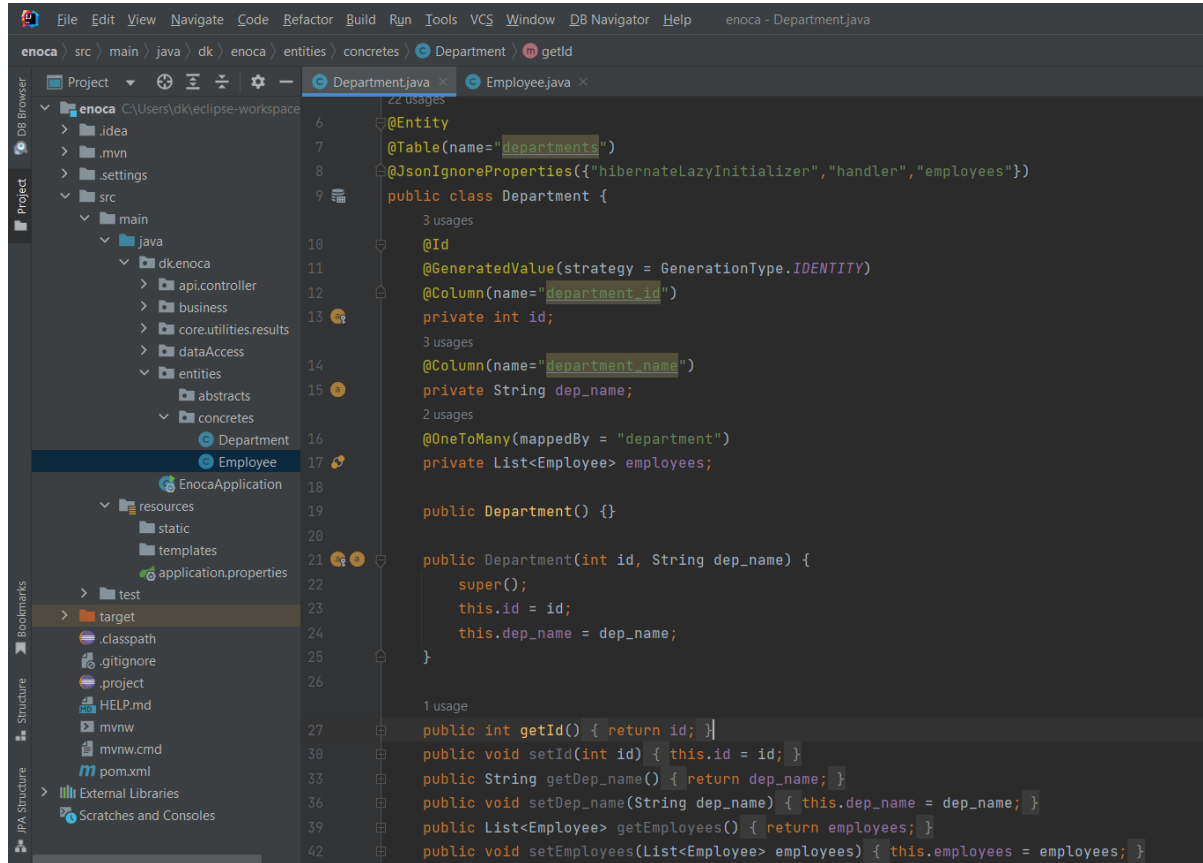


Java MVC mimarisi ile model ve controller katmanları oluşturup spring uygulaması ile ekleme,silme güncelleme ve listeleme servislerini oluşturmayı amaçladım.. Projede Java 17 versiyonunu,.API dokümantasyonu için Swagger arayüzü ve PostgreSQL veritabanını kullandım. Veritabanı bağlantısını application.properties kısmında yaptım.

