Türkkan Hastanesi Otomasyonu

Projeyi Geliştirenler

- İrem Karayel

- Gizemnur Arslan

- Umut Şahin

Proje Özeti

Bu proje, hastane süreçlerini dijitalleştirmek amacıyla geliştirilen bir otomasyon sistemidir. Sistem; kullanıcı yönetimi (hasta, doktor, admin), randevu takibi ve reçete işlemleri gibi modülleri içerir. Proje; PostgreSQL veritabanı, Node.js tabanlı bir backend ve modern JavaScript tabanlı bir frontend ile geliştirilmiştir.

Geliştirme Ortamı

- JavaScript

- PostgreSQL

- Visual Studio Code

Proje Yapısı

hastane-proje/

├── backend/ # Node.js Express API

│ └── scr

│ │ └── app.js

│ └── database.sql # Veritabanı tablolarını içeren SQL betiği

│ └── package-lock.json

│ └── package.json

│ └── server.js

│

├── frontend/ # frontend uygulaması

│ └── index.html

│ └── package-lock.json

│ └── package.json

│ └── vite.config.js

│ └── scr

│ │ └── App.css

│ │ └── App.jsx

│ │ └── index.css

│ │ └── main.jsx

│

└── README.md # Proje açıklama dosyası

├── images/ #Arayüz örnek görselleri

│ └── girisekranı.png

│ └── kayıtekranı.png

│ └── anasayfa.png

│ └── randevualma.png

│ └── randevular.png

│ └── yenirecete.png

│ └── receteler.png

Gereksinimler

Bu projeyi çalıştırabilmek için aşağıdaki yazılımların bilgisayarınızda kurulu olması gerekmektedir:

- [**Node.js**](https://nodejs.org/)

- [**PostgreSQL**](https://www.postgresql.org/)

Kurulum Adımları

1. Veritabanı Kurulumu (PostgreSQL)

1. pgAdmin programını başlatın.

2. Sol tarafta yer alan "Servers" bölümündeki sunucuya sağ tıklayın → `Create > Database` seçeneğini seçin.

3. Açılan pencerede veritabanı adı olarak `hastane_db` yazın ve `Save` butonuna tıklayın.

4. Yeni oluşturduğunuz `hastane_db` veritabanına sağ tıklayın → `Query Tool` seçeneğine tıklayın.

5. `backend/database.sql` dosyasındaki tablo oluşturma komutlarını aşağıdaki gibi sorgu penceresine yapıştırın ve çalıştırın:

`sql

```
CREATE TABLE users (
   id SERIAL PRIMARY KEY,
   name VARCHAR(100) NOT NULL,
   email VARCHAR(100) UNIQUE NOT NULL,
```

```
    password VARCHAR(100) NOT NULL,
    role VARCHAR(20) NOT NULL CHECK (role IN ('admin', 'doctor', 'patient')),
    department VARCHAR(100),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE appointments (
    id SERIAL PRIMARY KEY,
    patient_id INTEGER REFERENCES users(id),
    doctor_id INTEGER REFERENCES users(id),
    date TIMESTAMP NOT NULL,
    description TEXT,
    status VARCHAR(20) DEFAULT 'pending' CHECK (status IN ('pending', 'confirmed', 'cancelled')),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE prescriptions (
    id SERIAL PRIMARY KEY,
    patient_id INTEGER REFERENCES users(id),
    doctor_id INTEGER REFERENCES users(id),
    medications TEXT NOT NULL,
    instructions TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE medical_history (
    id SERIAL PRIMARY KEY,
    patient_id INTEGER REFERENCES users(id),
    doctor_id INTEGER REFERENCES users(id),
    diagnosis TEXT NOT NULL,
    treatment TEXT,
    allergies TEXT,
    chronic_conditions TEXT,
    last_updated TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE departments (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) UNIQUE NOT NULL,
    head_id INTEGER REFERENCES users(id),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- 1. VIEW: Randevular ile ilgili hasta ve doktor isimleri
CREATE OR REPLACE VIEW appointment_details AS
SELECT
    a.id,
    p.name AS patient_name,
    d.name AS doctor_name,
    a.date,
    a.description,
    a.status,
    a.created_at
FROM appointments a
JOIN users p ON a.patient_id = p.id
JOIN users d ON a.doctor_id = d.id;

-- 2. INDEXLER: Tarih ve durum alanlarına indeks
CREATE INDEX idx_appointments_date ON appointments(date);
CREATE INDEX idx_appointments_status ON appointments(status);

-- 3. TRIGGERLAR:

-- Fonksiyon: Yeni randevu eklendiğinde created_at güncellemesi
CREATE OR REPLACE FUNCTION set_appointment_created_at()
RETURNS TRIGGER AS $$
BEGIN
```

```
    NEW.created_at := CURRENT_TIMESTAMP;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_set_created_at
BEFORE INSERT ON appointments
FOR EACH ROW
EXECUTE FUNCTION set_appointment_created_at();

-- Fonksiyon: Randevu durumu 'cancelled' olduğunda log tutma
CREATE TABLE appointment_cancellations_log (
    id SERIAL PRIMARY KEY,
    appointment_id INTEGER,
    cancelled_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE OR REPLACE FUNCTION log_appointment_cancellation()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.status = 'cancelled' AND OLD.status <> 'cancelled' THEN
        INSERT INTO appointment_cancellations_log (appointment_id) VALUES (NEW.id);
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_log_cancellation
AFTER UPDATE ON appointments
FOR EACH ROW
WHEN (OLD.status IS DISTINCT FROM NEW.status)
EXECUTE FUNCTION log_appointment_cancellation();
```

2. Backend Kurulumu

1. Komut İstemcisi (cmd) çalıştırın.

2. Backend klasörüne geçiş yapın.

    cd D:\hastane-proje\backend

3. Node.js bağımlılıkları yükleyin.

    npm install

4. Backend sunucusu başlatın.

    npm start

3. Frontend Kurulumu

1. Yeni bir Komut İstemcisi (cmd) çalıştırın.

2. Frontend klasörüne geçiş yapın.

    cd D:\hastane-proje\frontend

3. Frontend bağımlılıkları yükleyin.

    npm install

4. Frontend uygulaması başlatın.

npm run dev

Frontend başladıktan sonra tarayıcıda otomatik olarak şu adres açılır:

http://localhost:5173

Eğer otomatik açılmazsa, tarayıcıya manuel olarak şu adres yazın:

http://localhost:5173

Arayüz Görselleri

![Giriş Ekranı](images/girisekranı.png)

![Kayıt Ekranı](images/kayıtekranı.png)

![Ana sayfa](images/anasayfa.png)

![Randevu alma Ekranı](images/randevualma.png)

![Randevular](images/randevular.png)

![Yeni Reçete Ekranı](images/yenirecete.png)

![Reçeteler](images/receteler.png)