**İÇİNDEKİLER**

**Özet …………………………………………………………………………………………………. 2**

1. **TABLOLAR …………………………………………………………………….. 2**

**1.1 Poliklinik Tablosu (Polyclinic) ……………………………………………………………. 3**

**1.2 Hasta Tablosu (Patient) …………………………………………………………………… 3**

**1.3 Doktor Tablosu (Doctor) ………………………………………………………………….. 3**

**1.4 Randevu Tablosu (Appointment) ………………………………………………………… 3**

1. **SQL SORGULARI ……………………………………………………………... 4**

**2.1 Database Oluşturma ……………………………………………………………………... 4**

**2.2 Tabloları Oluşturma .……………………………………………………………………... 4**

**2.3 Tablolara Veri Ekleme …………………………………………………………………… 5**

**2.4 Tetikleyiciler ………………………………………………………………………………. 5**

**2.5 Prosedürler ………………………………………………………………………………... 7**

**2.6 Diğer Sorgular …………………………………………………………………………….. 12**

**3. PROJE TANITIMI ……………………………………………………………… 13**

**3.1 Randevu Alma ……………………………………………………………………………... 14**

**3.2 Randevu İptali ……………………………………………………………………………... 14**

**3.3 İstatistikler ……………………………………………………………………………..…... 15**

**3.4 Kullanılan Teknolojiler………..…………………………………………………………... 16**

**3.5 Proje Linki …..……………………………………………………………………………... 17**

**3.6 Hazırlayanlar …..…………………………………………………………………………... 17**

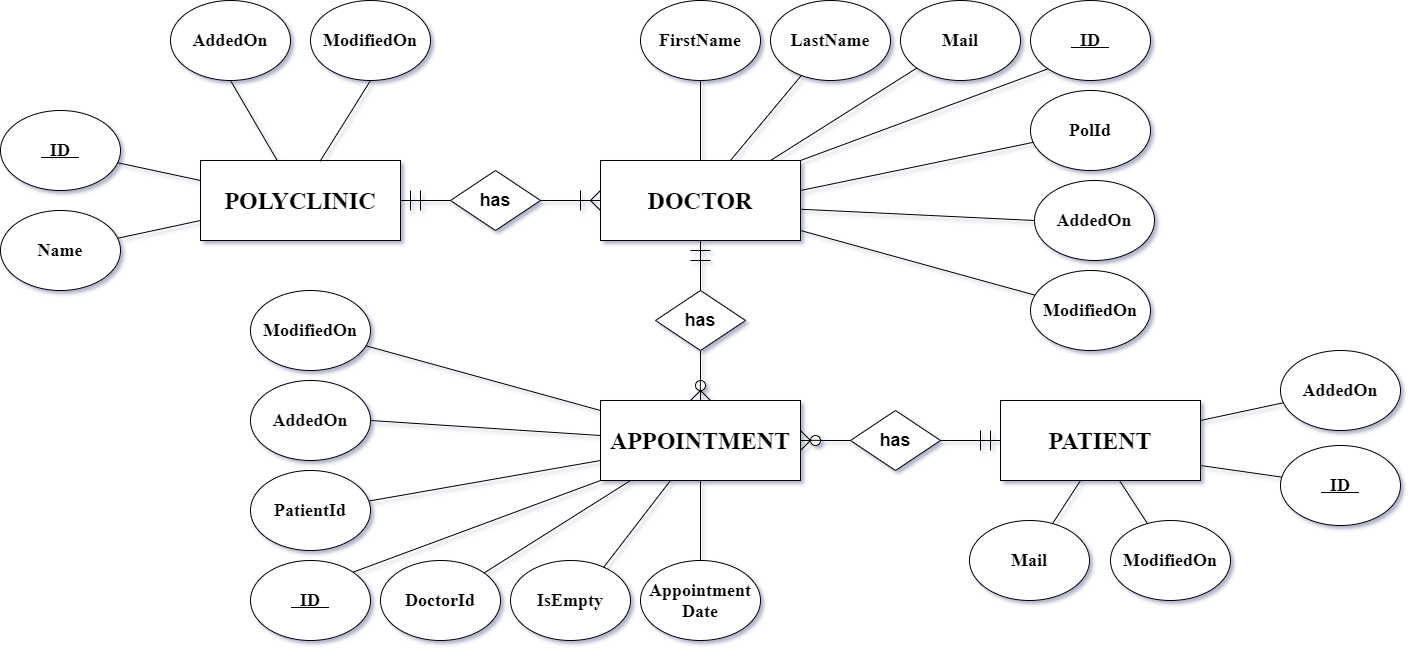
**PAINT++ HASTANE RANDEVU OTOMASYONU**

**ÖZET**

Üç poliklinik ve 6 doktor içeren özel bir hastanenin randevu sisteminde hasta TC kimlik numarası ve e-postası ile giriş randevu alabilmekte ve aynı şekilde TC kimlik numarasını girerek randevusunu iptal edebilmektedir. Ayriyeten istatistikler menüsü altında çeşitli bilgilere erişilebilmektedir.

