A blue text on a black background

Description automatically generated

T.C. Mudanya Üniversitesi

Veritabanı Yönetim Sistemleri Dersi

Proje Ödevi

|  |  |
| --- | --- |
| Okul No: | 234001077 |
| Ad: | Oktay |
| Soyad: | Aydoğan |

**Proje Tanıtımı**

**Proje Adı:** Kullanıcı Yönetim Sistemi

**Teknolojiler:** Spring Boot, JPA (Hibernate), PostgreSQL

**Proje Amacı:** Kullanıcı bilgilerini ve adreslerini yönetmek için bir RESTful API geliştirmek.

**İçerik**

1. **Giriş**
   * ORM nedir?
   * ORM ve JPA (Hibernate) ilişkisi
   * Projede kullanılan teknolojilerin kısa tanıtımı
2. **Projenin Kurulumu ve Çalıştırılması**
   * Gerekli yazılımlar ve kurulum adımları
   * Projenin çalıştırılması
3. **Proje Yapısı**
   * Katmanlı mimari (Controller, Service, Repository, Model)
   * DTO ve Entity arasındaki farklar
4. **Veri Tabanı Tasarımı**
   * Kullanıcı ve adres tablolarının tasarımı
   * İlişkilerin yönetimi (One-to-One ilişkisi)
5. **API Endpoints**
   * Kullanıcı kayıt (POST /register)
   * Kullanıcı arama (POST /search)
   * Tüm kullanıcıları listeleme (GET /users)
6. **Güvenlik ve Yedekleme**
   * Spring Security ile temel güvenlik önlemleri
   * Veri tabanı yedekleme stratejileri
7. **Sonuç**
   * Projenin değerlendirilmesi
   * Gelecekte yapılabilecek iyileştirmeler

**Proje Kodu ve Açıklamaları**

**1. Giriş**

**ORM Nedir?** ORM (Object Relational Mapping), nesne yönelimli programlama ile ilişkisel veri tabanları arasındaki uyumu sağlamak için kullanılan bir tekniktir. ORM, veri tabanındaki tabloları sınıflara, satırları nesnelere ve sütunları özelliklere dönüştürerek, veritabanı işlemlerini daha az kod yazarak gerçekleştirmemizi sağlar.

**JPA ve Hibernate** JPA (Java Persistence API), Java platformunda veri tabanı işlemlerini gerçekleştirmek için kullanılan bir spesifikasyondur. Hibernate ise, JPA spesifikasyonunu uygulayan popüler bir ORM aracıdır.

**Spring Boot** Spring Boot, Spring Framework'ün konfigürasyonel karmaşıklığını azaltan, hızlı ve üretim hazır uygulamalar geliştirmeyi kolaylaştıran bir araçtır.

**2. Projenin Kurulumu ve Çalıştırılması**

**Gerekli Yazılımlar**

* Java Development Kit (JDK) 8 veya üzeri
* Gradle
* PostgreSQL

**Kurulum Adımları**

1. PostgreSQL'i kurun ve **USERS-DB** adında bir veri tabanı oluşturun.
2. **application.yml** dosyasındaki veri tabanı bağlantı bilgilerini kendi sisteminize göre güncelleyin:

A screenshot of a computer

Description automatically generated

1. Proje dizininde .**/gradlew build** komutunu çalıştırarak bağımlılıkları yükleyin.
2. **./gradlew bootRun** komutunu çalıştırarak uygulamayı başlatın.

**3. Proje Yapısı**

**Katmanlı Mimari**

* **Controller**: HTTP isteklerini karşılar ve iş mantığı için Service katmanına yönlendirir.
* **Service**: İş mantığını içerir ve veri tabanı işlemleri için Repository katmanına erişir.
* **Repository**: Veri tabanı işlemlerini gerçekleştirmek için JPA'yı kullanır.
* **Model**: Veri tabanı tablolarını temsil eden sınıflardır.

**DTO ve Entity**

* **DTO (Data Transfer Object)**: Veri transferi sırasında kullanılan nesnelerdir. Genellikle istemci ve sunucu arasında veri taşımak için kullanılır.
* **Entity**: Veri tabanındaki tabloları temsil eden sınıflardır. JPA tarafından yönetilirler.

**4. Veri Tabanı Tasarımı**

**Kullanıcı Tablosu (Users)**

|  |  |  |
| --- | --- | --- |
| Alan Adı | Tip | Açıklama |
| id | Long | Birincil anahtar |
| name | String | Kullanıcı adı |
| surname | String | Kullanıcı soyadı |
| email | String | Kullanıcı email adresi |
| address | Users\_Addresses | Kullanıcı adresi ile ilişki (One-to-One) |

**Adres Tablosu (Users\_Addresses)**

|  |  |  |
| --- | --- | --- |
| Alan Adı | Tip | Açıklama |
| id | Long | Birincil anahtar |
| city | String | Şehir |
| district | String | İlçe |
| hometown | String | Memleket |
| user | Users | Kullanıcı ile ilişki (One-to-One) |

**İlişkiler**

* **Users** ve **Users\_Addresses** tabloları arasında bir One-to-One ilişkisi bulunur.

**5. API Endpoints**

**Kullanıcı Kayıt (POST /register)**

**A screen shot of a computer program

Description automatically generated**

**Kullanıcı Arama (POST /search)**

**A screen shot of a computer code

Description automatically generated**

**Tüm Kullanıcıları Listeleme (GET /all)**

**A screen shot of a computer program

Description automatically generated**

**6. Güvenlik ve Yedekleme**

**Spring Security ile Güvenlik** Spring Security kullanarak temel kimlik doğrulama ve yetkilendirme işlemleri gerçekleştirilebilir. Projeye güvenlik katmanı eklemek, API'lerin yetkisiz erişimlere karşı korunmasını sağlar.

**Veri Tabanı Yedekleme**

* PostgreSQL yedekleme araçları kullanarak düzenli yedeklemeler alınabilir.
* Yedekleme stratejileri ile veri kaybı riski minimize edilir.

**7. Sonuç**

Bu proje, ORM kullanarak Spring Boot ile nasıl bir kullanıcı yönetim sistemi geliştirilebileceğini göstermektedir. İleriye dönük olarak, kullanıcı yönetim sistemine ek özellikler eklenebilir ve güvenlik önlemleri daha da güçlendirilebilir.