Hastane Sistemi

Meryem PAŞAOĞLU
Bilgisayar Mühendisliği
220201123@kocaeli.edu.tr

***Abstract*—Bu doküman Prolab-II proje-3 için hazırlanmıştır. hastane yönetim sistemi projesi deteylarını içerir.**

I. GİRİŞ

Bu projede bir hastane yönetim sistemi tasarlandı ve geliştirildi. Projede, hastaların kayıt oluşturabileceği, doktorlarla randevu alabileceği, tıbbi raporları saklayabileceği ve genel olarak sağlıkla ilgili işlemleri yönetebileceği bir platform sağlanması hedeflendi.

A. Yapılan İşlemler

Projenin ana başlıkları şu şekildedir:

- Veritabanı Tasarımı:
- Hastalar tablosu: HastaID, Ad, Soyad, Doğum Tarihi, Cinsiyet, Telefon Numarası, Adres gibi bilgiler içerir.
- Doktorlar tablosu: DoktorID, Ad, Soyad, Uzmanlık Alanı, Çalıştığı Hastane gibi bilgiler içerir.
- Yönetici tablosu: YoneticiID gibi bilgiler içerir. Yöneticinin hasta, doktor ve tıbbi rapor ekleme yetkisi vardır.
- Randevular tablosu: RandevuID, Randevu Tarihi, Randevu Saati gibi bilgiler içerir.
- Tıbbi Raporlar tablosu: RaporID, Rapor Tarihi, Rapor İçeriği gibi bilgiler içerir. Ayrıca raporlar görüntü dosyaları veya JSON formatında saklanabilir.
- Nesne Yönelimli Programlama:
- Her bileşen (hasta, doktor, randevu, tıbbi rapor vb.) bir sınıf olarak modellendi ve uygun metotlar tanımlandı.
- Trigger fonksiyonları kullanılarak tablolara yapılan değişikliklerin diğer tablolara yansıtılması sağlandı.
- Arayüz Geliştirmeleri:
- ullanıcı arayüzü yenilendi ve hastaların, doktorların randevularını ve tıbbi raporlarını görüntüleyebileceği, yükleyebileceği ve indirebileceği bir arayüz sağlandı.
- Kullanıcıların tıbbi geçmişlerini, tedavi notlarını ve randevu geçmişlerini görebilecekleri dashboardlar eklendi.

II. YÖNTEM

Projede bulunan dosyalar şu şekilde özetlenebilir:

doktor.html: Bu HTML belgesi, bir doktor yönetim sistemi için bir kullanıcı arayüzü sağlar. Sayfa, doktor ekleme, listeleme, silme ve güncelleme gibi temel işlevleri gerçekleştirebilecek düğmeler ve formlar içerir.

Belgenin başlangıcı ¡!DOCTYPE html¿ ile HTML belgesi olduğunu belirtir. Ardından, ¡html lang=”en”¿ etiketi dil ve karakter kodlamasını tanımlar.

¡head¿ bölümü, belgenin başlığını ve CSS dosyasını içerir. Başlık, tarayıcı sekmesinde görüntülenecek metni sağlar. CSS dosyası, sayfanın stilini kontrol eder ve belgeye ayrı bir dosyadan bağlanmıştır.

¡body¿ bölümü, kullanıcı arayüzünü içerir. Birçok div etiketi, belgenin farklı bölümlerini tanımlar. Bu bölümler:

dashboard: Temel işlevleri içeren düğmeleri içerir. form-container: Doktor ekleme, silme ve güncelleme formlarını içerir. Silme ve güncelleme formları varsayılan olarak gizlidir. table-container: Doktorları listeleyen bir tablo içerecek, ancak şu anda içeriği yok. ¡script¿ etiketi, JavaScript kodunu içerir. Bu kod, kullanıcının etkileşimlerine yanıt vermek ve sunucuyla iletişim kurmak için kullanılır. Örneğin, doktorları listeleme, silme ve güncelleme işlevlerini içerir. Ayrıca, form gönderme olaylarını ele almak için birkaç fonksiyon içerir.

Özetle, bu HTML belgesi, kullanıcıların bir doktor yönetim sistemine erişebileceği bir arayüz sağlar. Bu arayüz, doktor ekleme, listeleme, silme ve güncelleme gibi işlevleri kolayca gerçekleştirmelerini sağlar.

doktor.js: Bu yapı, bir Express uygulaması oluşturur ve bir PostgreSQL veritabanına bağlanır. Uygulama, doktorlarla ilgili işlemleri gerçekleştirmek için HTTP isteklerini dinler.

İlk olarak, gerekli modüller express, pg (PostgreSQL istemcisi) ve fs (dosya sistemi işlemleri için) çağrılır.

Ardından, Express uygulaması oluşturulur ve belirli bir portta dinlemeye başlar.

Veritabanı bağlantı bilgileri ayarlanır ve Pool nesnesi oluşturularak bir bağlantı havuzu oluşturulur. Bu, veritabanı istemcileri arasında bağlantıları paylaşarak veritabanına daha verimli erişim sağlar.

Express uygulaması, URL kodlanmış verileri işlemek için express.urlencoded ara yazılımını kullanır ve statik dosyaları sunmak için express.static fonksiyonunu kullanır.

Ana sayfa (’/’) isteği ele alınır. Burada, bir HTML dosyası okunur ve istemciye gönderilir.

Doktor ekleme (’/doktorekle’) endpoint’i POST isteklerini işler. İstekten gelen doktor bilgileri alınır ve veritabanına eklenir. İşlem başarılıysa ”Doktor başarıyla eklendi.” mesajı gönderilir.

Doktorları listeleme (’/doktorlar’) endpoint’i GET isteklerini işler. Tüm doktorlar veritabanından alınır ve bir HTML tablosu olarak istemciye gönderilir.

Doktor silme (’/doktorsil/:doktorid’) endpoint’i DELETE isteklerini işler. Belirtilen doktor ID’sine sahip doktor veritabanından silinir. Eğer silme işlemi başarılıysa ”Doktor başarıyla silindi.” mesajı gönderilir.

Doktor güncelleme (’/update’) endpoint’i POST isteklerini işler. Belirtilen doktor ID’sine sahip doktorun bilgileri güncellenir. İşlem başarılıysa ”Doktor başarıyla güncellendi.” mesajı gönderilir.

Son olarak, sunucu belirtilen portta başlatılır ve konsola bir başlatma mesajı yazdırılır.

Bu kod, bir Express sunucusu oluşturarak PostgreSQL veritabanı üzerinde doktor bilgilerini eklemek, listelemek, güncellemek ve silmek için bir HTTP API sağlar.

doktorOlustur.js: Bu kod, PostgreSQL veritabanına rastgele doktorlar ekleyen bir Node.js betiğini içeriyor. İşlevler ve adımlar şu şekilde çalışır:

Bağımlılıkların İçe Aktarılması: İlk olarak, PostgreSQL veritabanı istemcisi (Client) ve dosya sistemi (fs) modülleri içe aktarılır. Ayrıca, dosya yolu oluşturma ve işlemleri için path modülü de içe aktarılır.