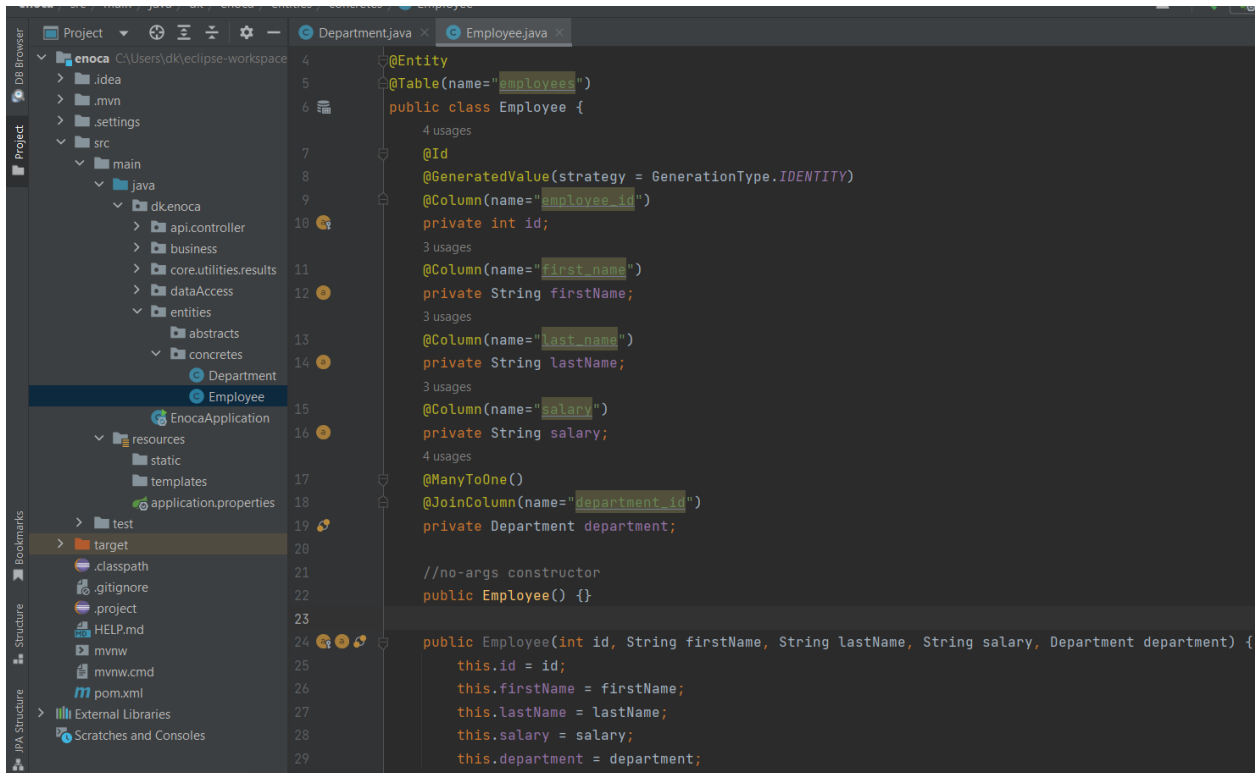
Model katmanında Department ve Employee java classlarını oluşturdum ve verilere ait alanları ekledim.



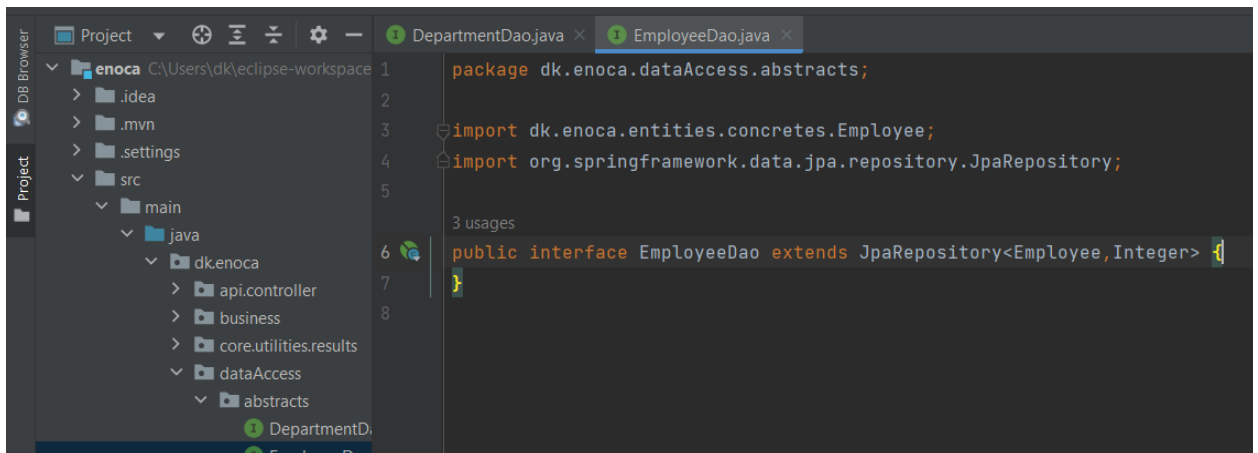
Şekil 1:Department.java

Şekil 1'de Department sınıfında id ve dep_name özelliklerini tanımladım. Ardından constructor ve getter-setter alanlarını ekledim. Department sınıfının bir entity olduğunu belirtmek için @Entity anotasyonunu kullandım. @Table anotasyonu ile veritabanında karşılık geldiği tablo adını belirttim. @Column anotasyonu ile hangi sütuna karşı geldiklerini belirttim. @Id anotasyonu id değişkeninin bir primary key olduğunu belirtiyor. @GeneratedValue anotasyonu da primary key için birbirinden farklı değerler oluşturulmasını sağlar.