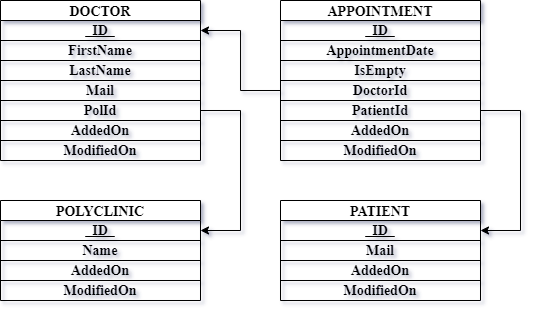
**1.TABLOLAR**

Veri tabanında; Doktor, Randevu, Hasta ve Poliklinik olmak üzere toplam dört adet tablo bulunmaktadır. Tabloların kendi içinde ve birbirleriyle olan ilişkileri Şekil 1.1’de gösterilmiştir.



**Şekil 1.1** *Genel Varlık Bağıntı Çizeneği*

Şekil 1.2’de ise tabloların **Foreign Key** ilişkileri gösterilmiştir.



**Şekil 1.2** *Tabloların Foreign Key İlişkisi*

**1.1 Poliklinik Tablosu (Polyclinic)**

Poliklinik tablosu; ID, Poliklinik Adı, Eklenme Tarihi ve Düzenleme Tarihi olmak üzere dört adet sütündan oluşmaktadır. Burada ID, Primary Key’dir ve Doktor tablosunda Foreign Key olan PolId ile Inner Join yapılabilmektedir. Poliklinik adı, polikliniğin adıdır. Kalan iki sütun ise sıralama yapmayı kolaylaştırmak adına tüm tablolarda bulunmaktadır.

**1.2 Hasta Tablosu (Patient)**

Hasta tablosu; ID, Mail, Eklenme Tarihi ve Düzenleme Tarihi olmak üzere dört adet sütündan oluşmaktadır. ID, Primary Key’dir ve hasta TC kimlik numarası ile giriş yaptığında bu onun ID’si olur. Aynı zamanda Randevu tablosunda Foreign Key olan PatienId ile Inner Join yapılabilmektedir. Mail, yine aynı şekilde giriş yaparken yazılan e-posta adresidir ve randevu alındığında veya iptal edildiğinde hastaya bilgi maili gönderilmesi için istenmektedir. Eklenme ve değiştirilme tarihi tüm tablolarda olduğu gibi mevcuttur.