Veritabanı Bağlantısı Oluşturma: PostgreSQL veritabanına bağlanmak için gerekli olan bağlantı bilgileri (kullanıcı adı, host adresi, veritabanı adı, şifre, port numarası) belirlenir ve Client sınıfı aracılığıyla bir bağlantı oluşturulur.

Veri Oluşturma Fonksiyonları:

randommVeriiAll(): Belirtilen bir dizi içinden rastgele bir öğe seçer. dosyaYolunuAll(): Belirtilen dosyanın yolunu alır ve içeriğini okuyarak satırlara ayırır. idKontroluu(): Veritabanında benzersiz bir doktor ID’si oluşturur. Bu, veritabanında aynı ID’ye sahip bir doktor olup olmadığını kontrol ederek gerçekleştirilir. veriiTabaniinaaKaydeett(): Belirtilen doktor verilerini PostgreSQL veritabanına ekler. Doktor Oluşturma Fonksiyonu: Belirli bir sayıda (parametre olarak alınan) rastgele doktor oluşturur. Her bir doktor için ad, soyad, uzmanlık alanı ve çalıştığı hastane gibi rastgele değerler oluşturulur. Doktor ID’si benzersiz olmalıdır, bu yüzden idKontroluu() fonksiyonu kullanılır.

Ana Fonksiyon Çağrısı: randommDoktorrOlusturr() fonksiyonu, örnek olarak 10 adet rastgele doktor oluşturmak için çağrılır. Bu fonksiyon, doktorları oluştururken hataları yakalar ve son olarak veritabanı bağlantısını sonlandırır.

Bu betik, veritabanına rastgele doktorlar ekleyerek belirli bir test ortamı oluşturabilir veya örnek veri sağlayabilir.

hasta.html: Bu HTML belgesi, bir hasta yönetim sistemi için kullanıcı arayüzü sağlar. Sayfa, hasta ekleme, listeleme, silme ve güncelleme gibi temel işlevleri gerçekleştirebilecek düğmeler ve formlar içerir.

Belgenin başlangıcı ¡!DOCTYPE html¿ ile HTML belgesi olduğunu belirtir. Ardından, ¡html lang=”en”¿ etiketi dil ve karakter kodlamasını tanımlar.

¡head¿ bölümü, belgenin başlığını ve CSS dosyasını içerir. Başlık, tarayıcı sekmesinde görüntülenecek metni sağlar. CSS dosyası, sayfanın stilini kontrol eder ve belgeye ayrı bir dosyadan bağlanmıştır.

¡body¿ bölümü, kullanıcı arayüzünü içerir. Birçok div etiketi, belgenin farklı bölümlerini tanımlar. Bu bölümler:

dashboard: Temel işlevleri içeren düğmeleri içerir. form-container: Hasta ekleme, silme ve güncelleme formlarını içerir. Silme ve güncelleme formları varsayılan olarak gizlidir. table-container: Hasta bilgilerini listeleyen bir tablo içerecek, ancak şu anda içeriği yok. ¡script¿ etiketi, JavaScript kodunu içerir. Bu kod, kullanıcının etkileşimlerine yanıt vermek ve sunucuyla iletişim kurmak için kullanılır. Örneğin, hastaları listeleme, silme ve güncelleme işlevlerini içerir. Ayrıca, form gönderme olaylarını ele almak için birkaç fonksiyon içerir.

Özetle, bu HTML belgesi, kullanıcıların bir hasta yönetim sistemine erişebileceği bir arayüz sağlar. Bu arayüz, hasta ekleme, listeleme, silme ve güncelleme gibi işlevleri kolayca gerçekleştirmelerini sağlar.

hasta.js: Bu kod parçacığı, bir Express uygulaması oluşturur ve bir PostgreSQL veritabanına bağlanır. Bu uygulama, hastalarla ilgili işlemleri gerçekleştirmek için HTTP isteklerini dinler.

İlk olarak, gerekli modüller express, pg (PostgreSQL istemcisi) ve fs (dosya sistemi işlemleri için) çağrılır.

Ardından, Express uygulaması oluşturulur ve belirli bir portta dinlemeye başlar.

Veritabanı bağlantı bilgileri ayarlanır ve Pool nesnesi oluşturularak bir bağlantı havuzu oluşturulur. Bu, veritabanı istemcileri arasında bağlantıları paylaşarak veritabanına daha verimli erişim sağlar.

Express uygulaması, URL kodlanmış verileri işlemek için express.urlencoded ara yazılımını kullanır ve statik dosyaları sunmak için express.static fonksiyonunu kullanır.

Ana sayfa (’/’) isteği ele alınır. Burada, bir HTML dosyası okunur ve istemciye gönderilir.

Hasta ekleme (’/hastaekle’) endpoint’i POST isteklerini işler. İstekten gelen hasta bilgileri alınır ve veritabanına eklenir. İşlem başarılıysa ”Hasta başarıyla eklendi.” mesajı gönderilir.

Hastaları listeleme (’/hastalar’) endpoint’i GET isteklerini işler. Tüm hastalar veritabanından alınır ve bir HTML tablosu olarak istemciye gönderilir.

Hasta silme (’/hastasil/:hastaid’) endpoint’i DELETE isteklerini işler. Belirtilen hasta ID’sine sahip hasta veritabanından silinir. Eğer silme işlemi başarılıysa ”Hasta başarıyla silindi.” mesajı gönderilir.

Hasta güncelleme (’/update’) endpoint’i POST isteklerini işler. Belirtilen hasta ID’sine sahip hasta bilgileri güncellenir. İşlem başarılıysa ”Hasta başarıyla güncellendi.” mesajı gönderilir.

Son olarak, sunucu belirtilen portta başlatılır ve konsola bir başlatma mesajı yazdırılır.

Bu kod, bir Express sunucusu oluşturarak PostgreSQL veritabanı üzerinde hasta bilgilerini eklemek, listelemek, güncellemek ve silmek için bir HTTP API sağlar.

hastaOlustur.js: Bu betik, PostgreSQL veritabanına rastgele hasta kayıtları ekleyen bir Node.js betiğini içeriyor. İşlevler ve adımlar şu şekilde çalışır:

Bağımlılıkların İçe Aktarılması: İlk olarak, PostgreSQL veritabanı istemcisi (Client) ve dosya sistemi (fs) modülleri içe aktarılır. Ayrıca, dosya yolu oluşturma ve işlemleri için path modülü de içe aktarılır.

Veritabanı Bağlantısı Oluşturma: PostgreSQL veritabanına bağlanmak için gerekli olan bağlantı bilgileri (kullanıcı adı,

host adresi, veritabanı adı, şifre, port numarası) belirlenir ve Client sınıfı aracılığıyla bir bağlantı oluşturulur.

