ve bu ismi Gosling'in ofisinin hemen yanında bulunan bir meşe ağacından almıştı. Daha sonra projenin ismi *Green* oldu ve en son Java adını aldı. Gosling, Java'yı C/C++'a benzer bir syntax ile tasarladı ve böylece programcılar için kolaylıkla öğrenilebilen bir dil oldu.

Tablo-2 – Java sürümleri

1.2 Java Kullanım Alanları

Kullanımı ücretsiz ve çok yönlü bir dil olması nedeniyle Java, yerelleştirilmiş ve dağıtılmış yazılımlar oluşturmada kullanılmaktadır. Java'nın yaygın kullanım alanları aşağıdakileri içerir:

1.2.1 Oyun Geliştirme

Mobil oyunlar ve bilgisayar oyunları dâhil birçok popüler video oyunu Java'da oluşturulmaktadır. Makine öğrenimi veya sanal gerçeklik gibi gelişmiş teknolojilerin kullanıldığı modern oyunlar bile Java teknolojisiyle oluşturulmaktadır.

1.2.2 Bulut bilgi işlem

Java, WORA [Write Once and Run Anywhere (Bir Kez Yazın ve Her Yerde Çalıştırın)] felsefesine uygun yapısı sayesinde, merkezi olmayan bulut tabanlı uygulamalar için ideal seçimdir. Bulut sağlayıcıları, programlarını çok çeşitli platformlarda çalıştırmak için Java dilini seçmektedir.

1.2.3 Büyük Veri

Java, karmaşık veri kümeleri ve devasa miktarda gerçek zamanlı veri ile birlikte çalışabilecek veri işleme altyapıları için kullanılır.

1.2.4 Yapay Zeka

Java, geniş kapsamlı makine öğrenimi kitaplıkları sunar. Kararlı ve hızlı bir programlama dili olması nedeniyle doğal dil işleme ve derin öğrenme gibi yapay zekâ uygulaması geliştirme çalışmaları için ideal seçimdir.

1.2.5. Nesnelerin İnterneti(IoT)

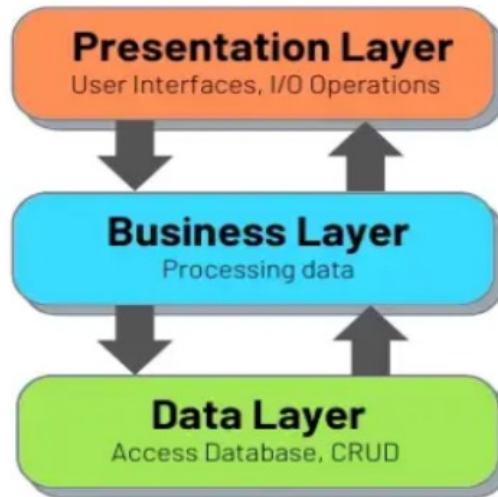
Java, bağımsız olarak internete bağlanabilen uç cihazlardaki sensörleri ve donanımları programlamak için kullanılmaktadır.

1.3 Katmanlı Mimari

Yazılım dünyasında sürdürülebilirlik çok önemlidir. Sürdürülebilirlikten kasıt kodumuzun değişime direnç göstermemesi, prensiplere uygun olması, temiz kod ile yazılması ve dokümantasyonu olmasıdır. Her ne kadar gelişigüzel kod yazarken asıl amacımız kodun çalışması olsa da aslında bir developer(geliştirici) için bu yanlıştır. Yazdığımız bir kodun, geliştirdiğimiz bir projenin okunabilirliği (readability), anlaşılabilirliği, tekrar kullanılabilirliği (reusability), bakım yapılabilirliği(maintainability) çok önemlidir. Bu yüzden projemizi belirli formatlarda geliştirmeliyiz. Bu açıdan mimari yaklaşımlardan yararlanılır.

Java ile uygulama geliştirilirken bazı durumlarda 3 katmanlı mimari kullanılır.

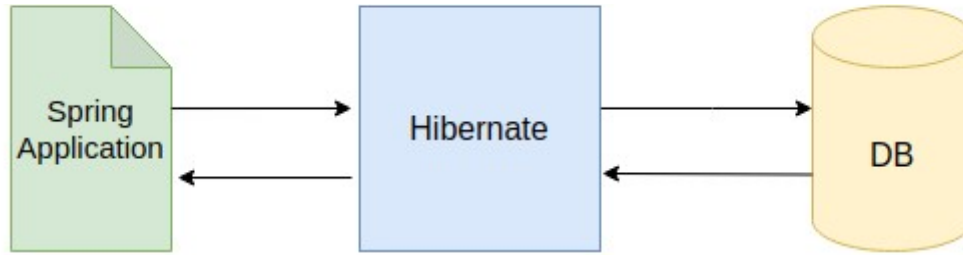
1.3.1 Katmanlı Mimarinin Katmanları



Şekil-2 – Katmanlı Mimari

1.3.1.1 Data Access Layer (Veri Ulaşım Katmanı)

Veri Ulaşım Katmanı'nda sadece veriye erişim işlemleri gerçekleştirilir. Yani veri tabanı bağlantıları bu katmanda gerçekleştirilir. Veri tabanı bağlantısının yanı sıra ekleme, silme, güncelleme ve veri tabanından veri çekme gibi işlemler bu katmanda yapılmaktadır. Bu katmanda iş kodları, cross-cutting işlemleri gibi işlemler gerçekleştirilmez.



Şekil-3 – Veri Erişim Katmanı

1.3.1.2 Business Layer (İş Katmanı)

İş Katmanı'nda ise sadece iş kodları yazılır yani iş kuralları burada yazılır. Ancak Business katmanı da Data Access Katmanından yararlanır. Data Access'te çektiğimiz verileri Business katmanda işleriz. Böylelikle projemizde bağımlılıkları ortadan kaldırmış, projenin okunabilirliği ve geliştirilebilirliği daha açık hale getirmiş oluruz.

1.3.1.3 Presentation Layer (Sunum Katmanı)

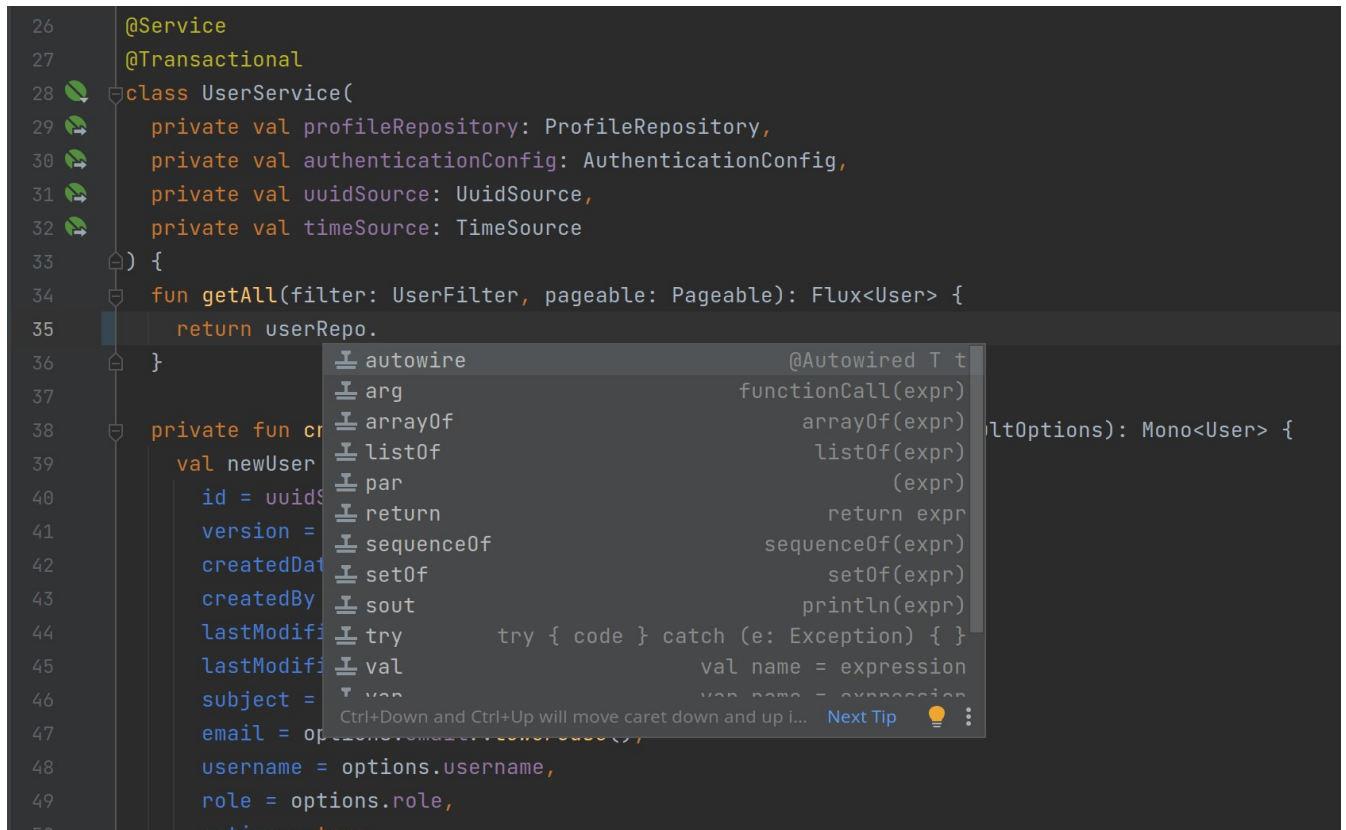
Sunum Katmanı ise MVC uygulaması, Windows Form uygulaması, Android Mobil Uygulama gibi uygulamaların arayüzkatmanıdır. Yani kullanıcıyla etkileşime geçilen işlemler bu katmanda gerçekleştirilir. Bu katmanda kullanıcıdan gelen verileri Businessile Data katmanına yönlendirir.