**1.3 Doktor Tablosu (Doctor)**

Doktor tablosu; ID, Ad, Soyad, Mail, Poliklinik ID’si, Eklenme Tarihi ve Düzenleme Tarihi olmak üzere yedi adet sütündan oluşmaktadır. ID, Primary Key’dir ve sadece doktora özeldir. Randevu tablosunda Foreign Key olan DoctorId ile Inner Join yapılabilmektedir. Ad ve soyad doktorun adı ve soyadıdır. Mail, doktorun mail adresidir ve kendisine randevu alındığında veya iptal edildiğinde bilgi maili gönderilmektedir. Poliklinik ID’si, bağlı olduğu polikliniğin ID’sidir ve Poliklinik tablosunda Primary Key olan ID ile Inner Join yapılabilmektedir. Eklenme ve değiştirilme tarihi tüm tablolarda olduğu gibi mevcuttur.

**1.4 Randevu Tablosu (Appointment)**

Randevu tablosu; ID, Randevu Tarihi, IsEmpty, Doktor ID’si, Hasta ID’si, Eklenme Tarihi ve Düzenleme Tarihi olmak üzere yedi adet sütündan oluşmaktadır. ID, Primary Key’dir ve randevu alındıkça otomatik olarak artar. Randevu tarihi, hastanın seçtiği randevu tarihidir. IsEmpty, o tarihteki randevunun boş olup olmadığını kontrol etmeyi sağlar. DoctorId, randevunun alındığı doktorun ID’sidir ve Doktor tablosunda Primary Key olan ID ile Inner Join yapılabilmektedir. PatientId, randevuyu alan hastanın ID’si ve Hasta tablosunda Primary Key olan ID ile Inner Join yapılabilmektedir. Eklenme ve değiştirilme tarihi tüm tablolarda olduğu gibi mevcuttur.

**2.SQL SORGULARI**

**2.1 Database Oluşturma**

CREATE DATABASE Hospital

USE Hospital

İlk sorgu ile ‘Hospital’ adına bir veritabanı oluştulur. İkinci sorguyla, bu andan sonra yazılacak sorguların ‘Hospital’ veritabanı üzerinde işlem yapması sağlanır.

**2.2 Tabloları Oluşturma**

CREATE TABLE Polyclinic

(

Id int PRIMARY KEY NOT NULL,

Name varchar(255) NOT NULL,

AddedOn datetime NOT NULL,

ModifiedOn datetime NULL

)

CREATE TABLE Doctor

(

Id int PRIMARY KEY NOT NULL,

FirstName varchar(255) NOT NULL,

LastName varchar(255) NOT NULL,

Mail varchar(255) NOT NULL,

PolId int FOREIGN KEY REFERENCES Polyclinic(Id),

AddedOn datetime NOT NULL,

ModifiedOn datetime NULL

)

CREATE TABLE Patient

(

Id varchar(255) PRIMARY KEY NOT NULL,

Mail varchar(255) NOT NULL,

AddedOn datetime NOT NULL,

ModifiedOn datetime NULL

)

CREATE TABLE Appointment

(

Id int IDENTITY(0,1) PRIMARY KEY NOT NULL,

AppointmentDate datetime NOT NULL,

IsEmpty BIT NOT NULL,

DoctorId int FOREIGN KEY REFERENCES Doctor(Id),

PatientId varchar(255) FOREIGN KEY REFERENCES Patient(Id),

AddedOn datetime NOT NULL,

ModifiedOn datetime NULL

)

Burada bulunan sorgular ile ‘Tablolar’ başlığı altında bahsedilen tablolar oluşturulmuştur. Her tabloda bulunan ‘Id’ sütunu Primary Key’dir ve Randevu tablosunda sıfırdan başlayıp yeni randevu alındıkça birer birer artmaktadır.