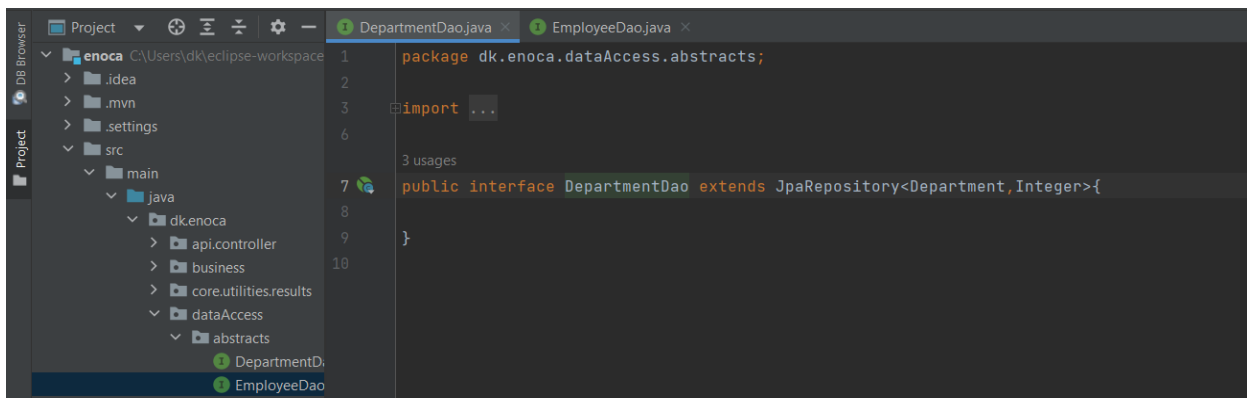
Şekil 2'de Employee modeli içinde veritabanında karşılık gelen alanları ekledim. @Entity, @Table, @Id @GeneratedValue ve @Column anotasyonlarını ilgili alanlara ekledim. Department ile Employee arasında 1-n ilişki olduğu için Department.java sınıfında List türünde tutulan employees alanına @OneToMany anotasyonu ekledim. Employee.java sınıfında Department türündeki department değişkenine @ManyToOne anotasyonunu ekledim department_id sütununa göre join işlemi yapıldığı için @JoinColumn anotasyonunu ekledim.



