ANKARA ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

PROJE RAPORU

Öğrenci

İsim: Ömer Faruk

Soyisim: Akçal

Numara: 19290206

Proje Özeti

Oluşturulan raporda dönem boyunca yapılan proje ve detayları ile ilgili bilgiler bulunmaktadır. Raporu yapılan projede java,javascript ve sql dilleri kullanılmış olup bu diller bir springboot işlemine tabii tutulmuştur. Bu proje gerçekleştirilirken projenin front-end kısmında javascript dili, back-end kısmında java dili kullanılmıştır. Sql dili ile de veriler tutulmuştur. Bu proje gerçekleştirilirken “visual studio code” , “intellij idea” ve “postgresql” kullanılmıştır. Proje süresince bir web arayüzünün nasıl tasarlandığını, back-end ve front-end tarafında neler yapıldığını, kullanılan programın nasıl ve hangi amaçla kullanıldığını arka planda veri işlemlerinin nasıl gerçekleştiğini tecrübe edindim. Ayrıca proje içerisinde tecrübe ettiğim bir başka konu react ve springboot kullanarak ORM kavramını tanıyabilmek oldu.

Proje Hakkında Bilgiler

ORM (Object Relational Mapping)

İlişkisel veri tabanı ile uygulama içerisinde kullandığımız modelleri/nesneleri birbirine bağlama tekniğidir.

Db objelerinin kod tarafında bir replikası bir yansıması var gibi düşünebilirsiniz. ORM bu mapleme tekniğinin adıdır.

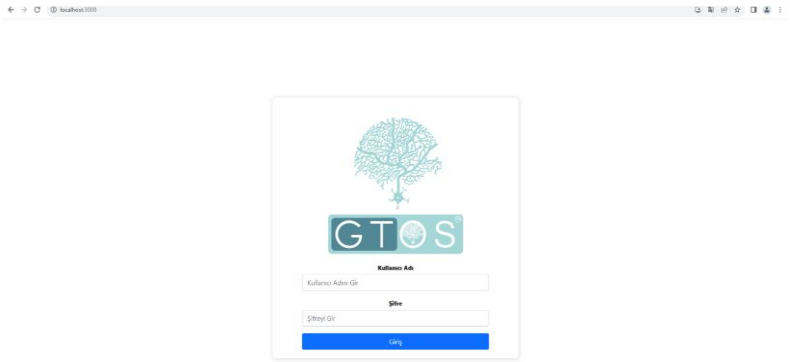
Model sınıflarımız ile database tablolarımızı map etmemizi sağlayan, bir köprü görevi gören programlama tekniğidir.

Büyük projelerde DB tasarlarken her bir tabloyu sorgular ile oluşturmaya çalışırsak günlerce sürebilir. ORM bizim için nesne ile ilişkisel veri tabanını bağlar.