1.4 IntelliJ IDEA Geliştirme Ortamı

Jetbrains firmasının geliştirdiği Java kodlama için kullanılan IntelliJ IDEA' nın geliştirme yaparken birçok faydası vardır.

- Kolay ve anlaşılabilir arayüz
- Kütüphanelere internet üzerinden erişim sağlama
- Kod düzenleme konusunda ekstra bir programa gerek duymadan kısayol ile düzenleme yapabilme
- Öğrencilere ve eğitim üyelerine ücretsiz sunulması
- Servislere direkt erişim sağlayabilme
- Springboot ile eş zamanlı olarak çalışması

Bu faydalara rağmen bir eksi olarak fazla RAM kullanımı söylenebilir.

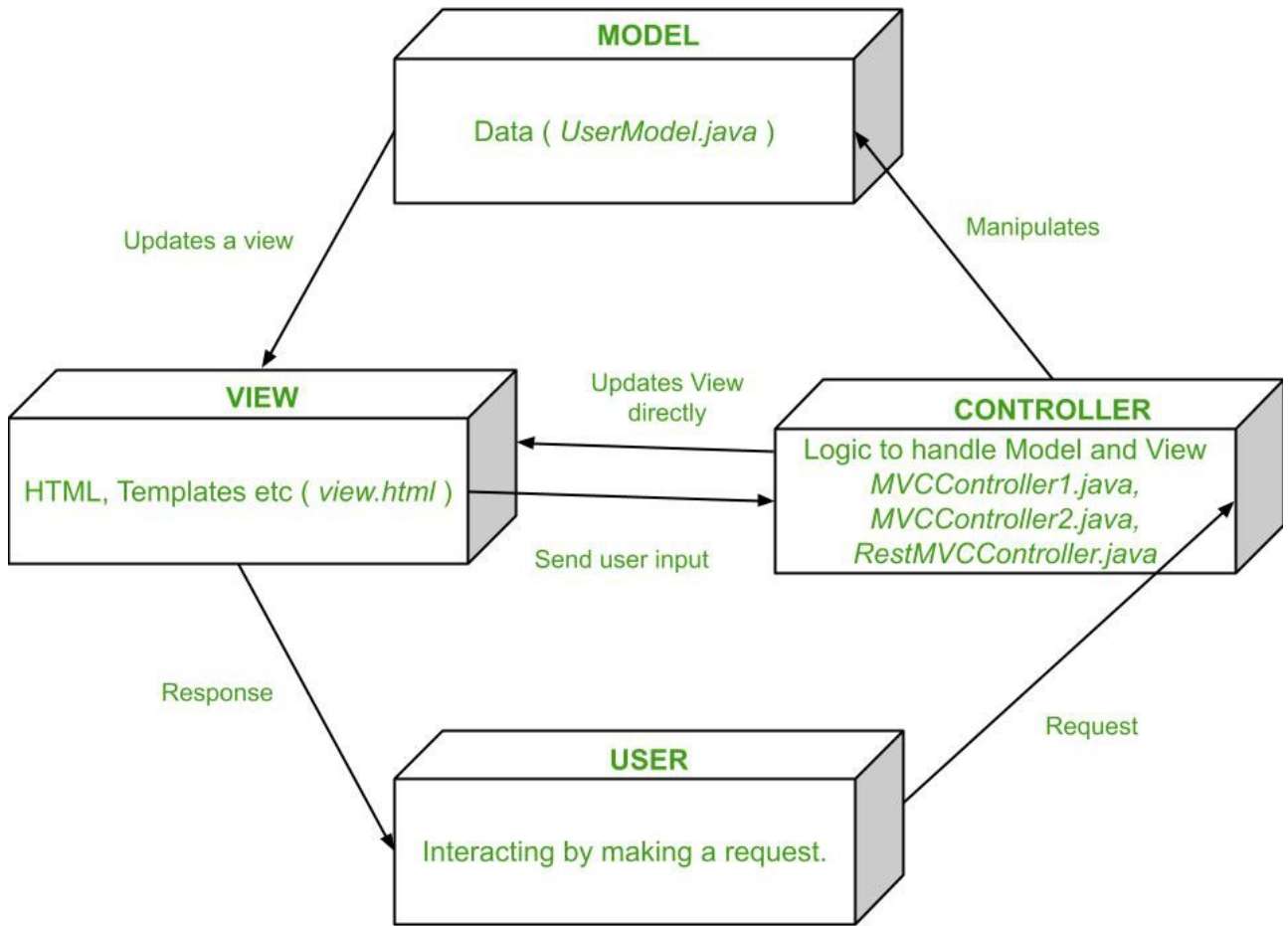


Şekil-4 – IntelliJ IDEA Çalışma Ortamı

2. Kullanılan Teknolojiler

2.1. Springboot Framework

Spring Framework Java için geliştirilmiş, açık kaynak olan bir uygulama geliştirme framework'üdür. Spring Framework'ün temel özellikleri herhangi bir Java uygulaması tarafından kullanılabilir. Eklentileri ile birlikte Java EE platformu üzerinde web uygulamaları geliştirmek için de kullanılabilir. Spring Framework Java toplulukları arasında Enterprise JavaBeans (EJB) modelinin yerine geçebilecek popüler bir alternatif haline gelmiştir.



Şekil-5 – Springboot Çalışma Katmanları

2.1.1. Springboot Anotasyonları

2.1.1.1 @SpringBootApplication Anotasyonu

@SpringBootApplication anotasyonu uygulamanın giriş metodunu belirtir. Yani halk arasındaki tabir ile main fonksiyondur. Uygulama bu metod ile başlar.

2.1.1.2. @Entity Anotasyonu

@Data ile belirtilen sınıf içerisindeki oluşturulan getter ve setter metotlarını veritabanı ile eşleştirir.

2.1.1.3. @Table Anotasyonu

İlgili veritabanındaki table ile eşleşmeyi sağlar. @Table(name = “tablename”) şeklinde kullanılır.

2.1.1.4. @Id Anotasyonu

İlgilin verinin veritabanı içerisinde Primary Key olarak tanımlanmasını sağlar.

2.1.1.5. @GeneratedValue Anotasyonu

Primary Key olarak tanımlanan verinin Auto Increment (Otomatik değer alması) olarak tanımlanmasını sağlar.

2.1.1.6 @Controller Anotasyonu

İlgili sınıfın bir Controller sınıfı olduğunu belirten anotasyondur.

2.1.1.7. @Autowired Anotasyonu

Controller sınıfındaki metotlar ile nesnelerin bağımlılığını sağlar.

2.1.1.8. @GetMapping Anotasyonu

İlgili metotun bağlanacağı katman ile mapping işlemini gerçekleştirir. @GetMapping(“/erişim yolu”) şeklinde tanımlanır.

2.1.1.9. @PostMapping Anotasyonu

İlgili metotun bağlanacağı katman ile mapping işlemini gerçekleştirir.

2.1.1.10. @RequestParam Anotasyonu

İlgili metotun gerçekleşebilmesi için bir parametreye ihtiyacı olduğunu belirten anotasyondur. @RequestParam(veri_tipi veri_adı) şeklinde tanımlanabilir.

2.1.1.11. @Repository Anotasyonu

Sorgulama ve filtreleme ihtiyaçları karşısında ortaya çıkmış bir anotasyondur. findByAll() gibi hazır metotların kullanımında yararlanır.

2.1.2. pom.xml Dosyası

POM, xml dosya formatında olan ve bir projenin build edilmesinden, nasıl edileceğine, bağımlılıklarından, packagingine kadar proje ile ilgili olan her türlü bilgiyi içerisinde barındıran bir dosyadır. Maven'in son versionunda pom.xml olarak adlandırılır, önceki versionlarında farklı isimlendirmeler vardı.

```
<!-- https://mvnrepository.com/artifact/org.thymeleaf/thymeleaf-spring5 -->
<dependency>
  <groupId>org.thymeleaf</groupId>
  <artifactId>thymeleaf-spring5</artifactId>
  <version>3.1.1.RELEASE</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.thymeleaf/thymeleaf -->
<dependency>
  <groupId>org.thymeleaf</groupId>
  <artifactId>thymeleaf</artifactId>
  <version>3.1.1.RELEASE</version>
</dependency>

<!-- https://mvnrepository.com/artifact/javax.servlet/javax.servlet-api -->
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>javax.servlet-api</artifactId>
  <version>4.0.1</version>
  <scope>provided</scope>
</dependency>
```

Şekil-6 – pom.xml Dosyası Örneği

POM içerisinde bazı tag'ler kullanılır :

- **Packaging** : Paketleme tipini belirtir, default olarak jar'dır. Jar, War, Rar vs gibi tipler kullanılabilir.
- **Name** : Projeye isim vermek içindir. Verilmesinden yanayım.
- **URL** : Proje'nin adres niteliğindedir.
- **Dependencies** : Uygulama bağımlılıklarının tanımlandığı kısımdır.
- **Scope** : Proje ortamını belirler, compile, test, runtime olabilir.