Şekil 2:Employee.java

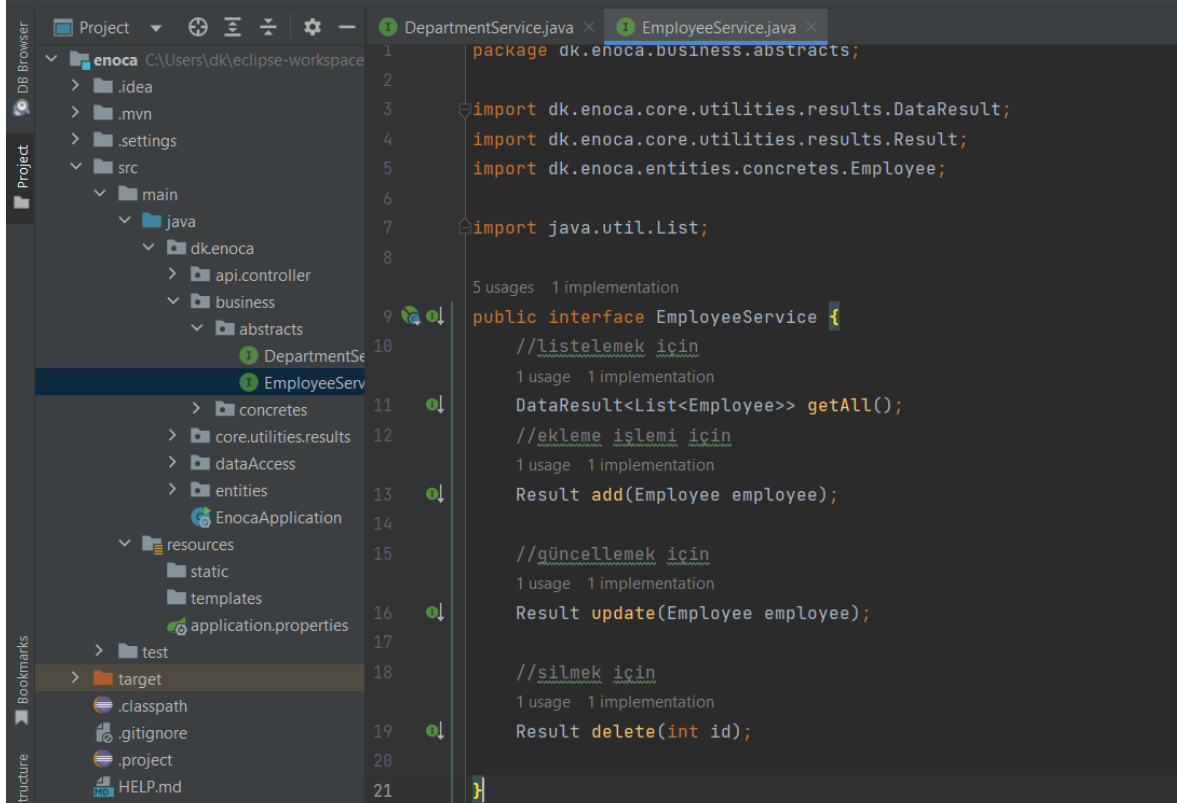


Şekil 3: EmployeeDao.java

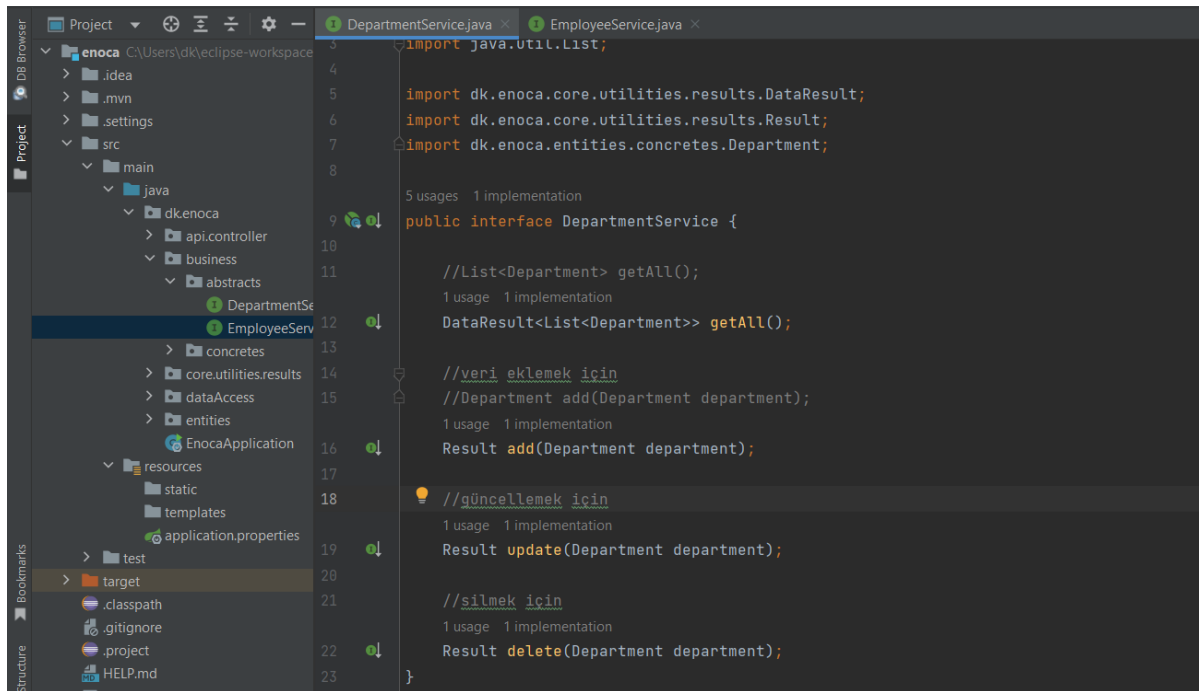


Şekil 4:DepartmentDao.java

Şekil 3 ve Şekil 4'te dataAccess paketi altında EmployeeDao ve DepartmentDao interfacelerini oluşturdum. CRUD sorgularını içinde barındıran JpaRepository'yi extend ettim ve veri tipi için @Entity anotasyonu eklediğim nesneyi ve primary key alanı integer değer olduğu için Integer değerlerini veriyorum.