**2.3 Tablolara Veri Ekleme**

INSERT INTO Polyclinic

VALUES

(1, 'Kardiyoloji','2021-05-20 00:00:00.000', NULL),

(2, 'Ortopedi ve Travmatoloji','2021-05-20 00:00:00.000', NULL),

(3, 'Dahiliye', '2021-05-20 00:00:00.000', NULL)

INSERT INTO Doctor

VALUES

(1, 'Utku', 'Kaya', 'gumus\_utku@paint.com', 1, '2021-05-20 00:00:00.000', NULL),

(2, 'Fatma', 'Aslan', 'yesim\_oztas@paint.com', 1, '2021-05-20 00:00:00.000', NULL),

(3, 'Kemal', 'Baycan', 'baycan\_kemal@paint.com', 2, '2021-05-20 00:00:00.000', NULL),

(4, 'Yaren', 'Atan', 'atan\_nazli@paint.com', 2, '2021-05-20 00:00:00.000', NULL),

(5, 'Fatih', 'Kuyu', 'kuyu\_fatih@paint.com', 3, '2021-05-20 00:00:00.000', NULL),

(6, 'Hatice', 'Sayın', 'sayin\_hatice@paint.com', 3, '2021-05-20 00:00:00.000', NULL)

INSERT INTO Patient

VALUES

('52621459889', 'kayhan21.coc@hotmail.com', '2021-05-20 00:00:00.000', NULL),

('94367459231', 'barisOkta18@gmail.com', '2021-05-20 00:00:00.000', NULL),

('49412459247', 'nur.15kirit@yahoo.com','2021-05-20 00:00:00.000', NULL),

('23621156863', 'hatic25esu@ismu.edu.tr', '2021-05-20 00:00:00.000', NULL),

('34526732824', 'kaykilmaz\_suzeynep93@gmail.com', '2021-05-20 00:00:00.000', NULL)

Bu hastanede ‘Kardiyoloji’, ‘Ortopedi ve Travmatoloji’ ve ‘Dahiliye’ olmak üzere toplam üç poliklinik ve her poliklinikte hizmet veren iki doktor olmak üzere toplamda altı doktor hizmet vermektedir. Doktorların bilgileri yukarıda verilmiştir.

İstatistiklerin değerlendirilebilmesi için örnek hasta ve randevu verileri, veri tabanı üzerinden eklenmiştir. Bir hastanın birden çok randevusu olabilir veya aldığı randevuyu iptal ettiyse hiç randevusu olmayabilir.

**2.4 Tetikleyiciler**

CREATE TRIGGER SendAppointmentGet

ON Appointment

AFTER INSERT

AS

BEGIN

SET NOCOUNT ON;

DECLARE @LastMail varchar(255)

Select TOP 1 @LastMail = Patient.Mail FROM Patient INNER JOIN Appointment ON Patient.Id = Appointment.PatientId ORDER BY Appointment.AddedOn DESC

DECLARE @LastDate varchar(255)

SELECT TOP 1 @LastDate = Appointment.AppointmentDate FROM Appointment ORDER BY Appointment.AddedOn DESC

DECLARE @body varchar(255)= 'Your appointment saved. Appointment date is ' + @LastDate + '. Please be at the hospital 15 minutes before your appointment.'

EXEC msdb.dbo.sp\_send\_dbmail

@recipients = @LastMail,

@profile\_name = 'HospitalMail',

@subject = 'New Appointment',

@body =@body

END

‘SendAppointmentGet’ tetikleyicisi randevu alındığında, randevu alan hastanın mail adresine randevu tarihiyle birlikte randevunun alındığı bilgisini iletir.

CREATE TRIGGER SendAppointmentDoctor

ON Appointment

AFTER INSERT

AS

BEGIN

SET NOCOUNT ON;

DECLARE @DocMail varchar(255)

Select TOP 1 @DocMail = Doctor.Mail FROM Doctor INNER JOIN Appointment ON Doctor.Id = Appointment.DoctorId ORDER BY Appointment.AddedOn DESC

DECLARE @LastDate varchar(255)

SELECT TOP 1 @LastDate=Appointment.AppointmentDate FROM Appointment ORDER BY Appointment.AddedOn DESC

DECLARE @DocName varchar(255)

SELECT TOP 1 @DocName = Doctor.FirstName + ' ' + Doctor.LastName FROM Doctor INNER JOIN Appointment ON Doctor.Id = Appointment.DoctorId ORDER BY Appointment.AddedOn DESC

DECLARE @body varchar(255)= 'Doctor ' + @DocName + ',' + CHAR(13) + CHAR(10) +'You have a new appointment on ' + @LastDate + '. For your information.'