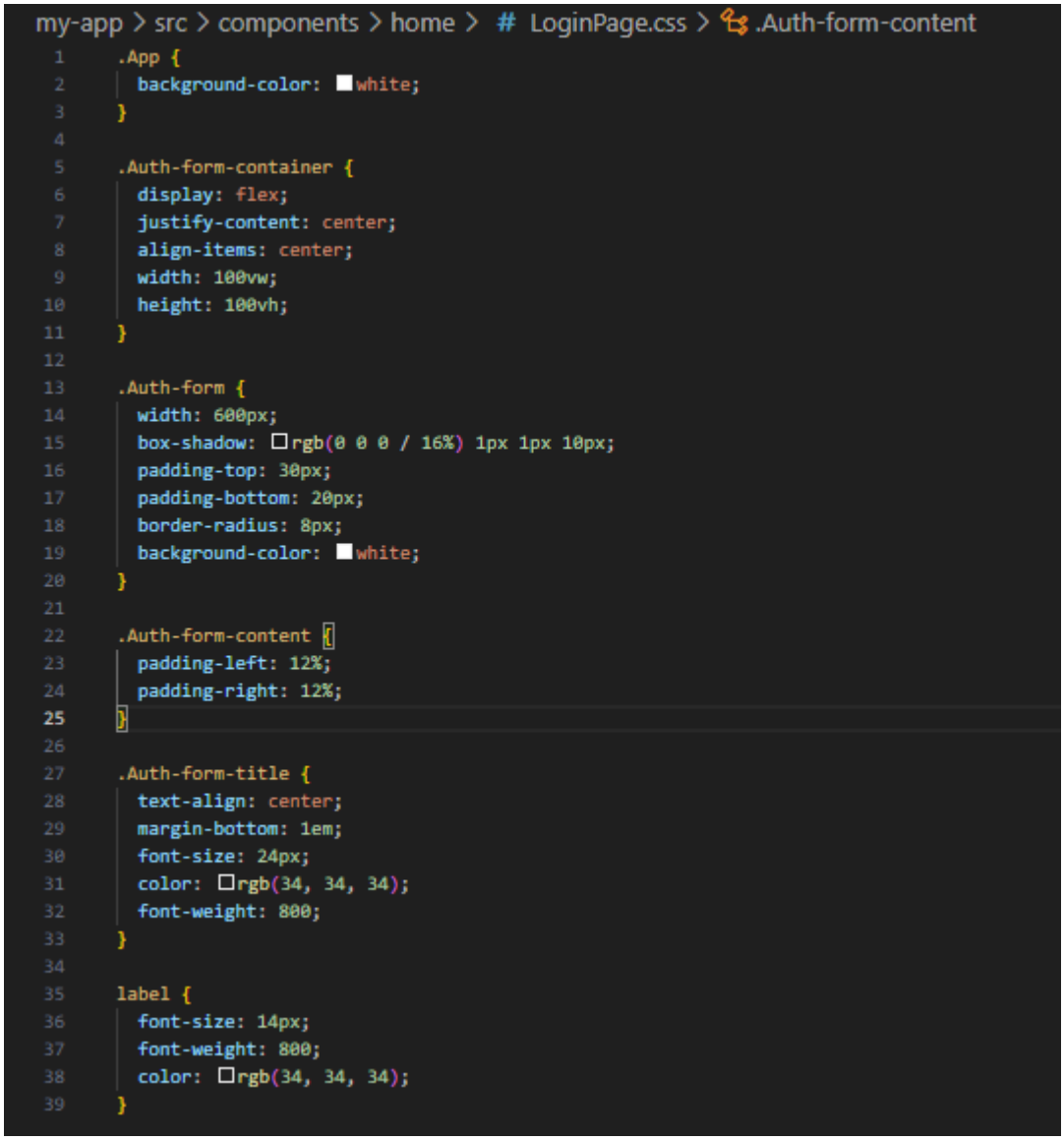
FRONT-END

LOGIN PAGE

Front-end kısmında yapılan işlemler javascript kullanılarak yazılmıştır. Arayüzün giriş sayfası şu şekildedir.

