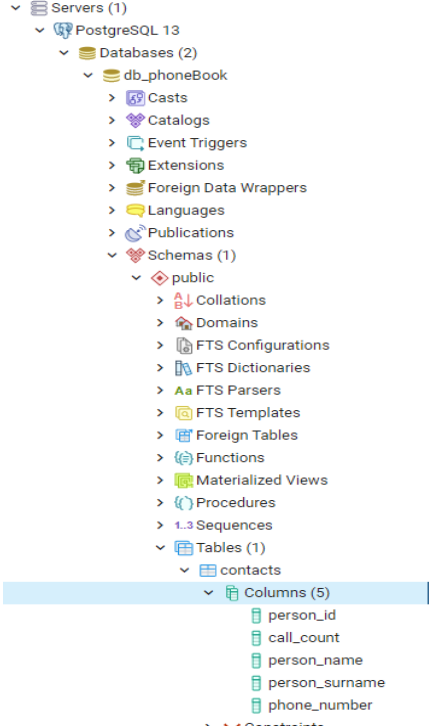


PROJE DOKÜMANI

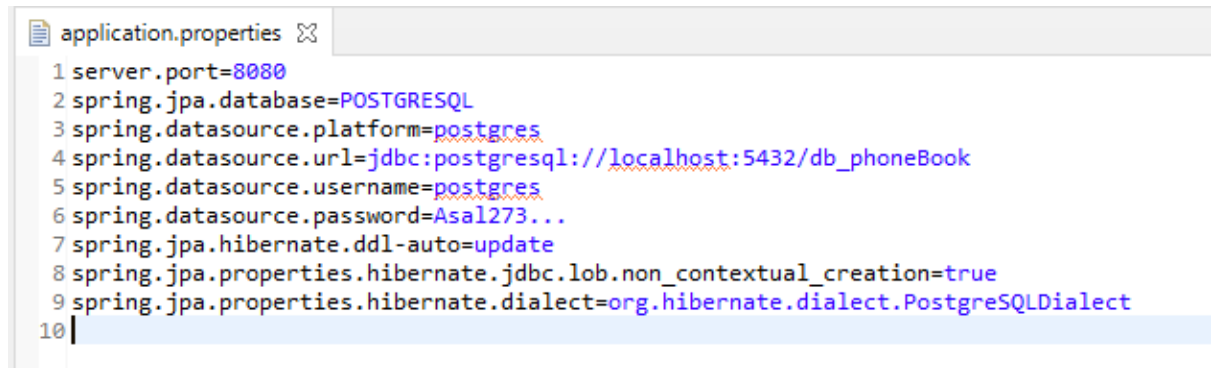
1. Veri Tabanı:



- PostgreSQL veritabanı kurulumu gerçekleştirildi.
- db_phoneBook Database'i inşaa edildi.
- contacts isimli tablo oluşturuldu.
- Rehberdeki kişilerin idsi , adı, soyadı, telefon numarası, ve her aramada artan aranma sayısı (Sık görüşülenlerin listelenmesi için kullandım.) bilgilerini tutan colonlar oluşturuldu.

2. Backend Projesi :

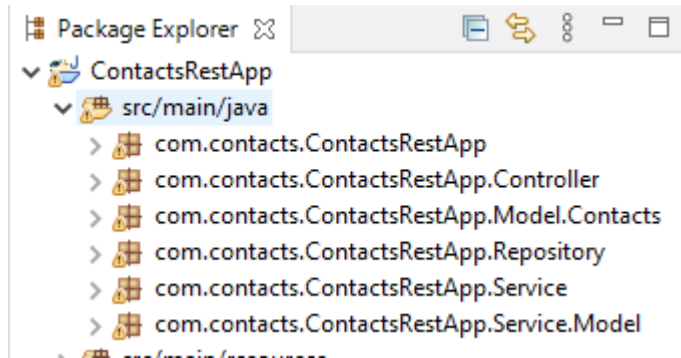
- Spring Initializr yardımı ile gradle projesini oluşturup bilgisayar üzerinde gerekli config ayarlarımı gerçekleştirdim.
- application.properties üzerinde postgresql bağlantılarımı Spring Data ile gerçekleştirdim.



- PostgreSQL bağlantısı için build.gradle içerisine tanımlama yapıldı.

```
4 dependencies {
5     implementation 'org.springframework.boot:spring-boot-starter'
6     implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
7     implementation 'org.springframework.boot:spring-boot-starter-web'
8     implementation 'javax.xml.bind:jaxb-api'
9     testImplementation 'org.springframework.boot:spring-boot-starter-test'
10    runtimeOnly 'org.postgresql:postgresql'
11 }
```

- Restfull Servisleri için Katmanlı bir yapı inşaa edildi.