Veri Oluşturma Fonksiyonları:

randommVeriiAll(): Belirtilen bir dizi içinden rastgele bir öğe seçer. dosyaYolunuAll(): Belirtilen dosyanın yolunu alır ve içeriğini okuyarak satırlara ayırır. randommTariih(): Belirli bir tarih aralığında rastgele bir tarih oluşturur. randommTelefonNumarasii(): Rastgele bir telefon numarası oluşturur. idKontroluu(): Veritabanında benzersiz bir hasta ID'si oluşturur. Bu, veritabanında aynı ID'ye sahip bir hasta olup olmadığını kontrol ederek gerçekleştirilir. veriiTabaniinaaKaydeett(): Belirtilen hasta verilerini PostgreSQL veritabanına ekler. Hasta Oluşturma Fonksiyonu: Belirli bir sayıda (parametre olarak alınan) rastgele hasta oluşturur. Her bir hasta için ad, soyad, doğum tarihi, cinsiyet, telefon numarası ve adres gibi rastgele değerler oluşturulur. Hasta ID'si benzersiz olmalıdır, bu yüzden idKontroluu() fonksiyonu kullanılır.

Ana Fonksiyon Çağrısı: randommHastaaOlusturr() fonksiyonu, örnek olarak 10 adet rastgele hasta oluşturmak için çağrılır. Bu fonksiyon, hastaları oluştururken hataları yakalar ve son olarak veritabanı bağlantısını sonlandırır.

Bu betik, veritabanına rastgele hasta kayıtları ekleyerek belirli bir test ortamı oluşturabilir veya örnek veri sağlayabilir.

rapor.html: Bu HTML belgesi, tıbbi raporların yönetimini sağlayan bir kullanıcı arayüzü sunar. Sayfa, rapor ekleme, listeleme, silme ve güncelleme gibi temel işlevleri gerçekleştirebilecek düğmeler ve formlar içerir.

Belgenin başlangıcı ¡!DOCTYPE html¿ ile HTML belgesi olduğunu belirtir. Ardından, ¡html lang="en"¿ etiketi dil ve karakter kodlamasını tanımlar.

¡head¿ bölümü, belgenin başlığını ve CSS dosyasını içerir. Başlık, tarayıcı sekmesinde görüntülenecek metni sağlar. CSS dosyası, sayfanın stilini kontrol eder ve belgeye ayrı bir dosyadan bağlanmıştır.

¡body¿ bölümü, kullanıcı arayüzünü içerir. Birçok div etiketi, belgenin farklı bölümlerini tanımlar. Bu bölümler:

dashboard: Temel işlevleri içeren düğmeleri içerir. form-container: Rapor ekleme, silme ve güncelleme formlarını içerir. Silme ve güncelleme formları varsayılan olarak gizlidir. table-container: Rapor bilgilerini listeleyen bir tablo içerecek, ancak şu anda içeriği yok. ¡script¿ etiketi, JavaScript kodunu içerir. Bu kod, kullanıcının etkileşimlerine yanıt vermek ve sunucuyla iletişim kurmak için kullanılır. Örneğin, raporları listeleme, silme ve güncelleme işlevlerini içerir. Ayrıca, form gönderme olaylarını ele almak için birkaç fonksiyon içerir.

rapor.js: Modül ve Kütüphanelerin İçe Aktarılması: İlk olarak, gerekli modüller ve kütüphaneler require fonksiyonuyla içe aktarılır. Bu modüller ve kütüphaneler şunlardır:

express: Web sunucusu oluşturmak ve HTTP isteklerini işlemek için kullanılır. pg: PostgreSQL veritabanına bağlanmak ve sorguları yürütmek için kullanılır. fs: Dosya işlemleri yapmak için kullanılır. path: Dosya yollarını işlemek için kullanılır. Express Uygulamasının Oluşturulması ve Port Belirlenmesi: Bir Express uygulaması oluşturulur ve bir port numarası belirlenir.

PostgreSQL Bağlantısı Kurulması: PostgreSQL veritabanına bağlanmak için gerekli olan bağlantı bilgileri belirlenir ve bu bilgilere göre bir bağlantı havuzu oluşturulur.

Express Middleware'lerin Kullanılması: URL kodlanmış verilerin işlenmesi ve statik dosyaların sunulması için gerekli olan Express middleware'ler (express.urlencoded ve express.static) kullanılır.

Ana Sayfanın İşlenmesi: Ana sayfa isteği ('/') ele alınır. Burada, bir HTML dosyası okunur ve istemciye gönderilir.

Rapor Ekleme Endpoint'inin Oluşturulması: Rapor ekleme ('/raporekle') endpoint'i POST isteklerini işler. İstekten gelen rapor bilgileri alınır, PostgreSQL veritabanına eklenir ve sonuç istemciye iletilir.

Raporları Listeleme Endpoint'inin Oluşturulması: Raporları listeleme ('/tibbiraporlar') endpoint'i GET isteklerini işler. PostgreSQL veritabanından tüm raporlar alınır ve bir HTML tablosu şeklinde istemciye iletilir.

Rapor Silme Endpoint'inin Oluşturulması: Rapor silme ('/raporsil/:raporid') endpoint'i DELETE isteklerini işler. Belirtilen rapor ID'sine sahip rapor veritabanından silinir.

Rapor Güncelleme Endpoint'inin Oluşturulması: Rapor güncelleme ('/update') endpoint'i POST isteklerini işler. Belirtilen rapor ID'sine sahip rapor bilgileri güncellenir.

Sunucunun Başlatılması: Express sunucusu belirlenen portta başlatılır ve konsola başlatma mesajı yazdırılır.

raporOlustur.js: Bu betik, PostgreSQL veritabanına rastgele tıbbi raporlar ekleyen bir Node.js betiğini içeriyor. İşlevler ve adımlar şu şekilde çalışır:

Bağımlılıkların İçe Aktarılması: İlk olarak, PostgreSQL veritabanı istemcisi (Client) modülü içe aktarılır.

Veritabanı Bağlantısı Oluşturma: PostgreSQL veritabanına bağlanmak için gerekli olan bağlantı bilgileri (kullanıcı adı, host adresi, veritabanı adı, şifre, port numarası) belirlenir ve Client sınıfı aracılığıyla bir bağlantı oluşturulur.

Veri Oluşturma Fonksiyonları:

randommVeriiAll(tablooAdii, sutunn): Belirtilen tablodan belirtilen sütundaki rastgele bir veriyi seçer. randomm-Tariih(baslangicc, bitiiss): Belirli bir tarih aralığında rastgele bir tarih oluşturur. idKontroluu(): Veritabanında benzersiz bir rapor ID'si oluşturur. Bu, veritabanında aynı ID'ye sahip bir rapor olup olmadığını kontrol ederek gerçekleştirilir. Rapor Oluşturma Fonksiyonu: Belirli bir sayıda (parametre olarak alınan) rastgele tıbbi rapor oluşturur. Her bir rapor için hastanın ID'si, rapor tarihi, rapor içeriği (örneğin: bir URL) ve benzersiz bir rapor ID'si oluşturulur.