2.1.3. application.properties Dosyası

Spring Boot varsayılan olarak yapılandırma ayarlarını *src/main/resources* dizini altındaki *application.properties* dosyasında tutar. Properties dosyaları kolay okunabilirlik ve yazım açısından oldukça kolay bir formattadır. Ancak, Spring Boot YAML formatına da destek verir ve iki açıdan da daha büyük kolaylık sağlar.

Properties için; *application.properties*, YAML için ise; *application.yml* dosyaları varsayılan olarak Spring tarafından tanınır.

Bu, konfigürasyonlarınızın ortam bazlı dinamik olarak yapılandırılabilmesi ve çoklu ortam kurulumları için çok kullanışlıdır.

```
spring.datasource.url = jdbc:mysql://localhost:3306/demoProdentDB
spring.datasource.username = admin
spring.datasource.password = Dogacan.228
server.port =8081
spring.jpa.show-sql = true
spring.jpa.hibernate.ddl-auto = update
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5Dialect
spring.security.user.name=admin
spring.security.user.password=12345
```

Şekil-7 – application.properties Dosyası Örneği

Şekil-7’deki bazı komutların amaçları:

- **spring.datasource.url** : Veritabanının yolunu gösterir.
- **spring.datasource.username** : Veritabana erişim için gerekli olan kullanıcı adını gösterir.
- **spring.datasource.password** : Veritabanına erişim için gerekli olan şifreyi gösterir.
- **server.port** : Tomcat Server’ın çalışacağı localhost adresini gösterir.
- **spring.security.user.name** : Spring Security’nin auth için kullandığı kullanıcı adını tanımlar.
- **spring.security.user.password** : Spring Security’nin auth için kullandığı şifreyi tanımlar. (Bu komut olmaması durumunda Spring Security rastgele bir şifre tanımlayacaktır.)

2.1.2. Spring Security

Spring framework kullanılarak geliştirilen doğrulama(Authentication), yetkilendirme(Authorization), şifreleme>Password Encoder) ve CSRF gibi güvenlik önlemleri sağlayan, Spring platformunda yer alan bir projedir.

Kullanıcılar uygulamalardan beklenen işlevin yanından kullanıcıların sahip olduğu yetkilere(yönetim, editör veya üye) göre işlem yapması istenebilir.

Uygulama bu ihtiyaç ile birlikte doğrulama, yetkilendirme ve saldırganın yetkileri elde ederek uygulama işlevini değiştirmesini önleme gibi güvenlik tedbirlerine gereksinim duyulur.

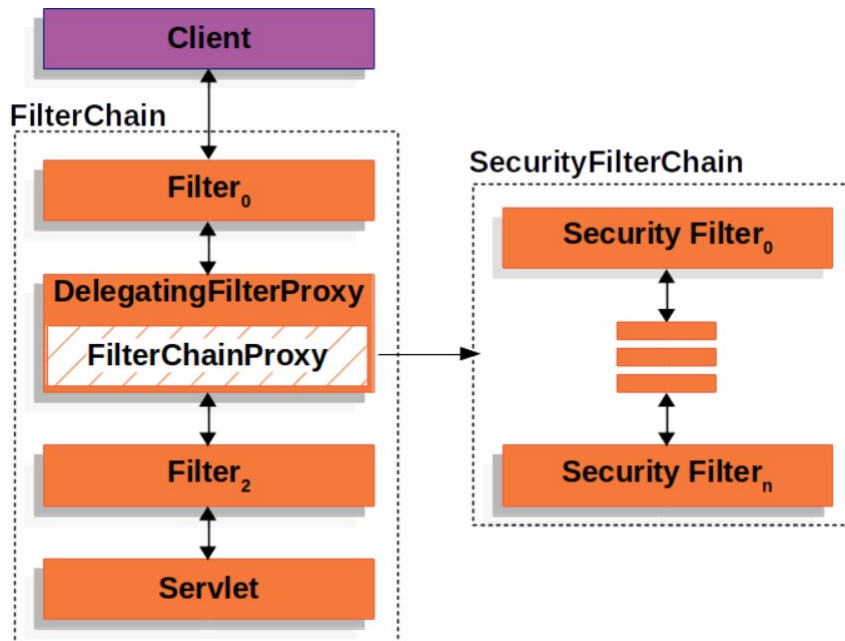
Her bir gereksinim içerisinde başka gereksinimlere ihtiyaç duyabilir.

Örneğin; Doğrulama sıradan bir dosyada yer alan kullanıcı bilgilerinin kontrolü, veritabanından kontrolü veya LDAP gibi protokollerden kontrolü olabilir.

Temel doğrulama ve yetkilendirmenin yanında saldırganlar sürekli olarak farklı saldırı yöntemleri kullanılarak güvenlik açıkları ortaya çıkarmaktadır.

Tüm bu güvenlik gereksinimleri uygulamadan beklenen işlevselliği yerine getirmek için kullanılan efor kadar efora neden olacaktır.

Spring Security doğrulama, yetkilendirme, şifreleme, güvenlik önlemlerini esnek, kolay ve sürekli olarak paketlerin güncellenmesi ile sağlar.



Şekil-8 – Spring Security Çalışma Katmanları

2.2. MySQL Veritabanı

MySQL, altı milyondan fazla sistemde yüklü bulunan çoklu iş parçacıklı (İng. İngilizce: multi-threaded), çok kullanıcı (İng. İngilizce: multi-user), hızlı ve sağlam (İng. İngilizce: robust) bir veri tabanı yönetim sistemidir.

UNIX, OS/2 ve Windows platformları için ücretsiz dağıtılmakla birlikte ticari lisans kullanmak isteyenler için de ücretli bir lisans seçeneği de mevcuttur. Linux altında daha hızlı bir performans sergilemektedir. Kaynak kodu açık olan MySQL'in pek çok platform için çalıştırılabilir ikilik kod halindeki indirilebilir sürümleri de mevcuttur. Ayrıca ODBC sürücülerini de bulunduğu için birçok geliştirme platformunda rahatlıkla kullanılabilir.

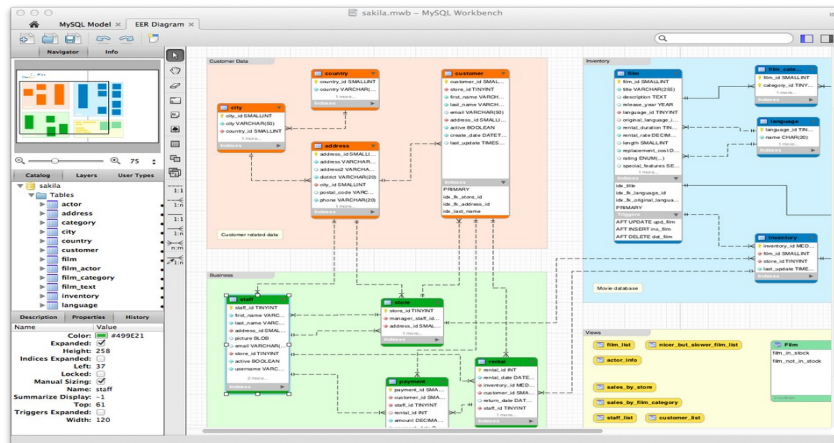
Geliştiricileri, 500'den fazlası 7.000.000 kayıt içeren 10.000 tablodan oluşan kendi veritabanlarını (100 gigabyte civarında veri) MySQL'de tuttuklarını söylüyorlar.

Web sunucularında en çok kullanılan veri tabanı olup ASP, PHP gibi birçok Web programlama dili ile kullanılabilir.

MySQL aşağıdaki veritabanı nesnelerini desteklemektedir.

- Tables (tablolar)
- Views (görüntü(leme)ler)
- Procedures (prosedürler)
- Triggers (tetikler)
- Cursors (imleçler)

Ayrıca sahip olduğu MySQL Workbench çalışma ortamı ile kolaylıkla veritabanı yönetimi yapmamızı sağlar.