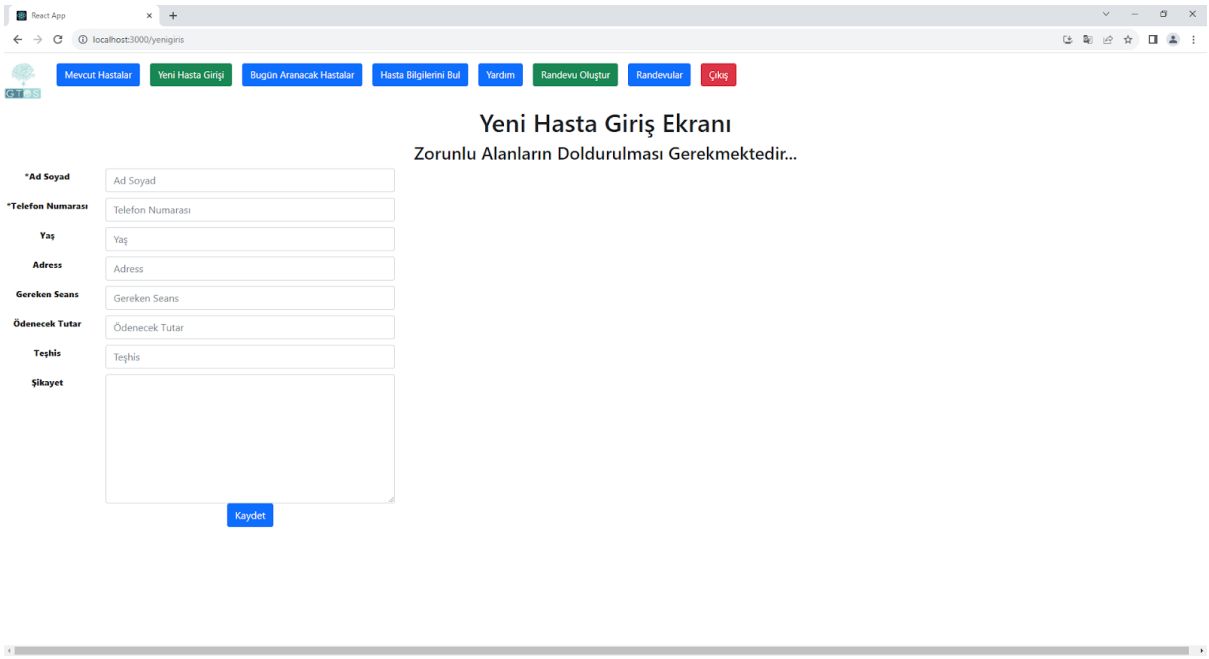



```jsx

function LoginPage() {
  const style = {
    margin: "15px 0",
  };
  const navigate = useNavigate();

  const [input, setInput] = useState('');
  const [password, setPassword] = useState('');

  const handleClick = () => {
    console.log("kullanıcı adı",input)
    console.log("şifre",password)
    if(input==="rifat")
    navigate("/odemeGiris");
    else
      navigate("/aranacakhastalar");
  }
  return (
    <div>

    <div className="Auth-form-container">
      <form className="Auth-form">
        <div className="Auth-form-content">
        <img src="/gtos-logo.png" class="img-fluid" alt="Responsive image"></img>

          <div className="form-group mt-3">

            <label>Kullanıcı Adı</label>
            <input
              id="kullanici"
              className="form-control mt-1"
              placeholder="Kullanıcı Adını Gir"
              value={input} onInput={e => setInput(e.target.value)}
            />

          </div>
          <div className="form-group mt-3">
            <label>Şifre</label>
            <input
              id="sifre"
              type="password"
              className="form-control mt-1"
              placeholder="Şifreyi Gir"
              autoComplete="on"
              value={password} onInput={e => setPassword(e.target.value)}
            />
          </div>
          <div className="d-grid gap-2 mt-3">
            <button
              type="submit"
              className="btn btn-primary"
              onClick={handleClick}
            >
              Giriş
            </button>
          </div>
        </div>
      </form>
    </div>
    </div>
  );
}

export default LoginPage;
```



Projenin giriş panelinin kodları da bu şekilde verilmiştir.

HASTA GİRİŞ EKRANI



Giriş yapıldığında önümüze yukarıdaki görsel çıkmaktadır.

SOLID

SOLID, yazılım tasarımlarını daha anlaşılır, esnek, sağlam ve sürdürülebilir hale getirmeyi amaçlayan 5 tasarım ilkesinin kısaltmasıdır . Bu uygulamaları benimsemek, kod kokularının önlenmesine de katkıda bulunabilir.

5 SOLID ilkesi şunlardır:

S - The single-responsibility principle:(1). Tek Sorumluluk İlkesi (The single-responsibility principle)

Bu ilke, “ bir sınıfın değişmesi için tek bir neden olması gerektiğini ” ifade eder , bu da her sınıfın tek bir sorumluluğu olması gerektiği anlamına gelir.

O - The open-closed principle(2). Açık-Kapalı Prensibi (The open-closed principle)

Bu ilke, “ yazılım varlıklarının (sınıflar vb.) genişletilmeye açık, ancak değiştirilmeye kapalı olması gerektiğini ” belirtir .

1.

Bu, bir sınıftaki hiçbir şeyi değiştirmeden genişletilebilir olması gerektiği anlamına gelir.

L - The Liskov substitution principle (3)-3. Liskov İkame Prensibi (The Liskov substitution principle)

Bu ilke, " türetilmiş sınıfların, temel sınıflarının yerine geçebilmesi gerektiğini " belirtir .