Ana Fonksiyon Çağrısı: randommRaporrOlusturr() fonksiyonu, örnek olarak 25 adet rastgele tıbbi rapor oluşturmak için çağrılır. Bu fonksiyon, raporları oluştururken hataları yakalar ve son olarak veritabanı bağlantısını sonlandırır.

Bu betik, belirtilen tabloya rastgele tıbbi raporlar ekleyerek bir test ortamı oluşturabilir veya örnek veri sağlayabilir.

randevu.html: Bu HTML belgesi, randevu yönetimini sağlayan bir kullanıcı arayüzü sunar. Sayfa, randevu alımı, listeleme, silme ve güncelleme gibi temel işlevleri gerçekleştirebilecek düğmeler ve formlar içerir.

Belgenin başlangıcı ¡!DOCTYPE html¿ ile HTML belgesi olduğunu belirtir. Ardından, ¡html lang=”en”¿ etiketi dil ve karakter kodlamasını tanımlar.

¡head¿ bölümü, belgenin başlığını ve CSS dosyasını içerir. Başlık, tarayıcı sekmesinde görüntülenecek metni sağlar. CSS dosyası, sayfanın stilini kontrol eder ve belgeye ayrı bir dosyadan bağlanmıştır.

¡body¿ bölümü, kullanıcı arayüzünü içerir. Birçok div etiketi, belgenin farklı bölümlerini tanımlar. Bu bölümler:

dashboard: Temel işlevleri içeren düğmeleri içerir. form-container: Randevu alımı, silme ve güncelleme formlarını içerir. Silme ve güncelleme formları varsayılan olarak gizlidir. table-container: Randevu bilgilerini listeleyen bir tablo içerecek, ancak şu anda içeriği yok. ¡script¿ etiketi, JavaScript kodunu içerir. Bu kod, kullanıcının etkileşimlerine yanıt vermek ve sunucuyla iletişim kurmak için kullanılır. Örneğin, randevuları listeleme, silme ve güncelleme işlevlerini içerir. Ayrıca, form gönderme olaylarını ele almak için birkaç fonksiyon içerir.

randevu.js: Bu kod, randevu alma ve yönetimi için bir HTTP API sağlar. İşlevselliği şu şekildedir:

Express ve PostgreSQL Bağlantısı: İlk olarak, gerekli modüller ve PostgreSQL veritabanı bağlantısı kurulur.

Express Middleware'lerin Kullanılması: express.urlencoded middleware'i form verilerini işlemek için, express.static middleware'i ise sunucuda bulunan dosyaları istemcilere sunmak için kullanılır.

Ana Sayfanın İşlenmesi: Ana sayfa isteği ('/') ele alınır. Bu endpoint, sunucuda bulunan bir HTML dosyasını okur ve istemciye gönderir.

Randevu Alma Endpoint'i: Randevu alma ('/randevual') endpoint'i POST isteklerini işler. İstemciden gelen randevu bilgileri PostgreSQL veritabanına eklenir.

Randevuları Listeleme Endpoint'i: Randevuları listeleme ('/randevular') endpoint'i GET isteklerini işler. PostgreSQL veritabanından tüm randevular alınır ve bir HTML tablosu şeklinde istemciye gönderilir.

Randevu Silme Endpoint'i: Randevu silme ('/randevusil/:randevuid') endpoint'i DELETE isteklerini işler. Belirtilen randevu ID'sine sahip randevu veritabanından silinir.

Randevu Güncelleme Endpoint'i: Randevu güncelleme ('/update') endpoint'i POST isteklerini işler. Belirtilen randevu ID'sine sahip randevunun bilgileri güncellenir.

Sunucunun Başlatılması: Express sunucusu belirlenen portta başlatılır ve konsola başlatma mesajı yazdırılır.

Bu kod, randevuları yönetmek için temel CRUD (Create, Read, Update, Delete) operasyonlarını destekler.

randevuOlustur.js: Bu betik, PostgreSQL veritabanına rastgele randevular ekleyen bir Node.js betiğini içeriyor. İşlevler ve adımlar şu şekilde çalışır:

Bağımlılıkların İçe Aktarılması: İlk olarak, PostgreSQL veritabanı istemcisi (Client) modülü içe aktarılır.

Veritabanı Bağlantısı Oluşturma: PostgreSQL veritabanına bağlanmak için gerekli olan bağlantı bilgileri (kullanıcı adı, host adresi, veritabanı adı, şifre, port numarası) belirlenir ve Client sınıfı aracılığıyla bir bağlantı oluşturulur.

Veri Oluşturma Fonksiyonları:

randommVeriiAll(tablooAdii, sutunn): Belirtilen tablodan belirtilen sütundaki rastgele bir veriyi seçer. randomm-Tariih(baslangicc, bitiiss): Belirli bir tarih aralığında rastgele bir tarih oluşturur. randommSaatt(suSaatten, buSaate): Belirli bir saat aralığında rastgele bir saat oluşturur. idKontroluu(): Veritabanında benzersiz bir randevu ID'si oluşturur. Bu, veritabanında aynı ID'ye sahip bir randevu olup olmadığını kontrol ederek gerçekleştirilir. Randevu Oluşturma Fonksiyonu: Belirli bir sayıda (parametre olarak alınan) rastgele randevu oluşturur. Her bir randevu için hastanın ID'si, doktorun ID'si, randevu tarihi, randevu saati ve benzersiz bir randevu ID'si oluşturulur.

Ana Fonksiyon Çağrısı: randommRandevuuOlusturr() fonksiyonu, örnek olarak 25 adet rastgele randevu oluşturmak için çağrılır. Bu fonksiyon, randevuları oluştururken hataları yakalar ve son olarak veritabanı bağlantısını sonlandırır.

Bu betik, belirtilen tabloya rastgele randevular ekleyerek bir test ortamı oluşturabilir veya örnek veri sağlayabilir.

ara.html: Bu HTML belgesi, hasta ve randevu bilgilerini almak için bir arayüz sağlar. Sayfa, belirli bir hasta veya doktorun randevu bilgilerini görüntülemek için düğmeler ve formlar içerir.

Başlık bölümünde, sayfanın başlığı ve viewport'un ayarları bulunur. CSS dosyası belgeye bağlanır.

Body bölümü, kullanıcı arayüzünü içerir. dashboard adlı bir bölüm, hasta ve doktorlar için farklı işlevleri gerçekleştiren düğmeler içerir. Ardından, her biri belirli bir işlevi gerçekleştiren dört farklı form bölümü bulunur: hasta-malumatForm, hastaRndevuuForm, doktorRandevuuForm ve raporrForm. Bu formlar varsayılan olarak gizlidir ve sadece ilgili düğmelere tıklandığında görünür hale gelirler.