Şekil-9 – MySQL Workbench Örneği

2.3. Bootstrap

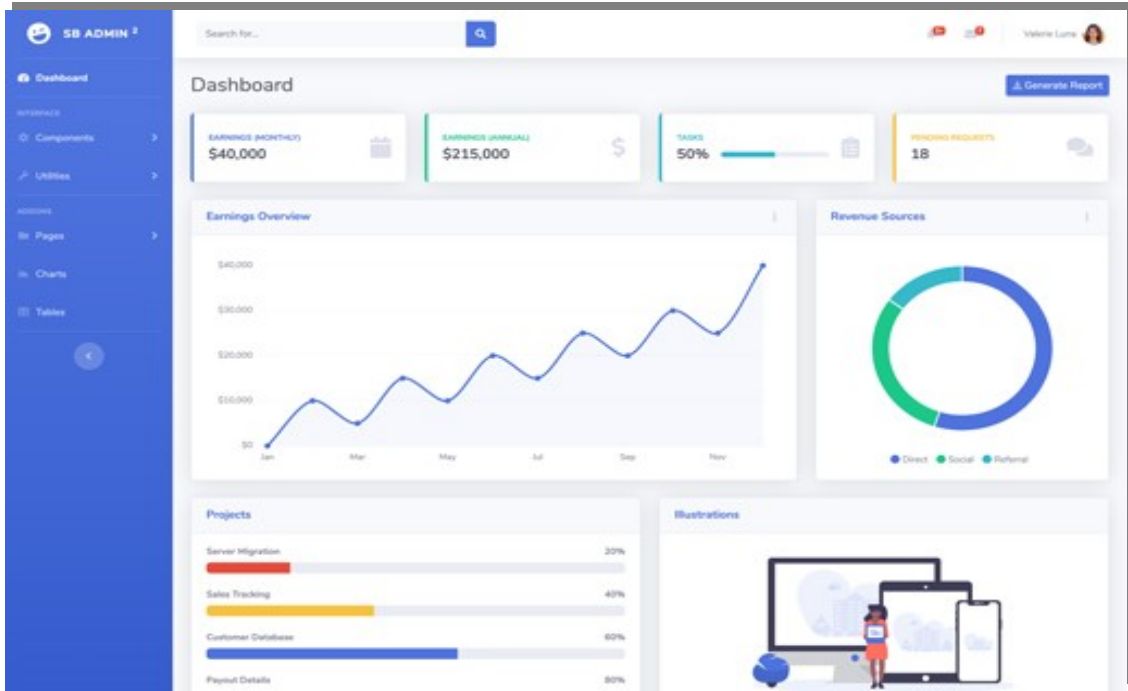
Bootstrap'i incelemeden önce, web uygulamalarının ön uç iskeletlerinin ne olduğuna yakından bakmakta fayda var. Bu iskeletler, bir uygulamanın kullanıcıların etkileşime geçtiği arayüzüdür. Yapıyı HTML kodları meydana getirir, basamaklı stil şablonları görsel formatı oluşturur ve JavaScript kodları slaytlar, açılan menüler gibi dinamik elementleri mümkün kılar.

İskeletler, web sitelerinin üzerlerine inşa edildikleri temellerdir. Bir ön uç iskeleti ve araç takımı olan Bootstrap, Twitter mühendisleri Mark Otto ve Jacob Thornton tarafından geliştirilmiştir. Temel amacı web uygulamalarının hızlı, kullanışlı ve daha duyarlı hale getirilmesini kolaylaştırmaktır. Bootstrap günümüzde duyarlı ve mobil öncelikli web siteleri geliştirmek için en popüler CSS iskeleti olma unvanına sahiptir.

Bootstrap'in avantajları şunlardır :

- **Daha hızlı geliştirme:** Bootstrap'in en büyük avantajlarından bir tanesi web geliştirme projelerini hızlandırması ve bunu yaparken kaliteden ödün verilmesini gerektirmemesidir. Bootstrap olmadan duyarlı web sitesi ve uygulama geliştirmek çok uzun sürebilirken, bu araç takımı gereken süreyi bir hayli kısaltır.
- **Kolay kullanılabilir:** Giriş seviyesi HTML ve CSS bilgisine sahip olan herkes Bootstrap kullanarak web siteleri, uygulamalar ve temalar geliştirebilir.
- **Duyarlı tasarım:** Yukarıda bahsettiğimiz duyarlı tasarımları kolaylaştırması Bootstrap'in alametifarikalarından bir tanesi. Bootstrap ile geliştirilen duyarlı web siteleri tüm ekran boyutlarına göre otomatik adapte olur.
- **Tasarım devamlılığı:** Bootstrap ile tasarlanan bir web sayfası Firefox, Chrome, Edge, Opera ve Safari gibi tüm modern web tarayıcılarında aynı şekilde görünür.
- **Açık kaynaklı:** Bootstrap'in en iyi özelliği açık kaynaklı ve tamamen ücretsiz olmasıdır. Birçok geliştirici bu platform üzerinde geliştirdiği kaynakları ücretli ve ücretsiz olarak diğerlerinin kullanıma sunar. Bu kaynaklar web siteleri ve uygulamaları geliştirmek için kullanılabilir.

- **Özelleştirme:** Bootstrap, web sitenizde ve uygulamanızda olduğu gibi kullanabileceğiniz ön yüklü bileşenlerle gelir. Navigasyon barları, açılır kapanır menüler, butonlar ve dahası gibi tasarım öğelerinden istediğinizi seçip kullanabilirsiniz. Bootstrap'in web sitesinden araç takımını indirmeden önce bazı özellikleri seçmeniz ve yalnızca istediklerinizi indirip kullanmaya başlamanız da mümkündür.
- **Dokümantasyon:** Bootstrap'i keşfetmek istiyorsanız, web sitesinde bulunan ve her kod parçası için tanımlayıcı ve açıklayıcı içerikler sunan dokümanlarından faydalanabilirsiniz. Bu açıklamalara eklenen kod örnekleri, yeni başlayan geliştiricilerin işini önemli ölçüde kolaylaştırır.
- **Temalar:** Bootstrap temelli geliştirilen web temaları yeni bir pazar haline dönüştü. Geliştiriciler ve tasarımcılar tarafından tasarlanan Bootstrap tabanlı şablonları alıp doğrudan kendi projelerinizde kullanabilir ya da kendi projelerinizi satışa çıkarabilirsiniz.



Şekil-10 – Bootstrap ile Oluşturulmuş Sayfa Örneği

2.4. Thymeleaf

Thymeleaf; açık kaynak kodlu ve kendini kanıtlamış, eklentiler (dialects) ile özelleştirilebilen, Spring Boot ve diğer Spring Framework projeleriyle tam uyumlu bir şablon motorudur.

Bunların yanı sıra, benim tercih etmemdeki en etkili sebep; XML ad uzayları ile direkt HTML etiketine uygulanması, dolayısıyla da sunum tarafındaki gereksiz kalabalıktan arındırması oldu.

2.5. Apache Tomcat Server

Apache Tomcat veya Tomcat Java tabanlı web uygulamalarını yayınlamak için kullanılan web sunucusudur.

Java, Java EE veya Java Teknolojileri içerisinde Java Servlet, JavaServer Pages, Java Expression Language, Java WebSocket gibi çeşitli teknolojiler yer alır.

Bu teknolojiler JCP (Java Community Process) olarak adlandırılan ve genellikle çeşitli firmalardaki geliştiriciler tarafından standart olarak belirlenir.

Standartlar belirlendikten sonra bu teknolojilerin kullanılabilmesi için bu standartların kodlara dökülmesi-implement gerekir.

Apache Tomcat bu standartları uygulayan ve içerisinde web sunucusu yer alan bir Java uygulamasıdır.

2.6. Hibernate

Hibernate Java geliştiriciler için geliştirilmiş bir ORM kütüphanesidir. Nesne yönelimli modellere göre veritabanı ile olan ilişkiyi sağlayarak, veritabanı üzerinde yapılan işlemleri kolaylaştırmakla birlikte kurulan yapıyı da sağlamlaştırmaktadır.

Nesneye yönelik yazılım ve ilişkisel veritabanı kullanımı günümüzde oldukça yaygındır. Bu iki gözde modelin belkide en önemli problemi en az onlar kadar yaygın ve dahası önü açık olan kuruluş uygulamaları(enterprise applications) ile birlikte kullanıldıklarında oldukça karışık, yorucu ve zaman alıcı olmalarıdır. Hibernate bir nesne/ilişkisel eşleme (Object/Relational Mapping) aracıdır. Burada nesne/ilişkisel eşleme terimi nesne modelindeki veri tanımlarının ilişkisel veri modeline eşleme (mapping) tekniğini ifade etmektedir.

Hibernate yalnızca Java sınıflarından veritabanı tablolarına veya Java veri tiplerinde SQL veri tiplerine dönüşümü yapmaz. Hibernate veri sorgulama(data query) ve veri çekme(data retrieval) işlemlerini de kullanıcı için sağlar. Bu özellikleriyle Hibernate geliştirme kolaylığı ve zamandan kazanç sağlar. Hibernate kullanımı olmadan tüm adı anılan işlemler için SQL ve JDBC'nin olanaklarından faydalanılarak el ile(manual) veri işleme(data handling) gerçekleştirilmesi zaruri olacaktır.