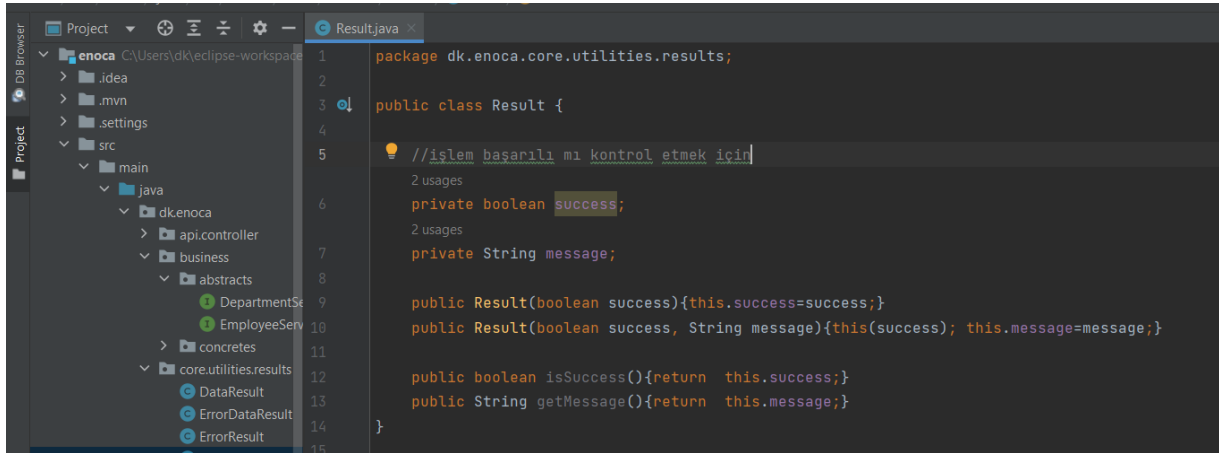


Şekil 5: EmployeeService.java



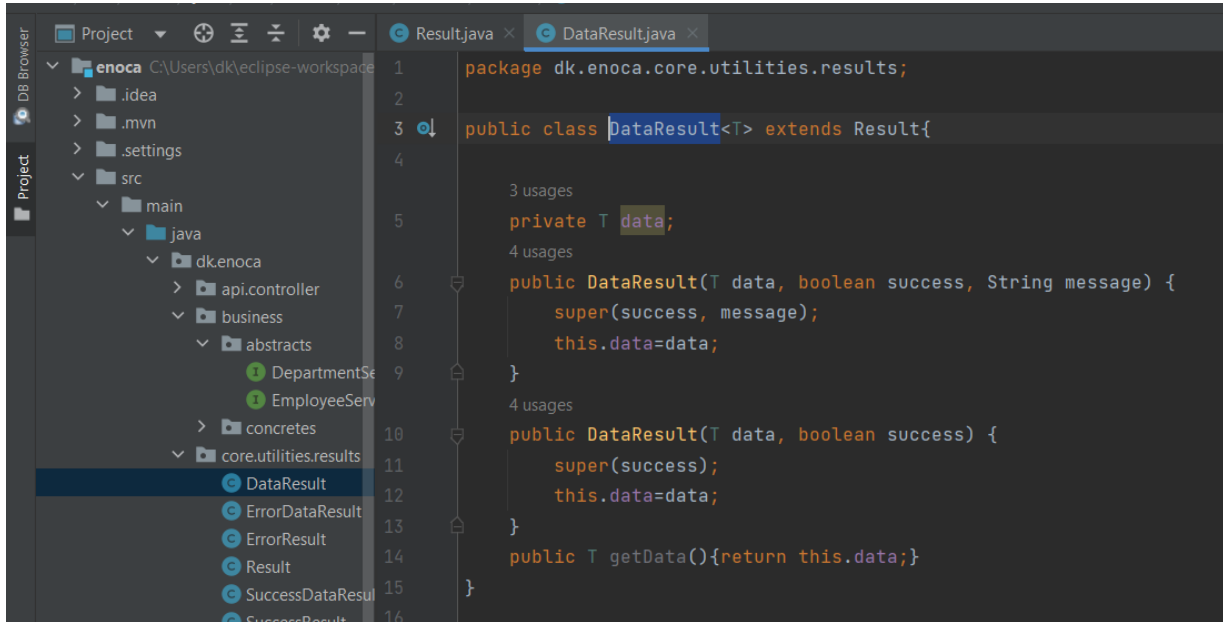
Şekil 6: DepartmentService.java

Şekil 5 ve Şekil 6'da business paketinin altında DepartmentService ve EmployeeService için interfaceler oluşturup ekleme, güncelleme, silme ve listeleme işlemleri için metod tanımları ekledim. Metod veri tiplerini gelen değerlerin başarılı mı başarısız mı olduğunu gelen isteklere göre yanıt vermek için Şekil 7'de yer alan görseldeki gibi Result sınıfı oluşturdum. Listeleme gibi içinde data olan istekler içinde Şekil 8'de yer alan Result sınıfını extend eden DataResult.java classını oluşturdum. Department Employee gibi birden fazla object veri tipi ile çalışacak olduğu için DataResult<T> şeklinde generic type tanımladım ve durum,message alanları için constructorlar tanımlayıp getter setter alanlarını ekledim.



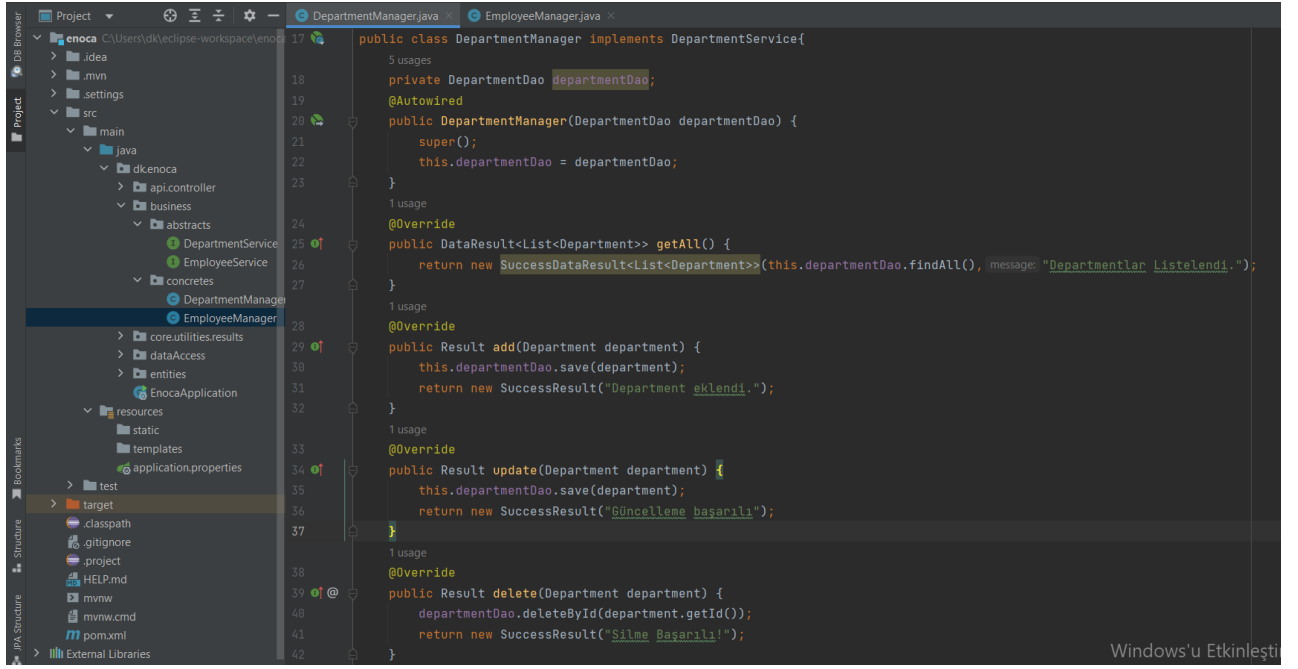
```
1 package dk.enoca.core.utilities.results;
2
3 public class Result {
4
5     //işlem başarılı mı kontrol etmek için
6     private boolean success;
7     private String message;
8
9     public Result(boolean success){this.success=success;}
10    public Result(boolean success, String message){this(success); this.message=message;}
11
12    public boolean isSuccess(){return this.success;}
13    public String getMessage(){return this.message;}
14 }
15
```

