1. Github dan projeyi al.
2. https://github.com/meteyildirim/school-management-backend-b326/blob/master/src/main/java/com/techproed/schoolmanagementbackendb326/payload/messages/SuccessMessages.java
3. Kendine baska bir repo olustur
4. Maven, Java 17 kurulu olsun. Intellj nin kendisinde zaten var
5. Project structure da sdk java 17 yap
6. Mvn clean build ile .jar file yap
7. Projenin ininde target dosyasinin altinda bir jar olusur. Pom xml artifact id kullanilir
8. Pgadmin databse olustur. school\_management\_my
9. mvn clean package in intellj to make jar
10. uom veya uml diagram
11. A diagram of a software system

    AI-generated content may be incorrect.Repository : Veritabanı işlemlerini (CRUD) yapar
12. Service :
    1.  Uygulamanın iş mantığını içerir.
    2.  Controller'dan gelen istekleri işler.
    3.  Gerekli hesaplama, kontrol ve dönüşümler burada yapılır.
    4.  Repository'yi çağırarak veri alır veya kaydeder.
13. Controller
    1.  HTTP isteklerini karşılar (GET, POST, PUT, DELETE).
    2.  İstekleri alır, Service katmanına yönlendirir.
    3.  Cevabı HTTP olarak geri döner.
14. Payload veya DTO: controller DTO alir verir. Dis dunyayigonderilen veri payload veya alinan
15. DAO: bu da database den gelen verinin service kismina verilis seklidir.Yani entity once DAO cevrilir sonra service verilir. Cunku entity database tam kendisi bunu dis dunyaya verip database hakkinda bilgi vermek istemeyiz.
16. Mapper: Entity yi DTO/DAO donusturme islemi yapilir. Mapper isi service isi ama cok karmasa olusturacaga icin ayrica mapper package olusturcagiz.
17. Bu bizim yapimiz, ama spring boot da calisinca kendinin ki olusturur. Core context her seyi icerir
18. Garbage collector ise yaramayan objeleri oldurur. Spring boot bunlari olsuturur ve isleri bitince garbage collector bunlari oldurur.
19. Object olsuturmak pahalidir onun icin cok olusturma, cunku hepsi daha cok heap memory yer ve ram kullanir.
20. Kural: Kullanilmayacak data yi asla database den cekme
21. 3. Ders
22. Paketler olutur config, controller, entity, exception, payload, repository, security, service,
23. Alt paketler olusturacagiz.
24. Entity de enums ve concretes
25. Concretes business ve bir tanede user olusturacagiz
26. Day enum olustur : moday to sunday
    1. Ctrl + shift +U
27. Gender enum olustur Male ve female
28. Note enum olustur
    1. Aa, ba, bb , cb, cc ,dd, ff
29. package com.sms.demo.entity.enums;  
      
    public enum Gender {  
     *MALE*,*FEMALE*}
30. package com.sms.demo.entity.enums;  
      
    public enum Day {  
      
     *MONDAY*, *TUESDAY*, *WEDNESDAY*, *THURSDAY*, *FRIDAY*, *SATURDAY*, *SUNDAY*;  
    }
31. package com.sms.demo.entity.enums;  
      
    public enum Note {  
      
     *AA*, *AB*, *BB* ,*CB*, *CC*, *DD* , *FF*}
32. role type enum olustur
33. package com.sms.demo.entity.enums;  
      
    public enum Term {  
     *FALL\_SEMESTER*,  
     *SPRING\_SEMESTER*}
34. package com.sms.demo.entity.enums;  
      
    import lombok.Getter;
35. package com.sms.demo.entity.enums;  
      
    import lombok.Getter;  
      
    @Getter  
    public enum RoleType {  
      
     *ADMIN*("Admin"),  
     *TEACHER*("Teacher"),  
     *STUDENT*("Student"),  
     *MANAGER*("Manager"),  
     *ASSISTANT\_MANAGER*("ViceDean");  
      
     public final String name;  
      
     RoleType(String name) {  
     this.name =name;  
     }  
      
    }
36. User entity olsutur, Role type entity olsutur
37. package com.sms.demo.entity.concretes.users;  
      
    import com.sms.demo.entity.enums.Gender;  
    import lombok.AllArgsConstructor;  
    import lombok.Builder;  
    import lombok.Data;  
    import lombok.NoArgsConstructor;  
      
    import javax.persistence.Entity;  
    import javax.persistence.Table;  
    import java.time.LocalDate;  
      
    @Entity // databse de tablo olarak olsuturur  
    @Data  
    @AllArgsConstructor  
    @NoArgsConstructor  
    @Builder  
    @Table(name = "t\_user") //  
    public class User {  
     private long id;  
     private String username;  
     private String ssn;  
     private String name;  
     private String surname;  
     private LocalDate birthday;  
     private String birthplace;  
     private String password;  
     private String phoneNumber;  
     private String email;  
     private Boolean buildIn;  
     private String motherName;  
     private String fatherName;  
     private int studentNumber;  
     private Boolean isActive;  
     private Boolean isAdvisor;  
     private Long advisorTeacherId;  
     private Gender gender;  
      