Hibernate genel anlamda Java sınıflarından veritabanı tablolarına dönüşümü ya da Java veri tiplerinden SQL veri tiplerine dönüşümü gerçekleştirir. Ayrıca veri sorgulama ve veri çekme işlemlerini de kullanıcı için sağlar. Bu özellikleriyle Hibernate uygulamaların geliştirilme aşamasında çok büyük kolaylık ve zamandan kazanç sağlar. Hibernate kullanmadan JDBC ile veri tabanına erişmek mümkündür. Ancak veri tabanındaki tablo sayısı arttığında buna bağlı olarak tablolar arası ilişkiler de artacaktır. Uygulama büyüdükçe bu ilişkiler çok karmaşık bir hal alabilir. Veritabanı işlemleri için connection açma kapama, ilişkili tablolar için çok karmaşık SQL sorguları yazma, aynı fonksiyon içinde birden fazla connection açmama gibi dikkat etmemiz gereken işler artacaktır. Bu işlemleri yaparken yapacağımız en ufak hata uygulamanın tümünü etkileyecektir. Uygulamamızın mimarisi ne kadar düzgün olursa yapısı da bir o kadar karmaşık olacaktır.

2.7.Lombok

Lombok, Java projesi geliştirirken IDE'ye entegre edilebilen bir anotasyon ile kod üretme (code generation) kütüphanesidir. Lombok ile daha temiz ve daha az kod yazmış oluruz.

Java'da proje geliştirirken yaygın olarak yapmamız gereken bazı işlemler bulunmakta. Bunlar projemizin iş tarafına gerçek bir değer getirmez iken kodumuzun çok fazla ayrıntı barındırmasını zorunlu hale getiriyor. Bu durumlarda Lombok ile çözümü sağlayabiliriz.

2.8. Maven

Maven genellikle Java platformunda yer alan komutların derlenmesi sırasında kullanılan otomasyon ve inşa aracıdır.

Java programlama dili ile uygulama geliştirirken çeşitli kütüphaneler kullanmak isteyebiliriz. Örneğin; Java ile PDF dosyası oluşturmak için Apache PDFBox, iText, JPOD gibi çeşitli kütüphaneleri kullanabiliriz.

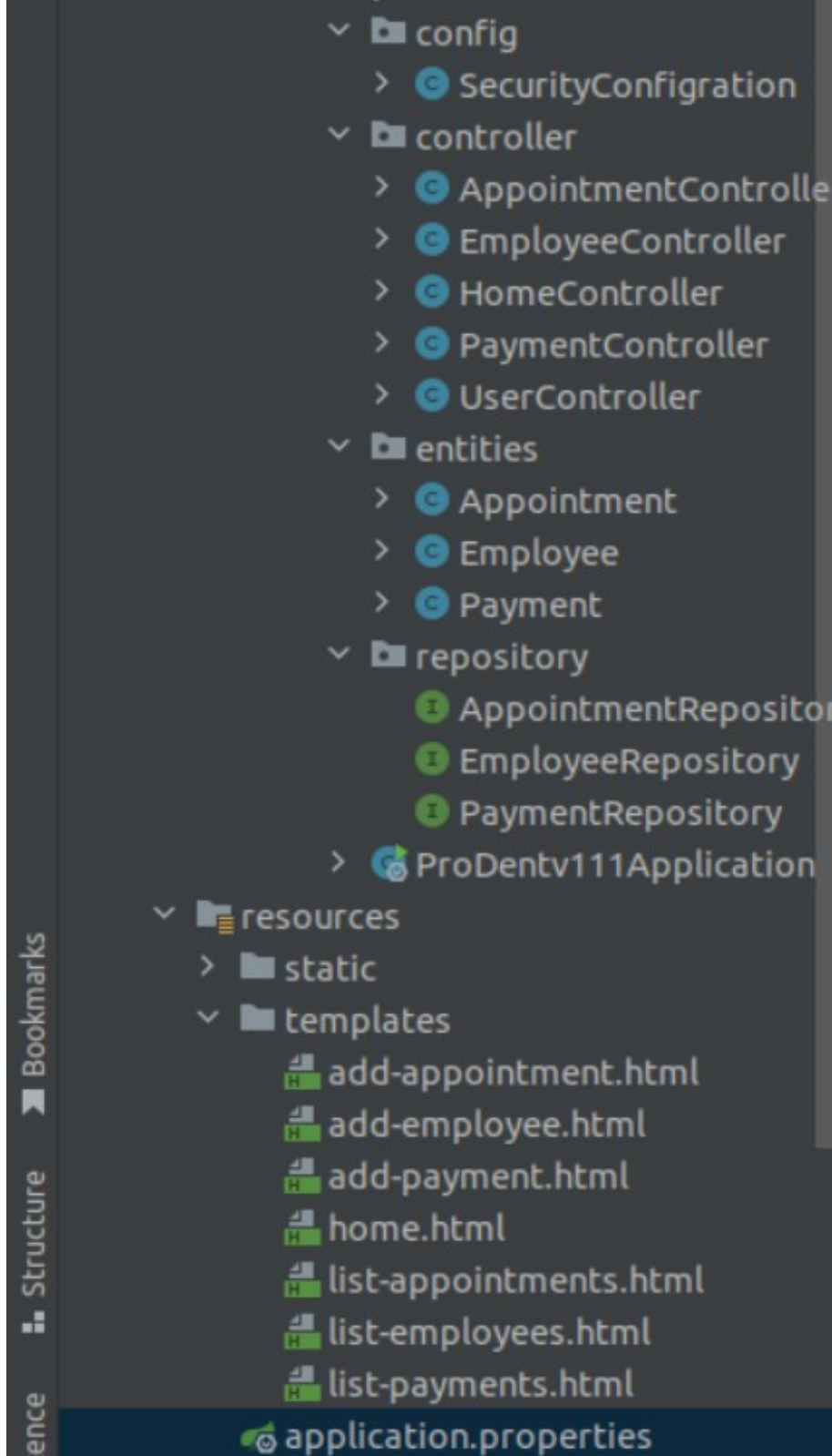
Her kütüphane için gerekli olan JAR dosyalarını indirmek ve projeye uygun olarak yerleştirmek (classpath) gerekir. Ancak sadece kütüphanelerin indirilmesi ve projeye dahil edilmesi yetmeyecektir. Ayrıca her yeni güncelleme sonrası güncel dosyaların takip edilmesi gerekecektir.

Maven proje dosyasına eklenen bağımlılıklar ile kolay bir şekilde indirmeyi ve proje yerleştirmeyi sağlar.Kullanılan kütüphaneler proje dosyasında yer aldığından taşınabilirlik sağlanmış olur.

Sunmuş olduğu dizin yapısı sayesinde diğer geliştiricilerin projeyi takibini kolaylaştır.

3. Java ile Veteriner Kliniği Otomasyonu Detayları

3.1. Dosya Yolları



Şekil-11 – Uygulamanın Dosya Yolları ve Düzeni

3.2. Proje Mimarisi

3.2.1. Config Paketi

Bu paket içerisinde SecurityConfiguration.java dosyası ile login özelliklerini tanımlanmıştır. Spring Security framework'u ile gerekli bağlantılar sağlanmıştır.

```
@EnableWebSecurity
public class SecurityConfiguration extends WebSecurityConfigurerAdapter {
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.httpBasic();
        //Dashboard yoluna gelen yolda izin sağlamaya yarar.
        http.authorizeHttpRequests().antMatchers("/") .authorizeHttpRequests(Configurer<...>.AuthorizedUrl
            .authenticated() .authorizeHttpRequests(Configurer<...>.AuthorizationManagerRequestMatcherRegistry
                .and() HttpSecurity
                .authorizeHttpRequests().antMatchers("/dashboard").permitAll();
    }
}
```

Şekil-12 – SecurityConfiguration.java içeriği

3.2.2. Repository Paketi

Bu paket içerisinde AppointmentRepository.java, EmployeeRepository.java, PaymentRepository.java dosyaları bulunmaktadır. İlgili dosyalar java.util kütüphanesinden Repository metodunu kullanarak bir Controller katmanında bir repo oluşturmamızı sağlar.

```
2 usages
@Repository
public interface EmployeeRepository extends JpaRepository<Employee, Long>{
```

Şekil-13 – EmployeeRepository.java İçeriği

3.2.3. Entities Paketi

Bu paket içerisinde *Appointment.java*, *Employee.java*, *Payment.java*, dosyaları bulunmaktadır. İlgili dosyalar gerekli nesnelerin oluşumunu ve *get()*, *set()* edilmesini sağlar. Ayrıca veritabanının ilgili kısmında tabloların oluşturulmasını sağlar.

```
@Entity
@Table(name="musteriler")
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Employee {
    no usages
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long musterID;
    no usages
    private String musterIDc;
    no usages
    private String musterAd;
    no usages
    private String musterSoyad;
    no usages
    private String musterAdres;
    no usages
    private String musterTelefon;
    no usages
    private String musterDogumTarihi;
    no usages
    private char musterCinsiyet;
    no usages
    private String musterKanGrubu;
}
```