Şekil 7: Result.java



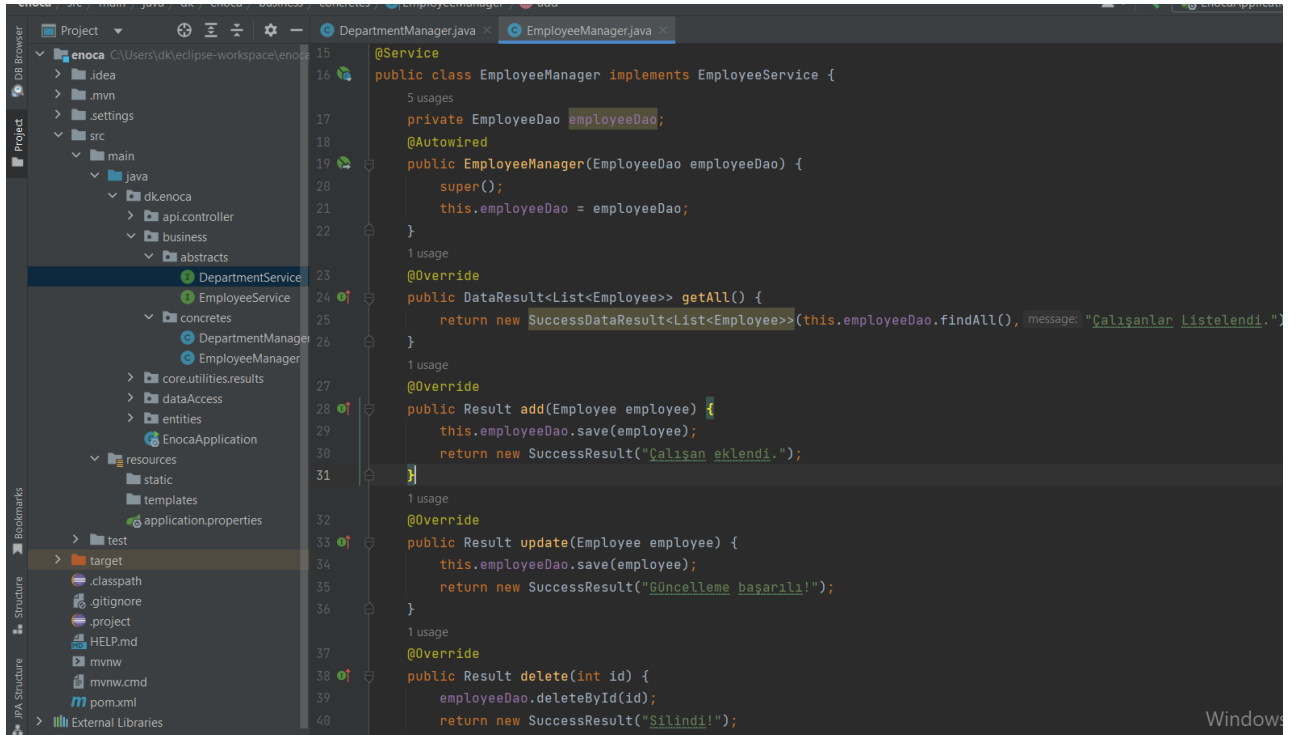
```
1 package dk.enoca.core.utilities.results;
2
3 public class DataResult<T> extends Result{
4
5     private T data;
6
7     public DataResult(T data, boolean success, String message) {
8         super(success, message);
9         this.data=data;
10    }
11
12    public DataResult(T data, boolean success) {
13        super(success);
14        this.data=data;
15    }
16
17    public T getData(){return this.data;}
18 }
19
```