      
      
    }
38. package com.sms.demo.entity.concretes.users;  
      
    import com.sms.demo.entity.enums.RoleType;  
    import lombok.AllArgsConstructor;  
    import lombok.Builder;  
    import lombok.Data;  
    import lombok.NoArgsConstructor;  
      
    import javax.persistence.\*;  
      
    @Entity  
    @Table(name="roles")  
    @Data  
    @AllArgsConstructor  
    @NoArgsConstructor  
    @Builder  
    public class UserRole {  
       
     @Id  
     @GeneratedValue(strategy = GenerationType.*IDENTITY*) // id nin otomatik olarak uretlmesi ve primary key olarak belirlenmesi saglar  
     private Long id;  
      
     @Enumerated(EnumType.*STRING*)  
     private RoleType roleType;  
     private String roleName;  
    }
39. ..
40. User class git ve ekle
41. @OneToOne  
    @JsonProperty(access = JsonProperty.Access.*WRITE\_ONLY*)  
    private UserRole userRole;
42. @Id  
    @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
    private long id;
43. User update
44. package com.sms.demo.entity.concretes.users;  
      
    import com.fasterxml.jackson.annotation.JsonFormat;  
    import com.fasterxml.jackson.annotation.JsonProperty;  
    import com.sms.demo.entity.enums.Gender;  
    import lombok.AllArgsConstructor;  
    import lombok.Builder;  
    import lombok.Data;  
    import lombok.NoArgsConstructor;  
      
    import javax.persistence.\*;  
    import java.time.LocalDate;  
      
    @Entity // databse de tablo olarak olsuturur  
    @Data  
    @AllArgsConstructor  
    @NoArgsConstructor  
    @Builder  
    @Table(name = "t\_user") //  
    public class User {  
     @Id  
     @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
     private long id;  
     @Column(unique = true)  
     private String username;  
     @Column(unique = true)  
     private String ssn;  
     private String name;  
     private String surname;  
     @JsonFormat(shape=JsonFormat.Shape.*STRING*, pattern = "yyyy-MM-dd")  
     private LocalDate birthday;  
     private String birthplace;  
     @JsonProperty(access = JsonProperty.Access.*WRITE\_ONLY*)  
     private String password;  
     @Column(unique = true)  
     private String phoneNumber;  
     @Column(unique = true)  
     private String email;  
     private Boolean buildIn;  
     private String motherName;  
     private String fatherName;  
     private int studentNumber;  
     private Boolean isActive;  
     private Boolean isAdvisor;  
     private Long advisorTeacherId;  
     private Gender gender;  
     @OneToOne  
     @JsonProperty(access = JsonProperty.Access.*WRITE\_ONLY*)  
     private UserRole userRole;  
      
      
      
    }
45. Buisness paketine EducationTerm ekle
46. package com.sms.demo.entity.concretes.business;  
      
    import com.fasterxml.jackson.annotation.JsonFormat;  
    import com.sms.demo.entity.enums.Term;  
    import lombok.AllArgsConstructor;  
    import lombok.Builder;  
    import lombok.Data;  
    import lombok.NoArgsConstructor;  
      
    import javax.persistence.\*;  
    import java.time.LocalDate;  
      
    @Entity  
    @Data  
    @AllArgsConstructor  
    @NoArgsConstructor  
    @Builder  
    public class EducationTerm {  
      
     @Id  
     @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
     private Long id;  
      
     @Enumerated(EnumType.*STRING*)  
     private Term term;  
      
     @Column(name="start\_date")  
     @JsonFormat(shape = JsonFormat.Shape.*STRING*, pattern ="yyyy-MM--dd")  
     private LocalDate startDate;  
      
     @Column(name="end\_date")  
     @JsonFormat(shape = JsonFormat.Shape.*STRING*, pattern ="yyyy-MM--dd")  
     private LocalDate endDate;  
      
     @Column(name="last\_registration\_date")  
     @JsonFormat(shape = JsonFormat.Shape.*STRING*, pattern ="yyyy-MM--dd")  
     private LocalDate lastRegistrationDate;  
    }
47. Bussiness Lesson class ekle
48. package com.sms.demo.entity.concretes.business;  
      
    import lombok.AllArgsConstructor;  
    import lombok.Builder;  
    import lombok.Data;  
    import lombok.NoArgsConstructor;  
      
    import javax.persistence.Entity;  
    import javax.persistence.GeneratedValue;  
    import javax.persistence.GenerationType;  
    import javax.persistence.Id;  
      
    @Entity  
    @Data  
    @AllArgsConstructor  
    @NoArgsConstructor  
    @Builder  
    public class Lesson {  
      
      
     @Id  
     @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
     private int id;  
      
     private String lessonName;  
      
     private Integer creditScore;  
      
     private Boolean isCompulsory;  
    }
49. Lesson program olustur
50. package com.sms.demo.entity.concretes.business;  
      
    import lombok.AllArgsConstructor;  
    import lombok.Builder;  
    import lombok.Data;  
    import lombok.NoArgsConstructor;  
      
    import javax.persistence.Entity;  
    import javax.persistence.GeneratedValue;  
    import javax.persistence.GenerationType;  
    import javax.persistence.Id;  
      