Şekil-14 – Employee.java İçeriği

3.2.4. Controller Paketi

Bu paket altında *AppointmentController.java*, *PaymentController.java*, *EmployeeController.java*, *HomeController.java*, *UserController.java* dosyaları bulunmaktadır. İlgili dosyalar gerekli Mapping işlemleri ve veri getirme işlemlerini sağlamaktadır. *HomeController.java* dosyası altında Ana Sayfa'nın Mapping işlemleri tanımlanmıştır. *UserController.java* dosyası altında Spring Security ile oluşturduğumuz Login Form'un uygun çalışıp çalışmadığını kontrol ettiğimiz bir metod bulunmaktadır.

```
@Controller
public class EmployeeController {
    /* Autowired anotasyonu ile bağımlılığı sağladık. */
    4 usages
    @Autowired
    private EmployeeRepository eRepo;

    /* Tüm müşterilerin listelenmesi için gerekli olan mapping işlemini uyguladık.
    @GetMapping anotasyonu gerekli link bağlantısı sağlıyor.
    Springboot'un kendi kütüphanesinde bulunan ModelAndView bir yer tutucu görevi görüyor ve bizi ulaşmak istediğimiz
    web sayfasına yönlendirip listeleme işlemini yapıyoruz. */

    no usages
    @GetMapping("/{allEmployees}")
    public ModelAndView getAllEmployees()
    {
        ModelAndView mav = new ModelAndView("list-employees");
        List<Employee> list = eRepo.findAll();
        mav.addObject("employees",list);
        return mav;
    }

    /* Müşteri ekleme kısmını tanımladık. add-employee.html sayfasındaki hidden olarak tanımladığımız buton ile
    hem kaydedip hem değiştirme işlemini bu metod üzerinden yapabiliyoruz. */
}
```

Şekil-15-1 – EmployeeController.java İçeriği

```
/* Müşteri silme işlemini tanımladık. mySQL'den musteriiD kolonuna göre silme işlemi gerçekleştiriyor.
İşlem tamamlandıktan sonra tüm müşterilerin listelendiği sayfaya yönlendirme yapıyor. */
no usages
@GetMapping("/{deleteEmployee}")
public String deleteEmployee(@RequestParam Long musteriiD)
{
    eRepo.deleteById(musteriiD);
    return "redirect:/allEmployees";
}
```

Şekil-15-2 – EmployeeController.java İçeriği

```

@GetMapping("/{addEmployee}")
public ModelAndView addEmployee()
{
    ModelAndView mav = new ModelAndView("add-employee");
    Employee yeniMusteri = new Employee();
    mav.addObject("employee", yeniMusteri);
    return mav;
}

/* @PostMapping anotasyonu ile mySQL'e kayıt işlemini bir buton vasıtasıyla yaptırıp tüm müşteriler sayfamıza
yönlendirme yaptık.
*/
no usages
@PostMapping("/{saveEmployee}")
public String saveEmployee(@ModelAttribute Employee employee)
{
    eRepo.save(employee);
    return "redirect:/allEmployees";
}

/* @RequestParam metodu ile mySQL'den müşteriID bağlantısını sağlayıp gerekli güncelleme işlemlerini tanımladık.
add-employee.html sayfasındaki hidden olarak tanımladığımız buton ile hem kaydedip hem değiştirme
işlemini bu metod üzerinden yapabiliyoruz. Raporda daha detaylı bilgiler bulunmaktadır.
*/
no usages
@GetMapping("/{updateEmployee}")
public ModelAndView updateEmployee(@RequestParam Long musterIID)
{
    ModelAndView mav = new ModelAndView("add-employee");
    Employee employee = eRepo.findById(musterIID).get();
    mav.addObject("employee", employee);
    return mav;
}

```

Şekil-15-3 – EmployeeController.java İçeriği

```

@Controller
public class HomeController {
    @GetMapping(value = {"/home", "/"})
    public String home() { return "home"; }
}

```

Şekil-16 – HomeController.java İçeriği

```
@RestController
public class UserController {
    no usages
    @GetMapping("/dashboard")
    public String dashboard(){
        return "login başarılı";
    }
}
```

Şekil-17 – UserController.java İçeriği

3.2.5. Resources Paketi

Bu paket içerisinde temel görünüm öğelerimiz bulunmaktadır. Login sayfası Spring Security'nin hazır paketi olan bir sayfa şeklinde dönmektedir.

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  
  <a class="navbar-brand" th:href="@{/home}">ProDENT Clinic Professional </a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav"
    aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbarNav">
    <ul class="nav nav-pills nav-fill">
      <li class="nav-item active">
        <a class="nav-link" th:href="@{/home}">Ana Sayfa <span class="sr-only">(current)</span></a>
      </li>
      <li class="nav-item">
        <a class="nav-link" th:href="@{/allEmployees}">Müşteriler</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" th:href="@{/allPayments}">Ödemeler</a>
      </li>
      <li class="nav-item">
        <a class="nav-link nav-justified" th:href="@{/allAppointments}">Randevular</a>
      </li>
    </ul>
  </div>
</nav>
```

Şekil-17 – add-employees.html (Navbar) İçeriği

Şekil-17’de bulunan kod dizisi tüm HTML sayfalarında bulunmaktadır. Navigation Bar tanımını yapıldığı kod dizisidir. Tasarım için Bootstrap 4 kullanılmıştır. Bağlantılar için Thymeleaf kullanılmıştır. Kodların devamı kaynak dosya içerisinde mevcuttur.

```
<div class = "container">
  <h2>Müşteri Ekle</h2>
  <hr/>
  <!-- Bu kısımda tablomuzu oluşturduk ve bağlantılar ile ilgili olan parametreleri mySQL'e kaydediyoruz. -->
  <form th:action="@{/saveEmployee}" th:object = "${employee}" method="POST">
    <input type="text" th:field="*{musteriTc}" placeholder="T.C Kimlik Numarası"
      class="form-control col-4 mb-4">
    <input type="text" th:field="*{musteriAd}" placeholder="İsim"
      class="form-control col-4 mb-4">
    <input type="text" th:field="*{musteriSoyad}" placeholder="Soyisim"
      class="form-control col-4 mb-4">
    <input type="text" th:field="*{musteriAdres}" placeholder="Adres"
      class="form-control col-4 mb-4">
    <input type="text" th:field="*{musteriTelefon}" placeholder="Telefon"
      class="form-control col-4 mb-4">
    <input type="text" th:field="*{musteriCinsiyet}" placeholder="Cinsiyet"
      class="form-control col-4 mb-4">
    <input type="text" th:field="*{musteriDogumTarihi}" placeholder="Doğum Tarihi"
      class="form-control col-4 mb-4">
    <input type="text" th:field="*{musteriKanGrubu}" placeholder="Kan Grubu"
      class="form-control col-4 mb-4">
    <button class="btn btn-primary" type="submit">Kaydet</button>
    <!-- Bu kısım hem kaydedip hem değiştirmeye yarıyor. musterController bağlantısı ile repository özelliği olarak
    hem güncelleme hem de kayıt işlemi için aynı buton kullanılabilir -->
    <input type="hidden" th:field="*{musteriID}" />
  </form>
  <hr/>
  <!-- Bu buton ile tüm müşterilerin olduğu sayfaya href bağlantısı ile geri dönüş sağladık. -->
  <a th:href="@{/allEmployees}">Geri Dön</a>
```

Şekil-18 – add-employees.html (body) İçeriği

Şekil-17’de bulunan kod dizisinin benzeri tüm HTML sayfalarında bulunmaktadır. Table tanımlamalarımız kullanıldığı kısımdır. Tasarım için Bootstrap 4 kullanılmıştır. Veri ekleme bağlantıları için Thymeleaf kullanılmıştır. Kodların devamı kaynak dosya içerisinde mevcuttur.