Her form, kullanıcının giriş yapabileceği bir veya daha fazla alan ve bu alanlardaki bilgileri göndermek için bir düğme içerir. Gönderilen verilere göre, belirli bir hasta veya doktorun randevu bilgileri alınır veya hastanın raporları getirilir.

Javascript bölümü, form gösterme işlevi ve her bir form için gerekli bilgileri sunucudan almak için XMLHttpRequest kullanarak asenkron veri alışverişini yönetir. Alınan veriler, HTML içeriğine dönüştürülerek belirli bir div içine yerleştirilir ve kullanıcıya sunulur.

ara.js: Express ve PostgreSQL Bağlantısı: İlk kısımda gerekli modüller (express, pg, fs, path) import edilir ve bir PostgreSQL havuzu oluşturulur.

Express Middleware'lerin Kullanılması: Express uygulaması, URL kodlanmış verileri işlemek ve sunucuda bulunan dosyaları istemcilere sunmak için iki middleware kullanır.

Ana Sayfa İşlenmesi: Ana sayfa ('/') isteği ele alınır. Sunucuda bulunan ara.html dosyası okunur ve istemciye gönderilir.

Hasta Bilgilerini Getirme Endpoint'i: Belirli bir hasta ID'sine sahip hastanın bilgilerini getiren ('/hastabilgilerii') bir endpoint tanımlanır. Bu endpoint, hasta ID'si ile eşleşen randevu kaydını bulur ve hastanın bilgilerini çeker.

Hasta Randevularını Getirme Endpoint'i: Belirli bir hasta ID'sine sahip hastanın randevu bilgilerini getiren ('/hastaran-

devulari') bir endpoint tanımlanır. Bu endpoint, hasta ID'si ile eşleşen tüm randevuları çeker.

Doktor Randevularını Getirme Endpoint'i: Belirli bir doktor ID'sine sahip doktorun randevu bilgilerini getiren ('/doktorrandevulari') bir endpoint tanımlanır. Bu endpoint, doktor ID'si ile eşleşen tüm randevuları çeker.

Hasta Raporlarını Getirme Endpoint'i: Belirli bir hasta ID'sine sahip hastanın rapor bilgilerini getiren ('/raporlarigetir') bir endpoint tanımlanır. Bu endpoint, hasta ID'si ile eşleşen tüm raporları çeker.

Sunucunun Başlatılması: Express uygulaması belirlenen bir portta dinlemeye başlar ve başlatma mesajı konsola yazdırılır.

Bu kod, bir sağlık uygulaması için API sağlar. Hastaların ve doktorların randevu ve rapor bilgilerini sorgulayabilmelerini sağlar.

index.html: HTML Sayfa Yapısı Başlık (Head) Bölümü HTML sayfasının dilini ve karakter setini tanımlar. Tarayıcı uyumluluğu ve mobil cihazlar için görünüm ayarlarını içerir. Sayfa başlığı "Hastane Sistemi" olarak belirlenmiştir. Bir CSS dosyasına (style.css) bağlantı içerir. Gövde (Body) Bölümü Header (Üst Bilgi) Bölümü:

Sabit üst bilgi içerir. Bir navigasyon menüsü barındırır: Home, Hizmetlerimiz, Hemen Dene, Yorumlar. Home Bölümü:

Sağlıklı yaşam temalı bir giriş sunar. Kullanıcıları "Hemen Başla" butonuyla yönlendirir. Menu (Hizmetlerimiz) Bölümü:

Hizmetlerin listelendiği bir bölüm. Doktor, hasta, randevu, rapor işlemleri ve ek hizmetler (Beden Kütle İndeksi Hesaplama, Egzersiz Çalışmaları) tanıtılır. Gallery (Hemen Dene) Bölümü:

Hizmetleri tanıtan kutular içerir. Her hizmet için "Şimdi Dene" butonu ile ilgili sayfaya yönlendirme yapılır. Ekstra Hizmetler Bölümü:

Beden Kütle İndeksi Hesaplama Sistemi ve Egzersiz Çalışmaları hakkında bilgi verir. Kullanıcıyı ilgili hizmet sayfasına yönlendirir. Reviews (Yorumlar) Bölümü:

Hastane sistemine ilişkin yorumlar içerir. Her yorum kartı doktorun adı, uzmanlık alanı ve yorum metnini içerir. Footer (Alt Bilgi) Bölümü:

Navigasyon menüsünü tekrarlar. Sayfa sahibi olarak "Meryem Paşaoğlu" belirtilir ve tüm hakların saklı olduğu belirtilir. JavaScript openPort(port, page) fonksiyonu, belirli bir port numarası ve sayfa adı alarak yeni bir tarayıcı sekmesinde belirtilen sayfayı açar. Bu sayfa, hastane sistemine ait hizmetleri ve kullanıcı yorumlarını içeren ve kullanıcıları çeşitli işlemleri denemeye yönlendiren bir web sayfasıdır.

CSS kodları: CSS kodu, bir web uygulamasının görünümünü düzenler. İşlevsellikten ziyade, sayfaların nasıl görüneceğini tanımlar. İşte bazı anahtar noktalar:

Body: Sayfanın genel görünümünü tanımlar. Font ailesi, arka plan rengi ve yükseklik gibi özellikleri ayarlar. flex özellikleri sayesinde içeriğin yatay olarak sıralanmasını ve sol üste hizalanmasını sağlar.

Dashboard: Bir panelin (ya da bir menünün) stilini belirler. Dikey olarak sıralanmış ve sola yaslanmış bir düzen sağlar. Arka plan rengi, dolgu ve kenarlık ayarları vardır.

Button: Butonların stilini tanımlar. Arka plan rengi, yazı rengi, kenarlık, dolgu ve köşe yuvarlama gibi özellikleri ayarlar. Fare üzerine gelindiğinde rengin değişmesini sağlayan bir hover etkisi eklenmiştir.

Form Container: Bir formun stilini belirler. Görüntülenmemiş (display: none) bir şekilde başlatılır ve belirli bir tetikleyici ile görünür hale getirilebilir. Arka plan rengi, dolgu, kenarlık, gölge, genişlik ve yükseklik gibi özellikleri ayarlar.

Form Elementleri: Form elemanlarının (input, button, label vb.) stilini tanımlar. Genişlik, yükseklik, dolgu, kenarlık ve köşe yuvarlama gibi özellikleri ayarlar. Butonun üzerine gelindiğinde rengin değişmesini sağlayan bir hover etkisi eklenmiştir.

Table: Tablonun stilini belirler. Kenarlık çizgileri, hücre içeriği, arka plan rengi gibi özellikleri ayarlar.

III. SONUÇLAR

Projede, hastane yönetim sistemi tasarlandı ve geliştirildi. Veritabanı normalizasyon kurallarına uygun olarak oluşturuldu. Nesne yönelimli programlama prensipleri kullanıldı. Arayüz geliştirmeleri yapıldı ve kullanıcıların kolayca kullanabileceği bir arayüz sağlandı.