Başka bir deyişle, A sınıfı B sınıfının bir çocuğuysa, programın mevcut davranışını kesintiye uğratmadan B'yi A ile değiştirebilmeliyiz.

I - The interface segregation principle (4)-4. Arayüz Ayrıştırma İlkesi (The interface segregation principle)

Bu ilke Arayüzler için geçerlidir ve tek sorumluluk ilkesine benzer. “** İstemci kullanmadığı bir arabirimi uygulamaya veya istemciler kullanmadıkları

yöntemlere bağımlı olmaya asla zorlanmamalıdır.**“.

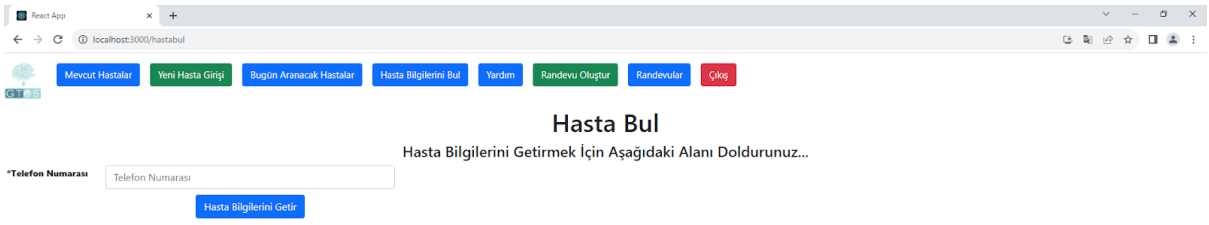
D - The dependency inversion principle (5)-Bağımlılık Tersine Çevirme İlkesi (The dependency inversion principle)

Dependency Inversion Principle (DIP), " varlıkların somut uygulamalara (sınıflara) değil, soyutlamalara (soyut sınıflar ve arayüzler) bağlı olması gerektiğini belirtir .

Ayrıca, yüksek seviyeli modül düşük seviyeli modüle değil, her ikisine de bağlı olmalıdır. soyutlamalara dayanmalıdır ".

HASTAYI BULMA EKRANI

verilen kod kullanıcıların yeni hasta bilgilerini girmelerine ve bu bilgileri bir API'ye göndermelerine olanak tanır. Bu bilgilerin doğruluğu ve eksiksizliği kontrol edilir, ve sonuca göre kullanıcıya geri bildirim sağlanır




my-app > src > components > hastabul > JS HastaBulPage.js > HastaBulPage > sendPostToApi > the
Hasta Bilgilerini Düzenle
*Ad Soyad
placeholder={this.state.people.adSoyad}
onChange={(e) => this.setState({ adSoyad: e.target.value })}
*Telefon Numarası
placeholder={this.state.people.telefonNumarasi}
onChange={(e) =>
this.setState({ telefonNumarasi: e.target.value })
Yaş
placeholder={this.state.people.yas}
onChange={(e) => this.setState({ yas: e.target.value })}
Adress
placeholder={this.state.people.adress}
onChange={(e) => this.setState({ adress: e.target.value })}

REACT

React, kullanıcı arayüzü oluşturmak ve yönetmek için kullanılan açık kaynaklı bir JavaScript kütüphanesidir. Facebook tarafından geliştirilen React, web uygulamaları ve kullanıcı arayüzleri oluşturmak için oldukça popülerdir ve yaygın olarak kullanılmaktadır.

SPRINGBOOT

Spring Boot, Java tabanlı web uygulamaları ve mikroservisler oluşturmayı kolaylaştırmak için geliştirilen bir Spring Framework projesidir. Spring Boot, uygulama geliştirme süreçlerini hızlandırmak, daha az kod yazmanızı sağlamak ve tümleşik bir çözüm sunmak amacıyla tasarlanmıştır.

BACK-END

Bu tarz projelerde Back-end kısmı projenin arkaplanıdır ve bir başka deyişle projenin beynidir. Projenin beyninin doğru çalışabilmesi için java,springboot ve PostgreSQL kullanılmıştır.