```

<table class="table table-bordered table-striped" id="musteri-table">
  <thead>
    <!-- Bu kısımda tablomuzu oluşturduk. -->
    <tr>
      <th>T.C. Kimlik Numarası</th>
      <th>İsim</th>
      <th>Soyisim</th>
      <th>Adres</th>
      <th>Telefon</th>
      <th>Cinsiyet</th>
      <th>Doğum Tarihi</th>
      <th>Kan Grubu</th>
      <th>İşlemler</th>
    </thead>
    <tbody>
      <!-- Bağlantılar ile ilgili olan parametreleri mySQL'den getiriyoruz. -->
      <tr th:each="employee : ${employees}">
        <td th:text = "${employee.musteriTc}"></td>
        <td th:text = "${employee.musteriAd}"></td>
        <td th:text = "${employee.musteriSoyad}"></td>
        <td th:text = "${employee.musteriAdres}"></td>
        <td th:text = "${employee.musteriTelefon}"></td>
        <td th:text = "${employee.musteriCinsiyet}"></td>
        <td th:text = "${employee.musteriDogumTarihi}"></td>
        <td th:text = "${employee.musteriKanGrubu}"></td>
        <td>
          <!--Değiştirme ve silme butonlarını tanımladık-->
          <a th:href="@{/updateEmployee(musteriID=${employee.musteriID})}"
            class="btn btn-info">Değiştir</a>
          <a th:href="@{/deleteEmployee(musteriID=${employee.musteriID})}"
            class="btn btn-danger ml-2">Sil</a>
        </td>
      </tr>
    </tbody>
  </table>

```

Şekil-19 – list-employees.html (body) İçeriği

Şekil-18’de bulunan kod dizisi *list-payments.html* ve *list-appointment.html* sayfalarında da bulunmaktadır. Table tanımlamalarımız kullanıldığı kısımdır. JQuery framework’unun DataTable() eklentisi kullanılarak tablolar oluşturulmuştur. Tasarım için Bootstrap 4 kullanılmıştır. Veri bağlantıları için Thymeleaf kullanılmıştır. Kodların devamı kaynak dosya içerisinde mevcuttur.

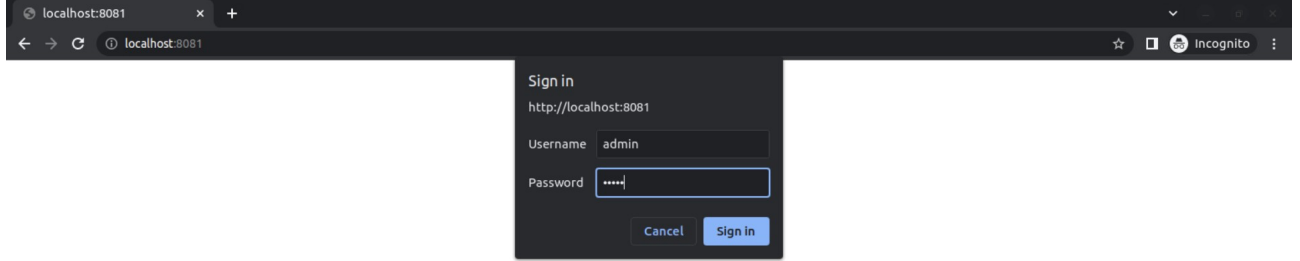
```
<!-- jQuery Datatables kurulumunu kaynağından yükledik.-->
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script type="text/javascript" src="https://cdn.datatables.net/v/bs4/dt-1.13.1/datatables.min.js"></script>
<!-- Tablomuzu jQuery Datatable ile bağladık.-->
<script>
    $(document).ready(function() {
        $('#musteri-table').DataTable({
            //Bu kısımda İşlemler kolonunun sıralama yapmasını kapattık.
            'aoColumnDefs' : [{
                'bSortable': false,
                'aTargets' : [-1] //-1 indexi İşlemler kolonunun indexidir.
            }]
        });
    })
</script>
```

Şekil-19-2 – list-employees.html (body) İçeriği

Şekil-19-2’de bulunan kod dizisi *list-payments.html* ve *list-appointment.html* sayfalarında da bulunmaktadır. JQuery ile DataTable() bağlantılarının yapıldığı kod dizisi ve tablonun son kolonu olan İşlemler kolonunun sıralanmasının (Sort) engellenmesini sağlayan JavaScript kodudur.

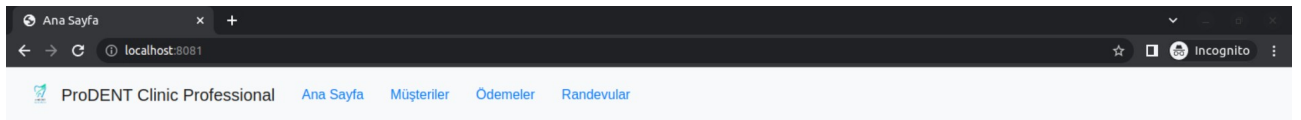
3.3. Tomcat Server Üzerinde Çalışır Hali

3.3.1. Login Sayfası



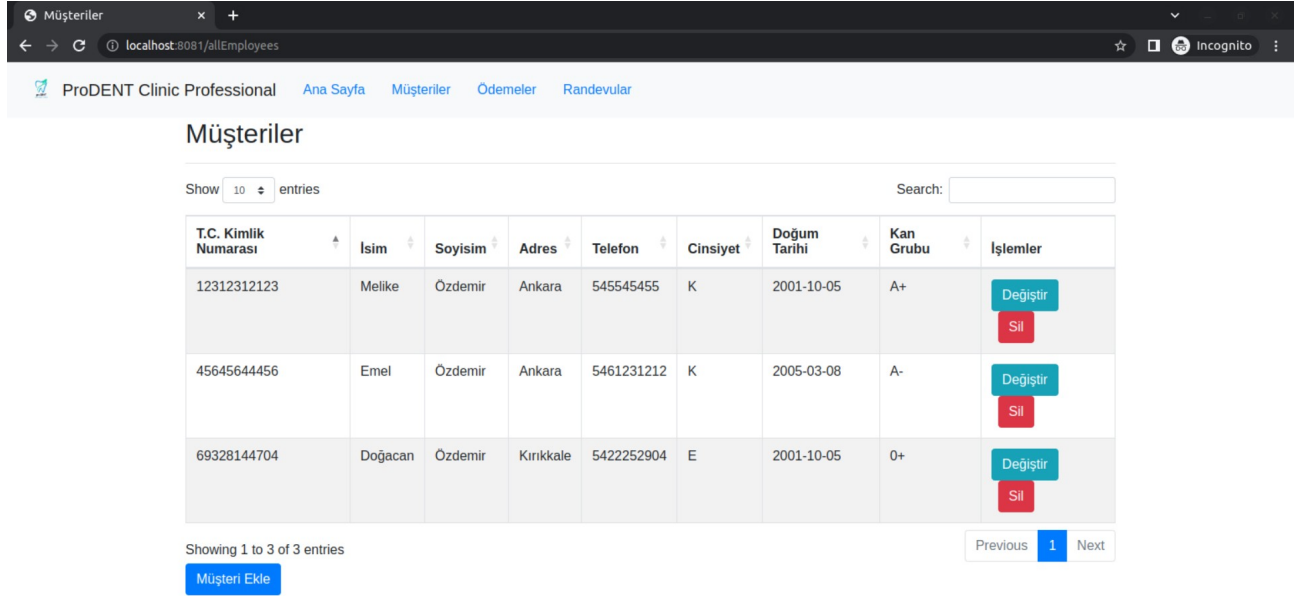
Şekil-20 – Login Sayfası

3.3.2. Ana Sayfa



Şekil-21 – Ana Sayfa

3.3.3. Müşteriler



| T.C. Kimlik Numarası | İsim | Soyisim | Adres | Telefon | Cinsiyet | Doğum Tarihi | Kan Grubu | İşlemler |