Şekil 8: DataResult.java

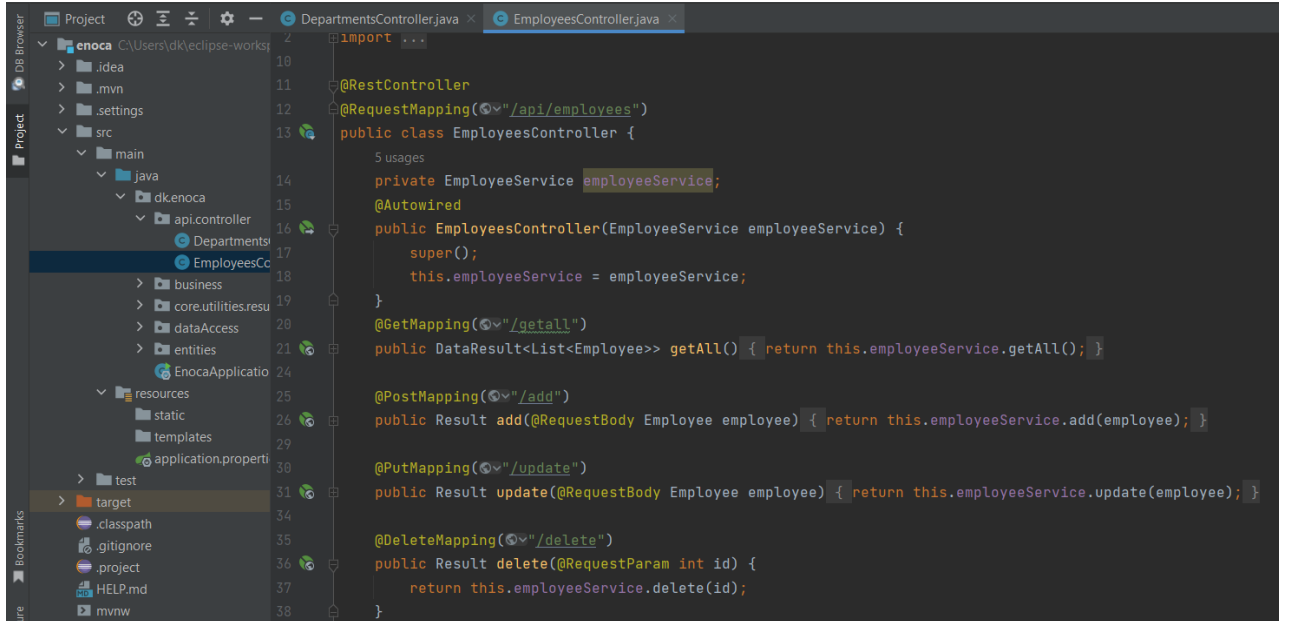


Şekil 9: DepartmentManager.java

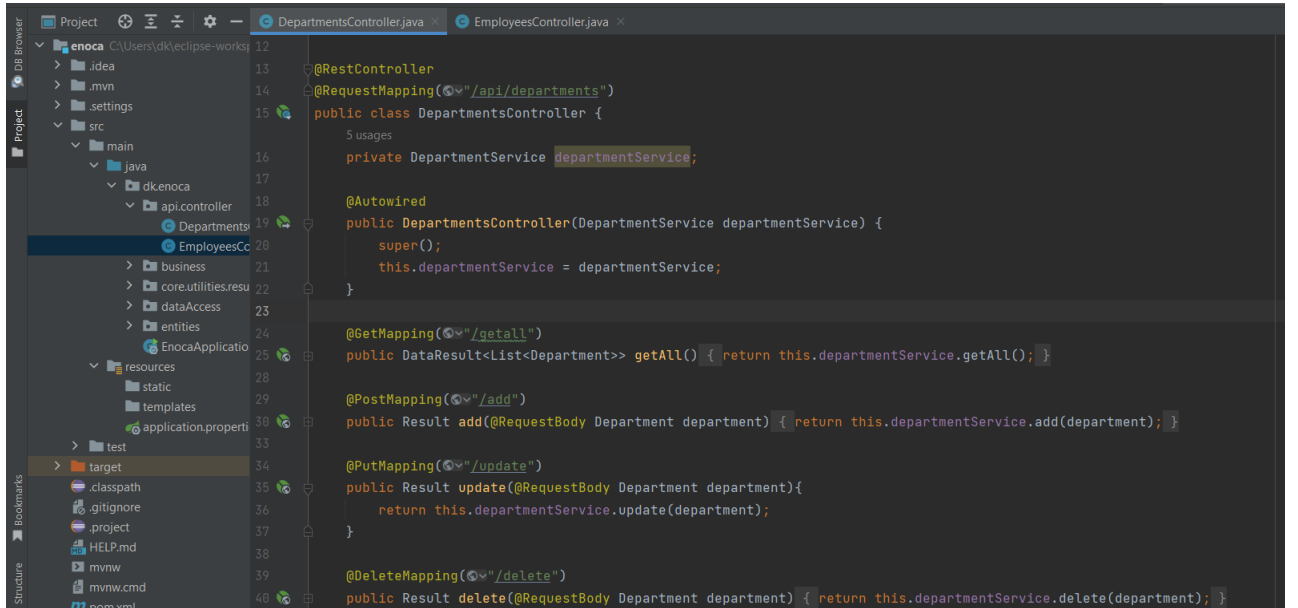
Şekil 9'da dataAccess tarafındaki daoya erişmek için private DepartmentDao departmentDao oluşturdum ve constructor injection yaptım. JpaRepositorynin verdiği değerlere göre oluşturulan DepartmentDao' karşılık gelecek bir bean oluşturuyor. Bunun içinde @Autowired anotasyonu ekledim. @Service anotasyonu ile DepartmentManager sınıfının servis görevi göreceğini belirttim. DepartmentService arayüzündeki metotları implement ettim ve listeleme ekleme güncelleme ve silme işlemleri için metotları oluşturdum. Şekil 10'da benzer işlemleri EmployeeManger.java classı için yaptım.



Şekil 10: EmployeeManager.java



Şekil 11: EmployeesController.java



Şekil 12: DepartmentsController.java

Controller dış dünyayla yani view ile sistemimizin iletişim kurduğu katmandır. Şekil 11 ve Şekil 11'de olduğu gibi EmployeesController ve DepartmentsController controller sınıfları oluşturdum.

@RestController anotasyonu ile controller sınıfı olduğunu belirttim. Gelen istekleri karşılamak için @RequestMapping anotasyonu ekledim. Verileri listelemek için istek yapıldığı için getAll() metodunun üstüne @GetMapping anotasyonunu ekledim ve @GetMapping("/getall") ile "getall" isteği gelirse bu metod çalışacak. Veriyi eklemek için @PostMapping, güncellemek için @PutMapping, silmek için de @DeleteMapping anotasyonlarını ekledim. Şekil 12'de DepartmentsController sınıfı içinde benzer işlemleri yaptım.