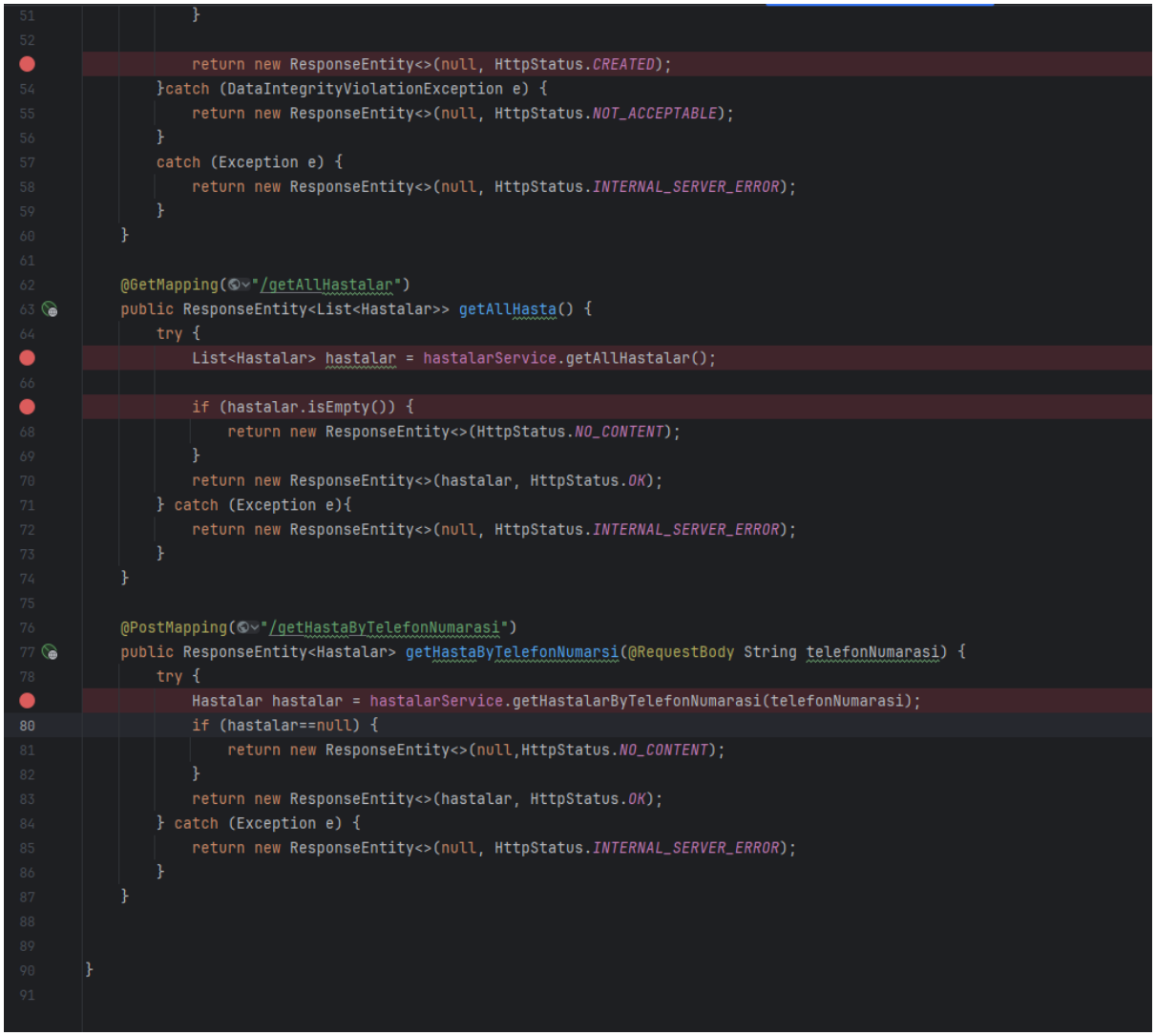
APPLICATION PROPERTIES

```
spring.datasource.url= jdbc:postgresql://localhost:5432/gtos
spring.datasource.username= omer
spring.datasource.password=4433
spring.jpa.properties.hibernate.jdbc.lob.non_contextual_creation= true
spring.jpa.properties.hibernate.dialect= org.hibernate.dialect.PostgreSQLDialect
# Hibernate ddl auto (create, create-drop, validate, update)
spring.jpa.hibernate.ddl-auto= none
```