    @Entity  
    @Data  
    @AllArgsConstructor  
    @NoArgsConstructor  
    @Builder  
    public class Lesson {  
      
      
     @Id  
     @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
     private int id;  
      
     private String lessonName;  
      
     private Integer creditScore;  
      
     private Boolean isCompulsory;  
    }
51. bu durumda educationterm e
52. @OneToMany(mappedBy = "educationTerm", cascade = CascadeType.*ALL*)  
    private List<LessonProgram> lessonProgram;
53. Meet class olusturuldu
54. package com.sms.demo.entity.concretes.business;  
      
    import com.fasterxml.jackson.annotation.JsonFormat;  
    import com.sms.demo.entity.concretes.users.User;  
    import lombok.AllArgsConstructor;  
    import lombok.Builder;  
    import lombok.Data;  
    import lombok.NoArgsConstructor;  
      
    import javax.persistence.\*;  
    import java.time.LocalDate;  
    import java.time.LocalTime;  
    import java.util.Date;  
    import java.util.List;  
      
    @Entity  
    @Data  
    @AllArgsConstructor  
    @NoArgsConstructor  
    @Builder  
    public class Meet {  
      
     @Id  
     @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
     private Long id;  
      
     private String description;  
      
     @JsonFormat(shape = JsonFormat.Shape.*STRING*, pattern ="yyyy-MM--dd")  
     private LocalDate date;  
      
     @JsonFormat(shape = JsonFormat.Shape.*STRING*, pattern ="HH:mm", timezone = "US")  
     private LocalTime startTime;  
       
     @JsonFormat(shape = JsonFormat.Shape.*STRING*, pattern ="HH:mm", timezone = "US")  
     private LocalTime stopTime;  
      
     @ManyToOne(cascade = CascadeType.*PERSIST*)  
     private User advisoryTeacher;  
      
     @ManyToMany  
     @JoinTable(name="meet\_student\_tab",  
     joinColumns = @JoinColumn(name="meet\_id"),  
     inverseJoinColumns = @JoinColumn(name="student\_id")  
     ) // many to many de yeni bir table olusturuldugu icin bunun ismi girilir.  
     private List<User> studentList;  
    }
55. Student Info entitisini olustur
56. package com.sms.demo.entity.concretes.business;  
      
    import com.fasterxml.jackson.annotation.JsonIgnore;  
    import com.sms.demo.entity.concretes.users.User;  
    import com.sms.demo.entity.enums.Note;  
    import lombok.AllArgsConstructor;  
    import lombok.Builder;  
    import lombok.Data;  
    import lombok.NoArgsConstructor;  
      
    import javax.persistence.\*;  
      
    @Entity  
    @Data  
    @AllArgsConstructor  
    @NoArgsConstructor  
    @Builder  
    public class StudentInfo {  
      
     @Id  
     @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
     private Long id;  
      
     private Integer absentee;  
      
     private Double midtermExam;  
      
     private Double finalExam;  
      
     private String infoMore;  
      
     private Double examAverage;  
      
     @Enumerated(EnumType.*STRING*)  
     private Note letterGrade;  
      
     @ManyToOne  
     @JsonIgnore()  
     private User teacher;  
      
     @ManyToOne  
     @JsonIgnore()  
     private User student;  
      
     @ManyToOne  
     private Lesson lesson;  
      
     @OneToOne  
     private EducationTerm educationTerm;  
      
    }
57. user ekle
58. @OneToMany(mappedBy = "")  
    private List<StudentInfo> studentInfos;

Ders 4

Bir onceki dersde tum entitiler olusturuldu.

<http://localhost:8090/login>

user name application.properties de ve gecici password ise consolda gözüküyor

Endpointleri yazacagiz. Controller kismi

1. Controller package da user ve bussiness package olustur
2. UserController class olustur
3. @RestController  
   @RequestMapping("user")  
   @RequiredArgsConstructor  
   public class UserController {  
   }
4. Service gidip business ve user package ve
5. reposotory business ve user package olustur
6. UserService class olustur
7. UserRepository interface olsutur
8. UserController git ve dependency injection yap. Yani userService private degiskenini olsutur. Bunun calismasi icin UserService gidip @Service eklemlisin. BU controllerin bunu görmesini saglar
9. Simdi DTO lari olusturacagiz. Payload package yapilcak. Mappers, messages , request, response packageleri yap.Payload DTO lar icin kullaniliyor
10. Request de abstract package yap, business yap, user yap
11. 37 dak devam
12. userRequest class olustur
13. response package de businnes, user, abstracts olusturalim
14. abstract da BaseUserResponse class olustur
15. Controller gidip , UserController class tamamlayalim
16. Payload/response/businness da ResponseMessage calss olusturalim
17. UserController tamamla.
18. Uc sekild ebilgi gonder biri path, diggeri body, diferide header