Swagger Screenshots and Endpoints

http://localhost:8080 - Generated server url

employees-controller

- PUT /api/employees/update
- POST /api/employees/add
- GET /api/employees/getall
- DELETE /api/employees/delete

departments-controller

- PUT /api/departments/update
- POST /api/departments/add
- GET /api/departments/getall
- DELETE /api/departments/delete

departments-controller'da /api/employees/add endpointi çağrılarak veritabanında departments tablosuna "Pazarlama Departmanı" Ekleniyor.

Curl

```
curl -X 'POST' \
  'http://localhost:8080/api/departments/add' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "id": 0,
    "dep_name": "Pazarlama Departmanı"
  }'
```

Request URL

http://localhost:8080/api/departments/add

Server response

Code Details

200

Response body

```
{
  "success": true,
  "message": "Department eklendi."
}
```

Response headers

```
connection: keep-alive
content-type: application/json
date: Sat, 30 Jul 2022 18:33:01 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

Responses

Code	Description	Links
200	OK	No links

Media type

/

Windows'u Etkinleştir
Windows'u etkinleştirmek için Ayarlar'a gidin

Tables (2)

- departments
 - Columns (2)
 - department_id
 - department_name
 - Constraints
 - Indexes
 - RLS Policies
 - Rules
 - Triggers

	department_id	department_name
	[PK] smallint	character varying (50)
1	1	İnsan Kaynakları Departmanı
2	2	Accounting Department
3	4	IT Department
4	5	Profesyonel Servisler
5	6	Pazarlama Departmanı

departments-controller'da /api/employees/update endpointi çağrılarak, id değeri 2 olan department_name alanı "Accounting Department" olan departman adı "Muhasebe Departmanı" olarak güncellendi.

Responses

Curl

```
curl -X 'PUT' \
  'http://localhost:8080/api/departments/update' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "id": 2,
    "dep_name": "Muhasebe Departmanı"
  }'
```

Request URL

http://localhost:8080/api/departments/update

Server response

Code Details

200

Response body

```
{
  "success": true,
  "message": "Güncelleme başarılı"
}
```

Response headers

```
connection: keep-alive
content-type: application/json
date: Sat, 30 Jul 2022 18:42:45 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

Responses

Code	Description	Links
200	OK	No links

Media type

/

Controls Accept header.

Example Value | Schema

Windows'u Etkinleştir
Windows'u etkinleştirmek için Aya

Tables (2)	
departments	
Columns (2)	
department_id	
department_name	
Constraints	
Indexes	
RLS Policies	
Rules	
Triggers	

Data Output	Explain	Messages	Notificat
department_id [PK] smallint	department_name character varying (50)		
1	1	İnsan Kaynakları Departmanı	
2	2	Muhasebe Departmanı	
3	4	IT Department	
4	5	Profesyonel Servisler	
5	6	Pazarlama Departmanı	

departments-controller'da /api/employees/delete endpointi çağrılarak, id değeri 5 olan Profesyonel Servisler departmanı silindi.

Tables (2)	
departments	
Columns (2)	
department_id	
department_name	
Constraints	
Indexes	
RLS Policies	
Rules	

Data Output	Explain	Messages	Notificat
department_id [PK] smallint	department_name character varying (50)		
1	1	İnsan Kaynakları Departmanı	
2	2	Muhasebe Departmanı	
3	4	IT Department	
4	6	Pazarlama Departmanı	

Responses

Curl

```
curl -X 'DELETE' \
  'http://localhost:8080/api/departments/delete' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "id": 5,
    "dep_name": "string"
  }'
```

Request URL

http://localhost:8080/api/departments/delete

Server response

Code	Details
200	<p>Response body</p> <pre>{ "success": true, "message": "Silme Başarılı!" }</pre> <p>Response headers</p> <pre>connection: keep-alive content-type: application/json date: Sat, 30 Jul 2022 18:49:08 GMT keep-alive: timeout=60 transfer-encoding: chunked</pre>

Responses

Code	Description	Links
200	OK	No links

Media type

/

Controls Accept header.

Example Value | Schema

Windows'u Etkinleştir
Windows'u etkinleştirmek için Ay

departments-controller'da /api/employees/getall endpointi çağrılarak, departments tablosundaki veriler listelendi.

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:8080/api/departments/getall' \
  -H 'accept: */*' \
```

Request URL

http://localhost:8080/api/departments/getall

Server response

Code	Details
200	<p>Response body</p> <pre>{ "success": true, "message": "Departmentlar Listelendi.", "data": [{ "id": 4, "dep_name": "IT Department" }, { "id": 1, "dep_name": "İnsan Kaynakları Departmanı" }, { "id": 6, "dep_name": "Pazarlama Departmanı" }, { "id": 2, "dep_name": "Muhasebe Departmanı" }] }</pre> <p>Response headers</p> <pre>connection: keep-alive content-type: application/json date: Sat, 30 Jul 2022 18:52:13 GMT keep-alive: timeout=60 transfer-encoding: chunked</pre>

Responses

Windows'u Etkinleştir
Windows'u etkinleştirmek için Ay