Biraz daha açmak gerekirse:

Hasta Yönetimi: Hasta eklemesi veya silinmesi, hasta bilgilerinin güncellenmesi gibi işlemler arayüz üzerinden kolayca gerçekleştirilebilir hale getirildi. Ayrıca, bu işlemler veritabanında gerçekleştiğinde ilgili tablolardaki güncellemeler anlık olarak takip edilebilir.

Doktor Yönetimi: Benzer şekilde, doktor ekleme, silme ve güncelleme işlemleri için arayüz üzerinden kullanıcı dostu bir ortam sağlandı. Aktif randevusu bulunan bir doktor silinemez şekilde tasarlandı.

Randevu Yönetimi: Hastaların belirli bir doktorla randevu alması veya randevusunu iptal etmesi, bu işlemlerin doğru bir şekilde veritabanına yansıtılmasıyla desteklendi. Randevuların tarihi, saati gibi bilgilerin yanı sıra randevuya ilişkin detaylar görüntülendi.

Tıbbi Rapor Yönetimi: Hastaların tıbbi raporlarını ekleyebilmesi ve gerektiğinde güncelleyebilmesi sağlandı. Bu raporların veritabanına doğru bir şekilde kaydedilip kaydedilmediği düzenli olarak kontrol edildi.

Güvenlik Önlemleri: Hassas verilerin korunması için HTTPS protokolleri ve şifreleme yöntemleri kullanıldı. Bu sayede, kullanıcı verilerinin gizliliği ve güvenliği sağlandı.

Arayüz Geliştirmeleri: Kullanıcıların kolayca erişebileceği, kullanıcı dostu bir arayüz geliştirildi. AJAX çağrıları kullanılarak sayfa yenilenmeden dosya yükleme ve indirme gibi işlemler gerçekleştirildi. Kullanıcıların tıbbi geçmişlerini, tedavi notlarını ve randevu geçmişlerini görebileceği dashboardlar eklenerek kullanıcı deneyimi artırıldı.

Sonuç olarak, bu projenin başarılı bir şekilde tamamlanmasıyla birlikte bir hastane yönetim sistemi sağlam altyapıya sahip, güvenli ve kullanıcı dostu bir platform haline geldi. Kullanıcıların sağlıkla ilgili işlemlerini ko-

layca gerçekleştirebilmeleri için gerekli olan tüm işlevselliği içermektedir.

REFERENCES

[1] https://www.guru99.com/tr/database-normalization.html
[2] https://www.freepik.com/
[3] https://dbdiagram.io/home
[4] https://code2flow.com/
[5] https://gist.github.com/emrekgn/493304c6445de15657b2
[6] https://gist.github.com/emrekgn/b4049851c88e328c065a