- Model katmanı : Entity katmanı, Burada veritabanıyla iletişime geçerken kullanacağımız değişkenler ve get set methodları mevcut.

```

5 @Entity
7 @Table(name="contacts")
8 public class ContactsRestApp implements Serializable {
9
10     private static final long serialVersionUID = -82439648328404424L;
11
12     @Id
13     @org.springframework.data.annotation.Id
14     /*@GeneratedValue(strategy = GenerationType.IDENTITY)*/
15     @Column(name="person_id")
16     private Long person_id;
17
18     @Column(name="call_count")
19     private Long call_count;
20
21     @Column(name="person_name")
22     private String person_name;
23
24     @Column(name="person_surname")
25     private String person_surname;
26
27     @Column(name="phone_number")
28     private Long phone_number;
29
30     public ContactsRestApp() {
31         super();
32     }
33
34     public ContactsRestApp(Long person_id, Long call_count, String person_name, String person_surname, Long phone_number) {
35         super();
36         this.person_id = person_id;
37         this.call_count = call_count;
38         this.person_name = person_name;
39         this.person_surname = person_surname;
40         this.phone_number = phone_number;
41     }
42
43     public void setPersonID(Long person_id) {
44         this.person_id = person_id;
45     }
46
47     public Long getPersonID() {
48         return person_id;
49     }
50
51     public void setCallCount(Long call_count) {
52         this.call_count = call_count;
53     }
54
55 }

```

- Repository katmanı : Burada crudRespository sayesinde veritabanında çekeceğimiz alanları filtreleyebildiğimiz queryler mevcut.

```

1 package com.contacts.ContactsRestApp.Repository;
2
3 import java.util.List;
4
5 @Repository
6 public interface ContactsRestAppRepository extends CrudRepository<ContactsRestApp, Long> {
7
8     @Query(value = "SELECT * FROM contacts ",nativeQuery=true)
9     public List<ContactsRestApp> getAllContactsList();
10
11     @Query(value = "SELECT MAX(person_id) FROM contacts ",nativeQuery=true)
12     public Long findMaxId();
13
14     @Query(value = "SELECT * FROM contacts WHERE person_id = :person_id",nativeQuery=true)
15     public ContactsRestApp findWithPersonId(@Param("person_id") Long person_id);
16
17     @Query(value = "SELECT * FROM contacts WHERE person_name like :person_name% or cast(phone_number as text) like :person_name%", nativeQuery = true)
18     public List<ContactsRestApp> searchContactsList(@Param("person_name") String person_name );
19
20 }

```

- Sevice Model katmanı :

Servis katmanı için model yapısı oluşturuldu.

```
1 package com.contacts.ContactsRestApp.Service.Model;
2
3 import java.io.Serializable;
4
5 public class ContactsRestAppContext implements Serializable {
6
7     private static final long serialVersionUID = 3906169278470348749L;
8
9     private Long person_id;
10    private Long call_count;
11    private String person_name;
12    private String person_surname;
13    private Long phone_number;
14
15    public Long getPersonID() {
16        return person_id;
17    }
18    public void setPersonID(Long person_id) {
19        this.person_id = person_id;
20    }
21    public Long getCallCount() {
22        return call_count;
23    }
24    public void setCallCount(Long call_count) {
25        this.call_count = call_count;
26    }
27    public String getPersonName() {
28        return person_name;
29    }
30    public void setPersonName(String person_name) {
31        this.person_name = person_name;
32    }
33    public String getPersonSurname() {
34        return person_surname;
35    }
36    public void setPersonSurname(String person_surname) {
37        this.person_surname = person_surname;
38    }
39    public Long getPhoneNumber() {
40        return phone_number;
41    }
42    public void setPhoneNumber(Long phone_number) {
43        this.phone_number = phone_number;
44    }
45 }
46
47
```