EXEC msdb.dbo.sp\_send\_dbmail

@recipients = @DocMail,

@profile\_name = 'HospitalMail',

@subject = 'New Appointment',

@body =@body

END

‘SendAppointmentDoctor’ tetikleyicisi randevu alındığında, randevu alınan doktorun mail adresine randevu tarihiyle birlikte kendisine randevu alındığı bilgisini iletir. @body değişkeni içinden kullanılan CHAR(13) + CHAR(10) komutları metin içerisinde bir alt satıra geçmeyi sağlar.

CREATE TRIGGER SendAppointmentDelete

ON Appointment

AFTER DELETE

AS

BEGIN

SET NOCOUNT ON;

DECLARE @LastMail varchar(255)

Select TOP 1 @LastMail = Patient.Mail FROM Patient INNER JOIN deleted ON Patient.Id = deleted.PatientId ORDER BY deleted.AddedOn DESC

DECLARE @LastDate varchar(255)

SELECT TOP 1 @LastDate = deleted.AppointmentDate FROM deleted ORDER BY deleted.AddedOn DESC

DECLARE @body varchar(255)= 'Your appointment on ' + @LastDate + ' has been cancelled.'

EXEC msdb.dbo.sp\_send\_dbmail

@recipients = @LastMail,

@profile\_name = 'HospitalMail',

@subject = 'Appointment Cancel',

@body =@body

END

‘SendAppointmentDoctor’ tetikleyicisi alınmış bir randevu iptal edildiğinde, hastanın mail adresine randevu tarihiyle birlikte randevunun iptal edildiği bilgisini iletir.

CREATE TRIGGER SendAppointmentCancelDoctor

ON Appointment

AFTER DELETE

AS

BEGIN

SET NOCOUNT ON;

DECLARE @DocMail varchar(255)

Select TOP 1 @DocMail = Doctor.Mail FROM Doctor INNER JOIN deleted ON Doctor.Id = deleted.DoctorId ORDER BY deleted.AddedOn DESC

DECLARE @LastDate varchar(255)

SELECT TOP 1 @LastDate = deleted.AppointmentDate FROM deleted ORDER BY deleted.AddedOn DESC

DECLARE @DocName varchar(255)

SELECT TOP 1 @DocName = Doctor.FirstName + ' ' + Doctor.LastName FROM Doctor INNER JOIN deleted ON Doctor.Id = deleted.DoctorId ORDER BY deleted.AddedOn DESC

DECLARE @body varchar(255)= 'Doctor ' + @DocName + ',' + CHAR(13) + CHAR(10) +'Your appointment on ' + @LastDate + ' has been cancelled. For your information.'

EXEC msdb.dbo.sp\_send\_dbmail

@recipients = @DocMail,

@profile\_name = 'HospitalMail',

@subject = 'Appointment Cancel',

@body =@body

END

‘SendAppointmentCancelDoctor’ tetikleyicisi randevu iptal edildiğinde, randevu alınan doktorun mail adresine iptal edilen randevu tarihiyle birlikte randevunun iptal edildiği bilgisini iletir.

**2.5 Prosedürler**

CREATE PROCEDURE IncSalaryDocsMail

AS

DECLARE @Mail varchar(255)

DECLARE @DocName varchar(255)

DECLARE @Count int = 1

DECLARE @Last int

SELECT TOP 1 @Last = Doctor.Id FROM Doctor ORDER BY Id DESC

WHILE @Count <= @Last

BEGIN

SELECT @Mail = Doctor.Mail, @DocName = Doctor.FirstName + ' ' + Doctor.LastName FROM Doctor WHERE Id = @Count

DECLARE @body varchar(255)= 'Doctor ' + @DocName + ',' + CHAR(13) + CHAR(10) + 'Your salary has been increased by $250. Stay healthy.'