POM.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.7.1</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.gtos</groupId>
    <artifactId>gtos</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>gtos</name>
    <description>Demo project for Spring Boot</description>
    <properties>
        <java.version>1.8</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
```

HASTA KONTROL EKRANI KODLARI

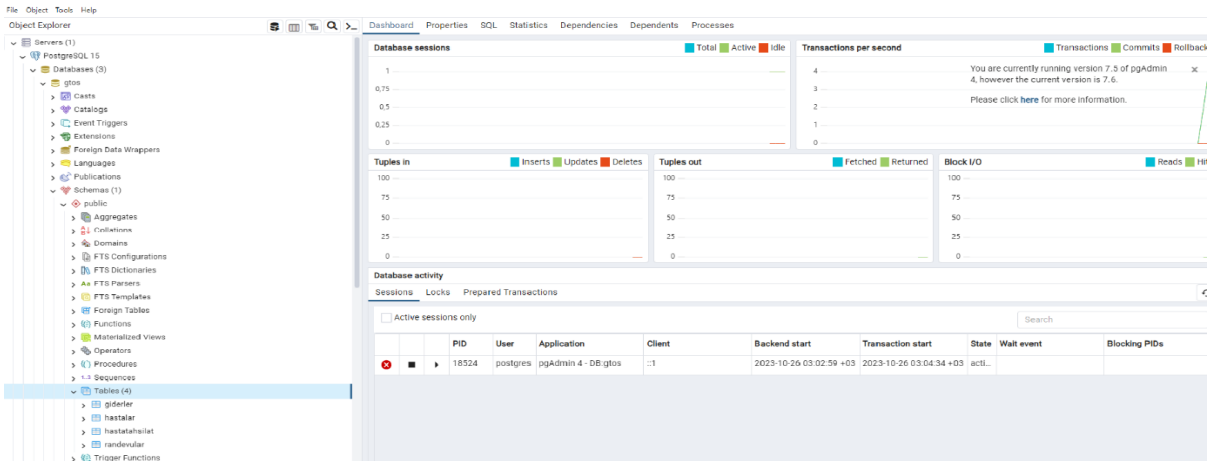


```
            }

            return new ResponseEntity<>(null, HttpStatus.CREATED);
        }catch (DataIntegrityViolationException e) {
            return new ResponseEntity<>(null, HttpStatus.NOT_ACCEPTABLE);
        }
        catch (Exception e) {
            return new ResponseEntity<>(null, HttpStatus.INTERNAL_SERVER_ERROR);
        }
    }

    @GetMapping("/getAllHastalar")
    public ResponseEntity<List<Hastalar>> getAllHasta() {
        try {
            List<Hastalar> hastalar = hastalarService.getAllHastalar();

            if (hastalar.isEmpty()) {
                return new ResponseEntity<>(HttpStatus.NO_CONTENT);
            }
            return new ResponseEntity<>(hastalar, HttpStatus.OK);
        } catch (Exception e){
            return new ResponseEntity<>(null, HttpStatus.INTERNAL_SERVER_ERROR);
        }
    }

    @PostMapping("/getHastaByTelefonNumarasi")
    public ResponseEntity<Hastalar> getHastaByTelefonNumarsi(@RequestBody String telefonNumarasi) {
        try {
            Hastalar hastalar = hastalarService.getHastalarByTelefonNumarasi(telefonNumarasi);
            if (hastalar==null) {
                return new ResponseEntity<>(null,HttpStatus.NO_CONTENT);
            }
            return new ResponseEntity<>(hastalar, HttpStatus.OK);
        } catch (Exception e) {
            return new ResponseEntity<>(null, HttpStatus.INTERNAL_SERVER_ERROR);
        }
    }


}
```

POSTGRESQL (DATABASE)

Bu tür projelerde sağlam bir veritabanına ihtiyaç duyulur. MySQL, Oracle,MSSQL gibi çeşitli veri tabanı sistemleri vardır. Bu projede ise veri tabanındaPostgreSQL kullanılmaktadır.

Tablolar aşağıdaki görsellerde verilmiştir.



GİTHUB: https://github.com/omerakcal33/Web-project