- Service Katmanı için Interface

```
1 package com.contacts.ContactsRestApp.Service;
2
3 import com.contacts.ContactsRestApp.Service.Model.ContactsRestAppContext;
4
5 public interface IContactsRestAppService {
6
7     public Long save(ContactsRestAppContext contactsRestAppContext);
8     public Long delete(Long person_id);
9     public ContactsRestApp findWithPersonID(Long person_id);
10    public Long update(ContactsRestAppContext contactsRestAppContext);
11    public Long updateCallCount(ContactsRestAppContext contactsRestAppContext);
12    public List<ContactsRestApp> searchContactsList(ContactsRestAppContext contactsRestAppContext);
13 }
14
```

com.contacts.ContactsRestApp.Service

- ContactsRestAppService.java
- IContactsRestAppService.java

com.contacts.ContactsRestApp.Service.Model

- Service Katmanı: Controller ile Dao (Model ve Repository) arasında

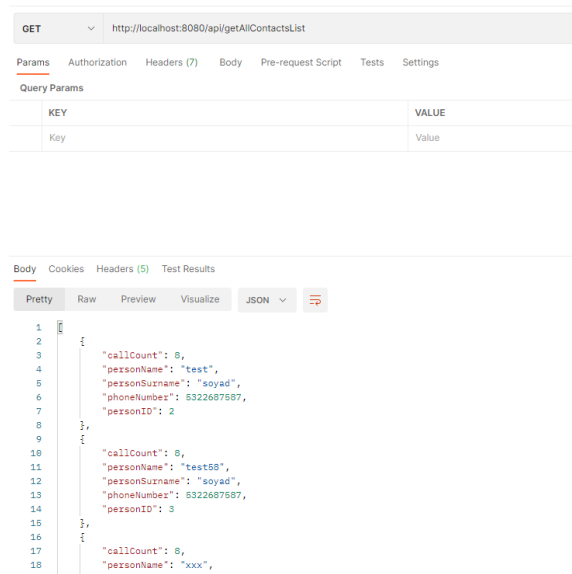
```
1 package com.contacts.ContactsRestApp.Service;
2
3 import java.util.ArrayList;
4
5 @Service
6 public class ContactsRestAppService implements IContactsRestAppService {
7
8     @Autowired
9     private ContactsRestAppRepository contactsRestAppRepository;
10
11    public List<ContactsRestApp> getAllContactsList() {
12        return contactsRestAppRepository.getAllContactsList();
13    }
14
15    public ContactsRestApp findWithPersonID(Long person_id) {
16        return contactsRestAppRepository.findWithPersonID(person_id);
17    }
18
19    public List<ContactsRestApp> searchContactsList(ContactsRestAppContext contactsRestAppContext) {
20        return contactsRestAppRepository.searchContactsList(contactsRestAppContext.getPersonName());
21    }
22
23    @Transactional
24    public Long save(ContactsRestAppContext contactsRestAppContext) {
25        Long maxId;
26        try {
27            maxId = contactsRestAppRepository.findMaxId() + 1;
28        } catch (Exception e) {
29            maxId = (Long) 1;
30        }
31
32        ContactsRestApp contactsRestApp = new ContactsRestApp();
33        contactsRestApp.setPersonID(maxId);
34        contactsRestApp.setCallCount(contactsRestAppContext.getCallCount());
35        contactsRestApp.setPersonName(contactsRestAppContext.getPersonName());
36        contactsRestApp.setPersonSurname(contactsRestAppContext.getPersonSurname());
37        contactsRestApp.setPhoneNumber(contactsRestAppContext.getPhoneNumber());
38        contactsRestApp = contactsRestAppRepository.save(contactsRestApp);
39
40        if (books.getPersonID() > 0) {
41            throw new RuntimeException("CUSTOM ERROR FOR ROLLBACK!");
42        }
43
44        return contactsRestApp.getPersonID();
45    }
46
47
```

- Controller Katmanı : Rest servislerin tanımlandığı katman.

```
1 @RestController
2 @RequestMapping("/api")
3 public class ContactsRestAppController {
4
5     @Autowired
6     private ContactsRestAppService contactsRestAppService;
7
8     @RequestMapping(value = "/getAllContactsList", method = RequestMethod.GET)
9     public List<ContactsRestApp> getAllContactsList() {
10        return contactsRestAppService.getAllContactsList();
11    }
12
13    @RequestMapping(value = "/contacts", method = RequestMethod.POST)
14    public Long save(@RequestBody ContactsRestAppContext contactsRestAppContext) {
15        return contactsRestAppService.save(contactsRestAppContext);
16    }
17
18    @RequestMapping(value = "/contacts/{person_id}", method = RequestMethod.DELETE)
19    public Long delete(@PathVariable("person_id") Long person_id) {
20        return contactsRestAppService.delete(person_id);
21    }
22
23    @RequestMapping(value = "/deleteAllContacts", method = RequestMethod.DELETE)
24    public String deleteAllContacts() {
25        return contactsRestAppService.deleteAllContacts();
26    }
27
28    @RequestMapping(value = "/contactsUpdate", method = RequestMethod.PUT)
29    public Long update(@RequestBody ContactsRestAppContext contactsRestAppContext) {
30        return contactsRestAppService.update(contactsRestAppContext);
31    }
32
33    @RequestMapping(value = "/contactsUpdateCallCount", method = RequestMethod.PUT)
34    public Long updateCallCount(@RequestBody ContactsRestAppContext contactsRestAppContext) {
35        return contactsRestAppService.updateCallCount(contactsRestAppContext);
36    }
37
38    @RequestMapping(value = "/contactsSearch", method = RequestMethod.POST)
39    public List<ContactsRestApp> searchContactsList(@RequestBody ContactsRestAppContext contactsRestAppContext) {
40        return contactsRestAppService.searchContactsList(contactsRestAppContext);
41    }
42 }
43
```