|----------------------|---------|---------|-----------|------------|----------|--------------|-----------|---|
| 12312312123 | Melike | Özdemir | Ankara | 545545455 | K | 2001-10-05 | A+ | Değiştir Sil |
| 45645644456 | Emel | Özdemir | Ankara | 5461231212 | K | 2005-03-08 | A- | Değiştir Sil |
| 69328144704 | Doğacan | Özdemir | Kırıkkale | 5422252904 | E | 2001-10-05 | 0+ | Değiştir Sil |

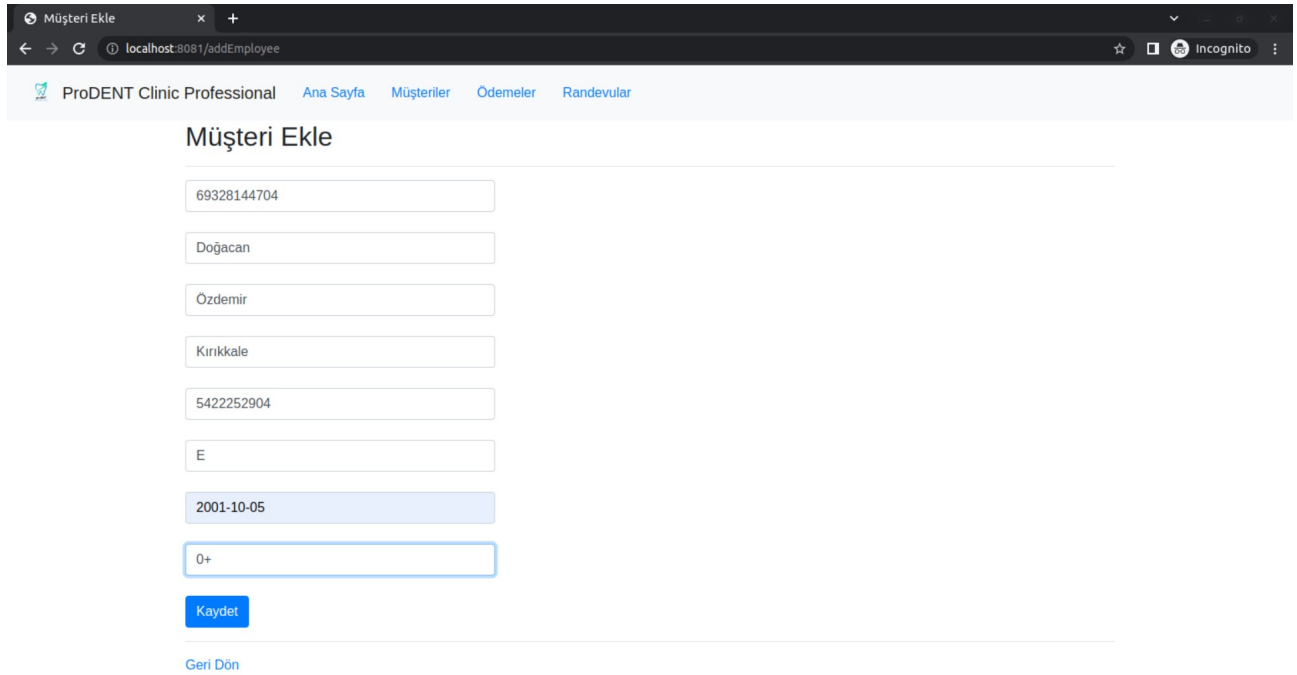
Showing 1 to 3 of 3 entries

[Müşteri Ekle](#)

Previous 1 Next

Şekil-22 – Müşteriler

3.3.4. Müşteri Ekle



Müşteri Ekle

69328144704

Doğacan

Özdemir

Kırıkkale

5422252904

E

2001-10-05

0+

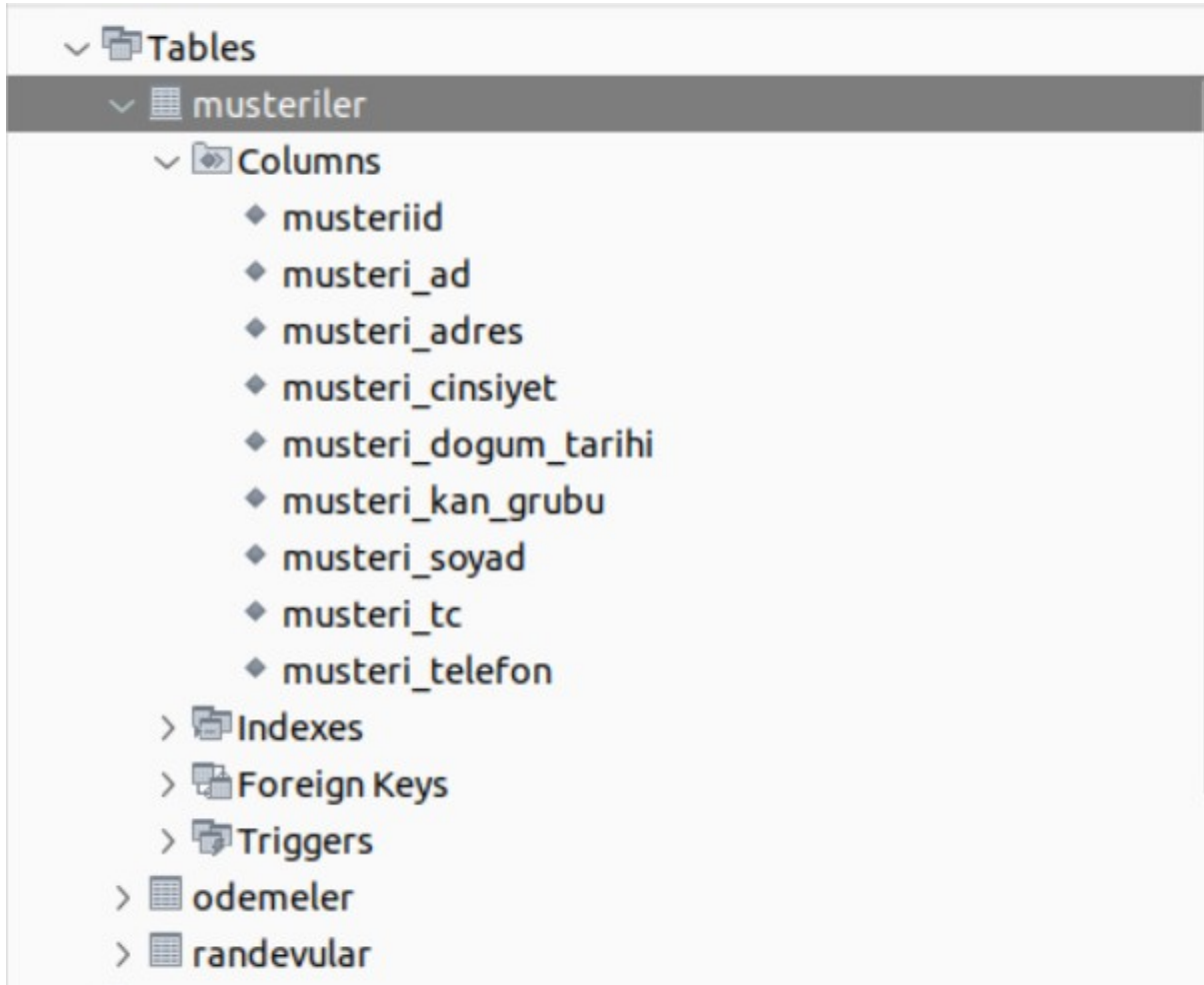
[Kaydet](#)

[Geri Dön](#)

Şekil-23 – Müşteri Ekle

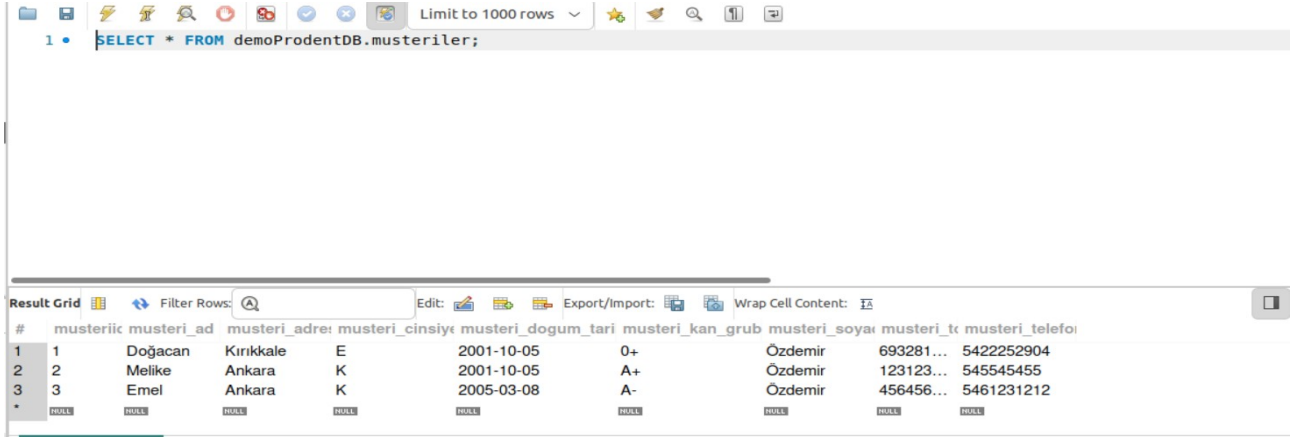
3.4. MySQL

3.4.1. Müşteri Tablosu



Şekil-24 – MySQL Müşteri Tablosu

3.4.2. Veri Girişi Yapılmış Tablo



Limit to 1000 rows

1 • `SELECT * FROM demoProdentDB.musteriler;`

Result Grid

| # | musteri_id | musteri_ad | musteri_adre: | musteri_cinsiy: | musteri_dogum_tari | musteri_kan_grub | musteri_soyadı | musteri_tc | musteri_telefonu |
|---|------------|------------|---------------|-----------------|--------------------|------------------|----------------|------------|------------------|
| 1 | 1 | Doğacan | Kırıkkale | E | 2001-10-05 | 0+ | Özdemir | 693281... | 5422252904 |
| 2 | 2 | Melike | Ankara | K | 2001-10-05 | A+ | Özdemir | 123123... | 545545455 |
| 3 | 3 | Emel | Ankara | K | 2005-03-08 | A- | Özdemir | 456456... | 5461231212 |
| * | | | | | | | | | |

Şekil-25 – MySQL Veri Girişi Yapılmış Müşteri Tablosu

4. Sonuçlar ve Öneriler

Springboot, Thymeleaf, Bootstrap, Hibernate, MySQL, Lombok, Maven ve Spring Security teknolojilerini kullanarak localhost üzerinden çalışan bir web uygulaması geliştirdim.

Springboot'un sağladığı avantajları yeterince kullandığım bir proje oldu. Daha büyük çaplı projelerde Springboot kullanımının daha büyük avantajlar sağlayacağını düşünüyorum.

Hibernate ile MySQL sorguları yazmak yerine hazır metotlar kullanmak ve bunların arkaplanda gerçekleşiyor olması işimi çok kolaylaştırdı.

Spring Security'nin kendi içerisinde barındırdığı authenticator sistemi kusursuz bir şekilde çalışmakta. Geliştirmeye açık olması bu özelliğinin üstüne daha çok projede kullanılabilir hale getiriyor.

Proje dosyasında bulunmayan ödeme ve randevu kısmı kaynak dosyalarda mevcut şekilde bulunmaktadır. Ekleme, silme ve güncelleme işlemleri de aynı sayfalar üzerinde tanımlıdır.

Sonuç olarak geliştirdiğim proje çok büyük çaplı bir proje olmamasına rağmen öğrendiğim bilgiler gelecek için faydalı bilgilerdi. Java ile geliştirme yapan geliştiricilerin bu teknolojileri iyice kavraması ve öğrenmesi gerektiğini düşünüyorum.

5. Kaynakça

- [1] https://tr.wikipedia.org/wiki/Programlama_dili
- [2] <https://aws.amazon.com/tr/what-is/java/>
- [3] <https://www.medium.com>
- [4] [alicanakkus.github.io](https://github.com/alicanakkus)
- [5] www.opendart.com
- [6] www.yusufsezer.com.tr
- [7] tr.wikipedia.org
- [8] farukgenc.com
- [9] www.jetbrains.com