EXEC msdb.dbo.sp\_send\_dbmail

@recipients = @Mail,

@profile\_name = 'HospitalMail',

@subject = 'Salary Update',

@body =@body

SET @Count += 1

END

‘IncSalaryDocsMail’ prosedürü, toplu mail atar. Burada tüm doktorların maaşında 250$ artış yapıldığı bildiriliyor.

**DeleteAppointment**

CREATE PROCEDURE DeleteAppointment @Id int

AS

DELETE FROM Appointment WHERE Id=@Id

Arayüz kısmında iptal edilmesi için işlem yapılan Randevu’ya ait id DeleteAppointment procedure üne parametre olarak verilir ve randevu silme işlemi procedure içerisindeki sorguya göre yapılır.

**DeleteDoctor**

CREATE PROCEDURE DeleteDoctor @Id int

AS

DELETE FROM Doctor WHERE Id=@Id

Silinecek olan doktora ait Id DeleteDoctor procedure’üne parametre olarak verilir ve bu idye sahip olan doktorun silinme işlemi procedure içerisindeki sorguya göre yapılır.

**DeletePatient**

CREATE PROCEDURE DeletePatient @Id int

AS

DELETE FROM Patient WHERE Id=@Id

Kullanıcı kendisini sistemden silmek isteyebilir veya Hastane yönetimini kararıyla bu kullanıcı sistemden çıkartılabilir. Silinecek olan kullanıcıya ait Id(TC Kimlik No) DeletePatient procedure’üne parametre olarak verilir ve bu id ye ait hasta kullanıcısının sistemden silinme işlemi procedure çalıştırılarak yapılır.

**DeletePolyclinic**

CREATE PROCEDURE DeletePolyclinic @Id int

AS

DELETE FROM Polyclinic WHERE Id=@Id

Hastanedeki değişikliklere göre kaldırılması icap eden polikliniğin sistemden kaldırılması için DeletePolyclinic procedure’üne silinmesi istenen polikliniği id’si parametre olarak verilir ve bu id ye ait polikliniğin sistemden silinme işlemi procedure çalıştırılarak gerçekleştirilir.

**GetAppointmentById**

CREATE PROCEDURE GetAppointmentById @Id int

AS

SELECT \* FROM Appointment WHERE Id=@Id

Id’ye göre tek bir randevu getirmeye yarayan procedure.

**GetAppointmentsByPatientId**

CREATE PROCEDURE GetAppointmentsByPatientId @patientId varchar(50)

AS

SELECT \* FROM Appointment WHERE PatientId=@patientId

Hastanın kendisinin geçmiş ve gelecekteki tüm randevu kayıtlarına erişebilmesi için GetAppointmensByPatienId procedure’üne hastanın Id’si(TC Kimlik Numarası) parametre olarak verilir ve hastaya ait tüm randevulara ait bilgiler getirilir.

**GetAppointmentsDetailByDateRange**

CREATE PROCEDURE GetAppointmentsDetailByDateRange @PatientId varchar(11), @StartingTime DateTime

AS

SELECT AppointmentDate, Doctor.FirstName+' ' +Doctor.LastName as DoctorName,Polyclinic.Name AS PolyclinicName FROM Appointment,Doctor,Polyclinic

WHERE Appointment.PatientId=@PatientId AND Appointment.DoctorId=Doctor.Id AND Polyclinic.Id=Doctor.PolId AND Appointment.AddedOn BETWEEN @StartingTime AND GETDATE();

Bir hastaya ait randevuları günümüz ve belirli tarih aralığında filtrelemek için GetAppointmentsDetailByDateRange procedure’üne hastanın Id’si ve günümüzden kaç gün öncesine kadar olan verilerin getirilmesini istiyorsak o tarihi parametre olarak verilir ve hastaya ait randevu bilgileri; Randevunun EKLENME tarihin belirtilen tarih ve günümüz arasında olacak şekilde getirilir.