3.Postman ile servislerin kontrolü:

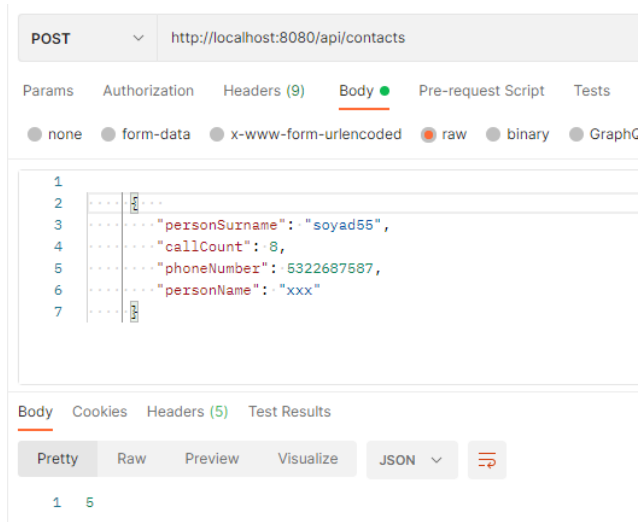
- Rehberi listeleyen servis



Postman interface showing a GET request to `http://localhost:8080/api/getAllContactsList`. The request is successful, and the response body is displayed in JSON format. The response contains an array of three contact objects.

```
1 {
2   "callCount": 8,
3   "personName": "test",
4   "personSurname": "soyad",
5   "phoneNumber": 5322687587,
6   "personID": 2
7 },
8 {
9   "callCount": 8,
10  "personName": "test58",
11  "personSurname": "soyad",
12  "phoneNumber": 5322687587,
13  "personID": 3
14 },
15 {
16   "callCount": 8,
17   "personName": "xxx",
```

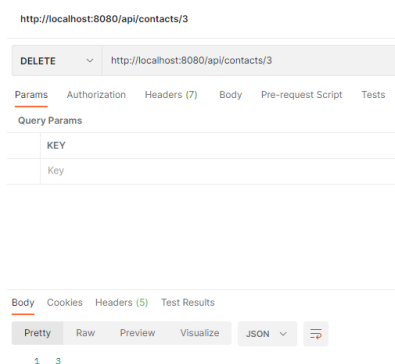
- Yeni kişi ekleme servisi:



Postman interface showing a POST request to `http://localhost:8080/api/contacts`. The request is successful, and the response body is displayed in JSON format. The response contains a single contact object.

```
1 {
2   "personSurname": "soyad55",
3   "callCount": 8,
4   "phoneNumber": 5322687587,
5   "personName": "xxx"
6 }
```

- Silme servisi:



Postman interface showing a DELETE request to `http://localhost:8080/api/contacts/3`. The request is successful, and the response body is displayed in JSON format. The response contains a single contact object.

```
1 {
2   "personSurname": "soyad55",
3   "callCount": 8,
4   "phoneNumber": 5322687587,
5   "personName": "xxx"
6 }
```

- Kişi güncelleme servisi:

http://localhost:8080/api/contactsUpdate

PUT http://localhost:8080/api/contactsUpdate

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   ... "personID": 1,
3   ... "personSurname": "soyad55",
4   ... "callCount": 8,
5   ... "phoneNumber": 5322687587,
6   ... "personName": "testzzz"
7 }
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

1 1

- Arama Servisi:

POST http://localhost:8080/api/contactsSearch

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   ... "personName": "xx"
3 }
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   ... {
3     "callCount": 0,
4     "personName": "xxx",
5     "personSurname": "soyad55",
6     "phoneNumber": 5322687587,
7     "personID": 4
8   },
9   ... {
10    "callCount": 0,
11    "personName": "xxx",
12    "personSurname": "soyad55",
13    "phoneNumber": 5322687587,
14    "personID": 5
15  }
16 }
```

- Arama sayısını güncelleme :

PUT http://localhost:8080/api/contactsUpdateCallCount

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   ... "personID": 1,
3   ... "personSurname": "soyad55",
4   ... "callCount": 9,
5   ... "phoneNumber": 5322687587,
6   ... "personName": "testzzz"
7 }
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

1 1

4.Docker kurulumu:

Docker kurulumu ardından Dockerfile ve build.gradle içerisine gerekli tanımlamalar yapıldı.