IV. ER DİYAGRAMI VE AKIŞ ŞEMALARI

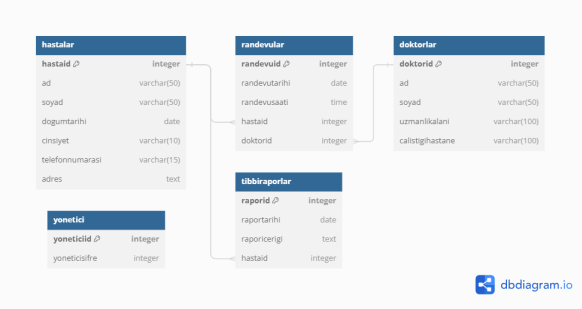

Fig. 1: ERD

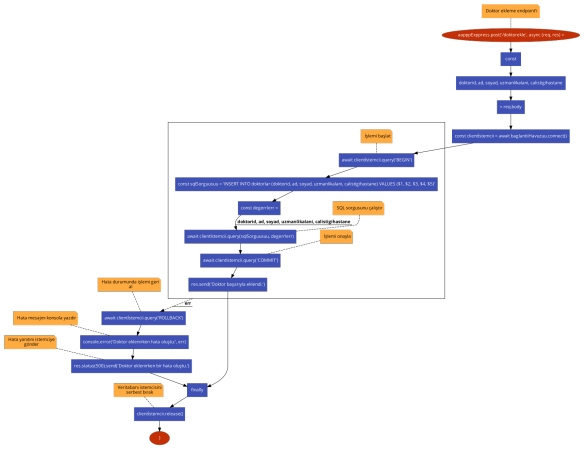

Fig. 2: doktor ekleme fonksiyonu

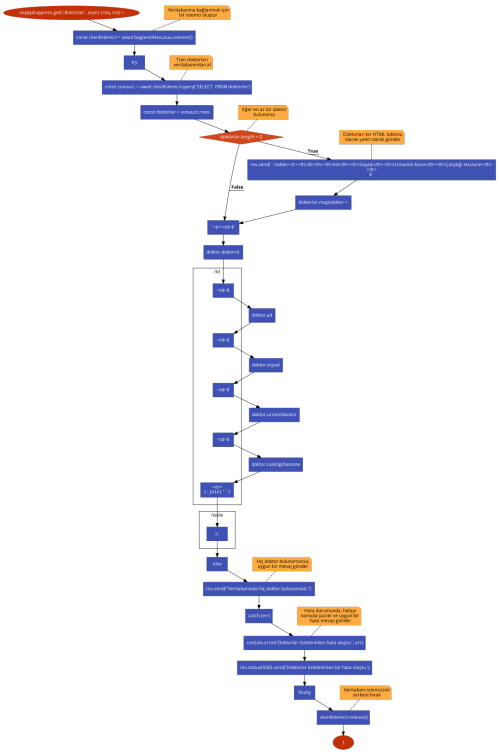

Fig. 3: doktor listeleme fonksiyonu

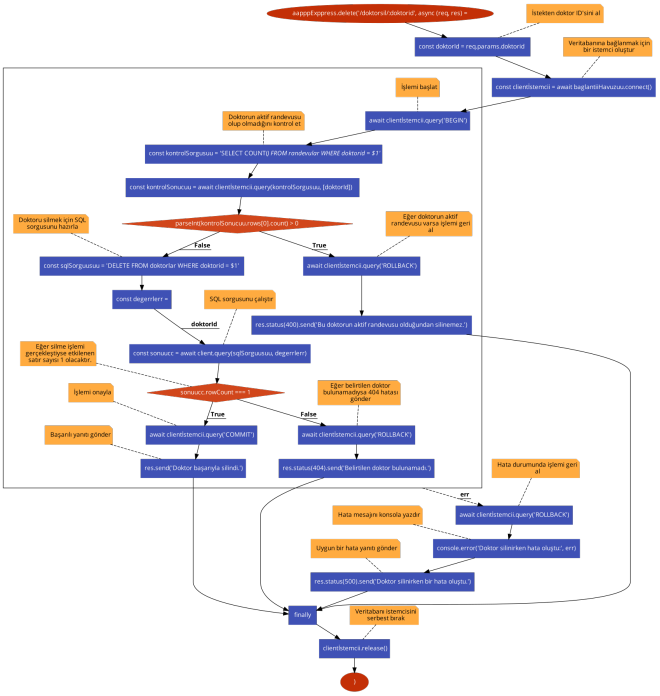

Fig. 4: doktor silme fonksiyonu

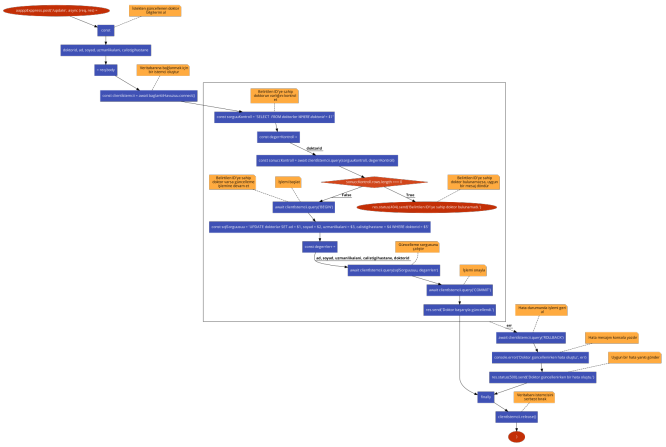

Fig. 5: doktor güncelle

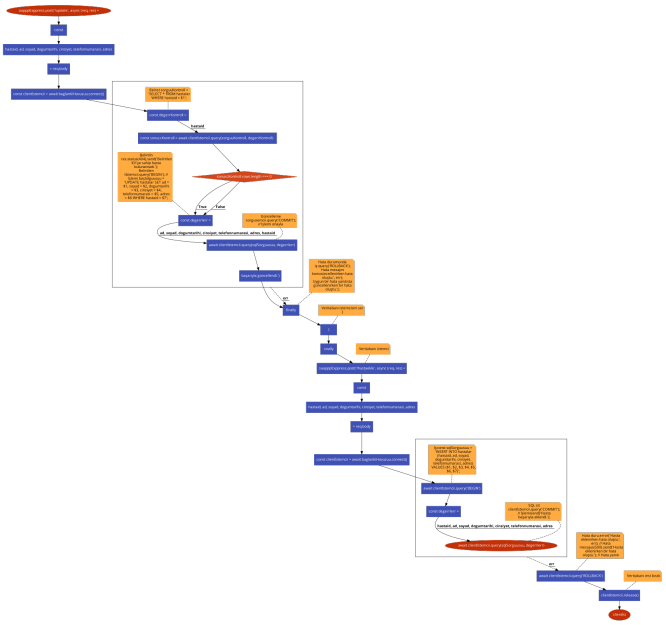

Fig. 6: hasta fonksiyonu-1

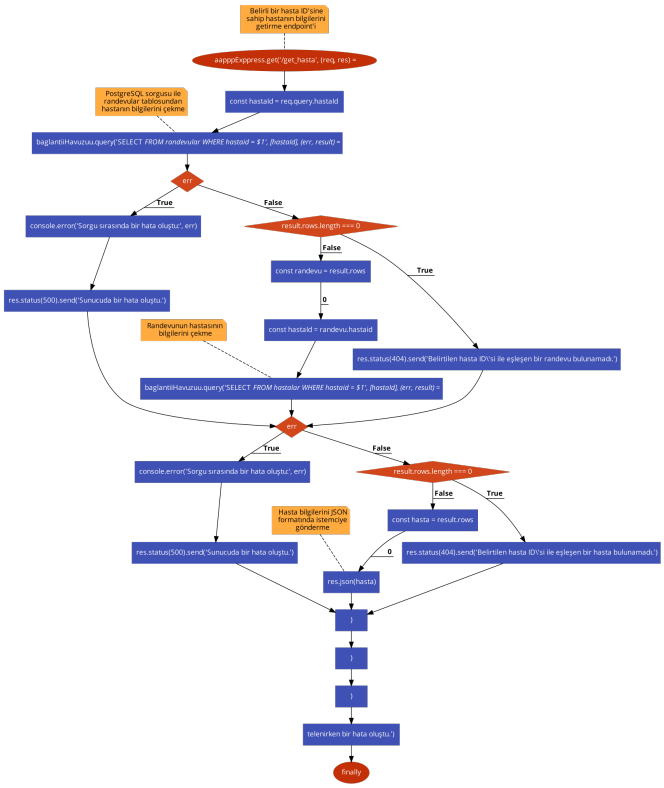