**GetDoctorById**

CREATE PROCEDURE GetDoctorById @Id int

AS

SELECT \* FROM Doctor WHERE Id=@Id

Bilgileri getirilmesi istenen doktora ait Id GetDoctorById procedure’üne parametre olarak verilir ve id si verilen doktora ait bilgiler bu procedure çalıştırılarak getirilir.

**GetDoctorsByPolId**

CREATE PROCEDURE GetDoctorsByPolId @polId int

AS

SELECT \* FROM Doctor WHERE PolId=@PolId

Belirli bir poliklinikte çalışan tüm doktorların listelenmesi için GetDoctorsByPolId procedure’üne poliklinik id si parametre olarak verilir ve id si verilen poliklinikte bulunan tüm doktorlara ait bilgiler bu procedure çalıştırılarak getirilir.

**GetPatientById**

CREATE PROCEDURE GetPatientById @Id varchar(50)

AS

SELECT \* FROM Patient WHERE Id=@Id

Bilgileri getirilmesi istenen hastaya ait Id(TC Kimlik Numarası) GetPatientById procedure’üne parametre olarak verilir ve id si verilen hastaya ait bilgiler bu procedure çalıştırılarak getirilir

**GetPolyclinicById**

CREATE PROCEDURE GetPolyclinicById @Id int

AS

SELECT \* FROM Polyclinic WHERE Id=@Id

Bilgileri getirilmesi istenen polikliniğe ait Id GetPolyclinicById procedure’üne parametre olarak verilir ve id si verilen polikliniğe ait bilgiler bu procedure çalıştırılarak getirilir.

**GetAppointmentCountByDoctorId**

CREATE PROCEDURE GetAppointmentCountByDoctorId @doctorId int, @date varchar(255)

AS

Select count(\*) from Appointment where doctorId=@doctorId and AppointmentDate Between @date and GETDATE()

Belirtilen bir doktorun belirtilen bir tarihten günümüze kadar olan randevu sayısını bulmak için GetAppointmentCountByDoctor id procedure’üne doktorun id si ve hangi tarihten itibaren randevuların sayılması isteniyorsa o tarih parametre olarak verilir ve procedure çalıştırılarak istenen veri eldi edilir.

Örneğin EXEC GetAppointmentCountByDoctorId 1,'2021-05-20' in çıktısı şu şekildedir;



**Şekil 2.5.1** *GetAppointmentCountByDoctorId Örnek Çıktı*

20 Mayıs 2021 tarihinden günümüze 1 id’sine sahip olan doktora 2 randevu alındığı belirtiliyor.

**FindAppointmentCountWithPolyclinicName**

CREATE PROCEDURE FindAppointmentCountWithPolyclinicName

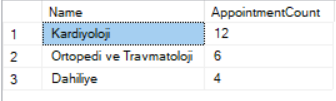
AS

SELECT Polyclinic.Name, Count(\*) AS 'AppointmentCount' FROM Appointment

INNER JOIN Doctor ON Doctor.Id=Appointment.DoctorId

INNER JOIN Polyclinic ON Polyclinic.Id=Doctor.PolId group by Polyclinic.Name Order By 'AppointmentCount' DESC

Poliklinikleri randevu sayısına göre büyükten küçüğe listelemek için FindAppointmentCountWithPolyclinicName procedure’u çalıştırılır ve poliklinikler randevu sayılarıyla birlikte büyükten küçüğe sıralanmış bir şekilde getirilir.