```
Dockerfile
1 FROM openjdk:8
2 VOLUME /tmp
3 ARG JAR_FILE
4 COPY ${JAR_FILE} app.jar
5 ENTRYPOINT ["java", "-jar", "/app.jar"]

docker {
    name "contactsRestApp"
    dockerfile file('src/docker/Dockerfile')
    copySpec.from(jar).rename(".*", "app.jar")
    buildArgs(['JAR_FILE': "app.jar"])
}
```

5.Swagger Kurulumu:

build.gradle ve ContactsRestAppApplication.java üzerinde config ayarları yapıldı.

```
4 dependencies {
5     implementation 'org.springframework.boot:spring-boot-starter'
6     implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
7     implementation 'org.springframework.boot:spring-boot-starter-web'
8     implementation 'javax.xml.bind:jaxb-api'
9     testImplementation 'org.springframework.boot:spring-boot-starter-test'
0     runtimeOnly 'org.postgresql:postgresql'
1     implementation 'io.springfox:springfox-swagger2:2.6.1'
2     implementation 'io.springfox:springfox-swagger-ui:2.0.2'
3 }
```

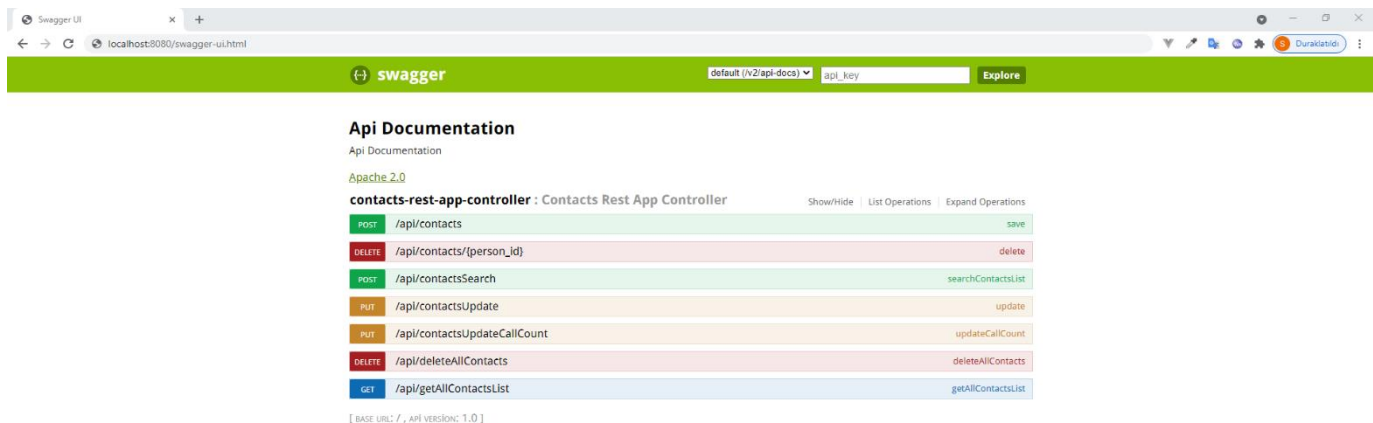
```
import springfox.documentation.builders.RequestHandlerSelectors;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

@EnableJpaRepositories(basePackages = { "com.contacts.ContactsRestApp.Repository" })
@SpringBootApplication
@EnableSwagger2
public class ContactsRestAppApplication {

    public static void main(String[] args) {
        SpringApplication.run(ContactsRestAppApplication.class, args);
    }

    @Bean
    public Docket swaggerApi() {
        return new Docket(DocumentationType.SWAGGER_2).select()
            .apis(RequestHandlerSelectors.basePackage("com.contacts.ContactsRestApp")).build();
    }
}
```

- Test edildi: <http://localhost:8080/swagger-ui.html>



6.Frontend Projesi :

Proje çalıştırılmadan önce servislerin çalışır olması gerekmektedir. Projemin önyüzünde süremin yeterli olmaması sebebiyle şuan için sadece listeleme ve silme fonksiyonları çalışmaktadır. Proje dosyası üzerinde terminal üzerinden npm start ile projeyi çalıştırabilirsiniz.