Fig. 7: hasta arama fonksiyonu

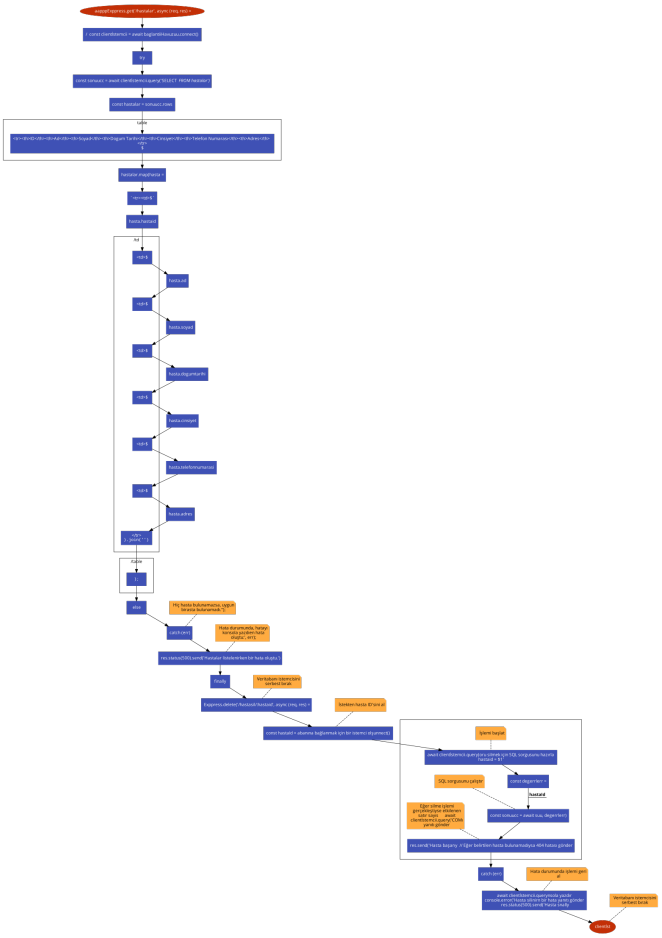

Fig. 8: hasta fonksiyonu-2

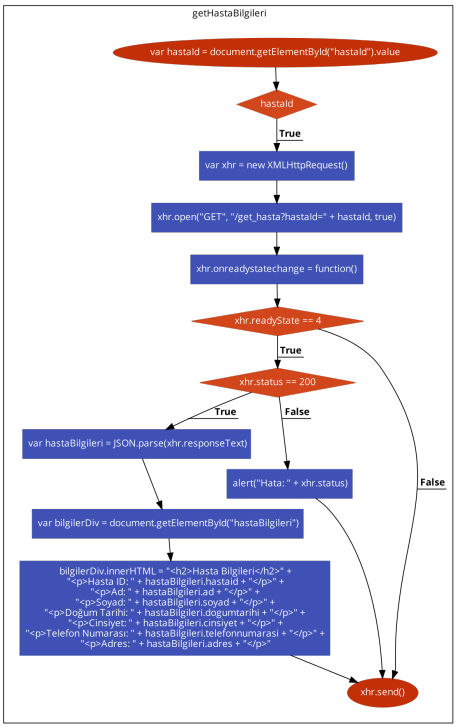

Fig. 9: hasta bilgisi fonksiyonu

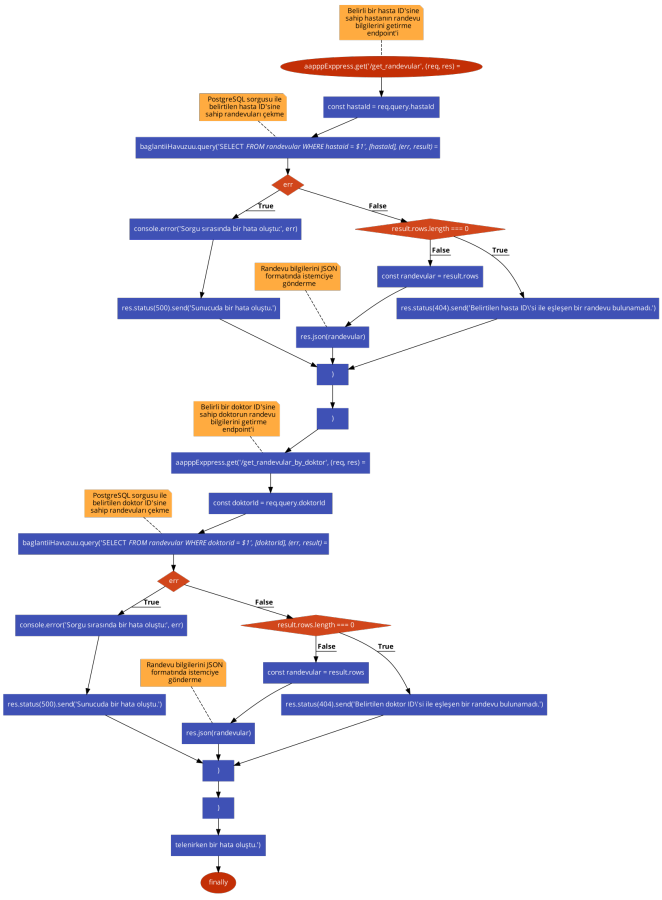

Fig. 10: randevu arama fonksiyonu

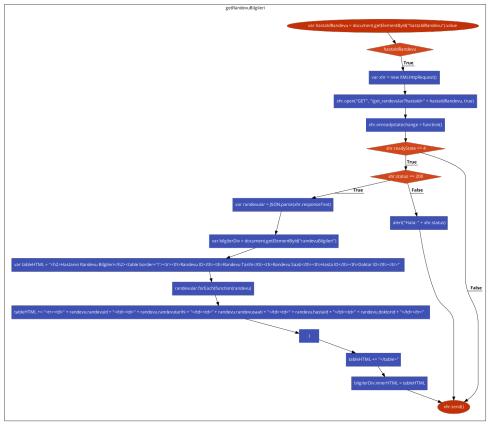

Fig. 11: randevu bilgisi fonksiyonu

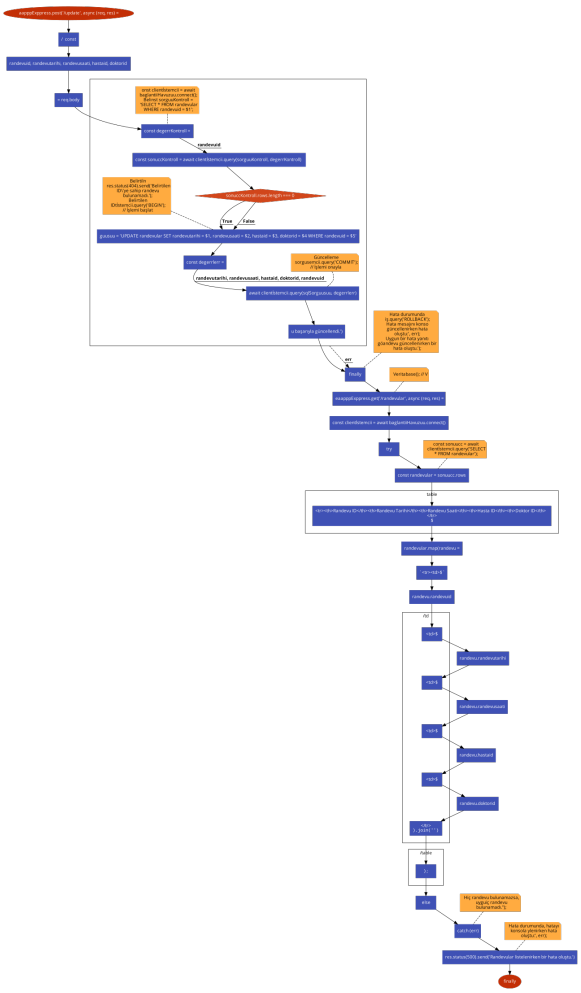

Fig. 13: randevu fonksiyonu-2

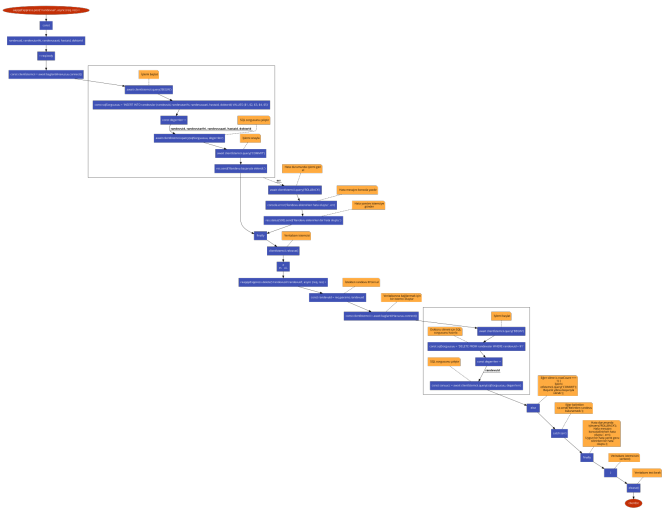

Fig. 12: randevu fonksiyonu-1