**Şekil 2.5.2** *Randevu Sayılarına Göre Poliklinik Sıralaması*

**FindWeekDayWithAppointmentCount**

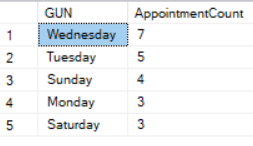
CREATE PROCEDURE FindWeekDayWithAppointmentCount

AS

SELECT DATENAME(weekday,AppointmentDate) as 'GUN',Count(AppointmentDate) as 'AppointmentCount'

From Appointment Group by DATENAME(weekday,AppointmentDate) Order By AppointmentCount DESC;

En çok randevu alınan günlerin büyükten küçüğe listeyelerek getirmek için FindWeekDayWithAppointmentCount procedure’ü çalıştırılır ve günler randevu sayılarıyla birlikte büyükten küçüğe sıralanmış bir şekilde getirilir.



**Şekil 2.5.3** *En Çok Randevu Alınan Gün Sıralaması*

(Programda normal şartlarda Cumartesi ve Pazar günü randevu alınamamaktadır. Fakat biz deneme amaçlı randevu alımlarında GETDATE() metodunu kullandık.)

**FindAppointmentCountWithHour**

CREATE PROCEDURE FindAppointmentCountWithHour @StartTime int, @EndTime int

AS

SELECT DATEPART(HOUR,AppointmentDate) as HOURS, Count(\*) as AppointmentCount from Appointment

WHERE DATEPART(HOUR,AppointmentDate) BETWEEN @StartTime AND @EndTime Group By DATEPART(HOUR,AppointmentDate)

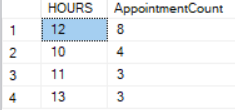
Order By AppointmentCount DESC

Belirli saat aralıklarındaki saatleri randevu sayılarına göre büyükten küçüğe listeleyerek getirmek için FindAppointmentCountWithHour metoduna aralığın başlangıç saati ve aralığın bitiş saati parametre olarak verilir.

Örneğin: Saat 10 ile 16 saatleri arasındaki randevu dağılımını büyükten küçüğe olacak bir şekilde listelemek için;

EXEC FindAppointmentCountWithHour 10,16 şeklinde procedure çalıştırılır.

Çıktısı şu şekilde;



**Şekil 2.5.4** *En Çok Randevu Alınan Saat Sıralaması*

Saat 12:00-12:59 aralığında bu zamana kadar 8 randevu alınmış, Saat 10-10:59 aralığında 4 vb..

**2.6 Diğer Sorgular**

---- Randevu sayılarına göre tarihleri ve randevu sayılarını randevu sayısına göre büyükten küçüğe getiriyor----

Select Convert(date, AppointmentDate) AS Date, Count(Convert(date, AppointmentDate)) As AppointmentCount from Appointment Group By Convert(date, AppointmentDate) Order By AppointmentCount Desc

----- En Az Randevu Alınan Günü ve O güne ait randevu sayısını veriyor—

Select TOP 1 \* from ( Select Convert(date, AppointmentDate) AS Date, Count(Convert(date, AppointmentDate)) As AppointmentCount from Appointment Group By Convert(date, AppointmentDate))As EnDusukGun Order By AppointmentCount DESC

---- En fazla randevu alınan günü ve o güne ait randevu sayısını veriyor---

Select TOP 1 \* from ( Select Convert(date, AppointmentDate) AS Date, Count(Convert(date, AppointmentDate)) As AppointmentCount from Appointment Group By Convert(date, AppointmentDate))As EnYuksekGun Order By AppointmentCount DESC

--- Doktorları ve randevu sayılarını randevu sayısına göre sıralayarak getirir—

Select Doctor.FirstName + ' ' + Doctor.LastName as FullName,count(Appointment.DoctorId) as AppointmentCount from Doctor LEFT JOİN Appointment ON Doctor.Id=Appointment.DoctorId Group By Doctor.FirstName + ' ' +Doctor.LastName Order By AppointmentCount DESC

--- Tüm randevu sayılarını getirir ---

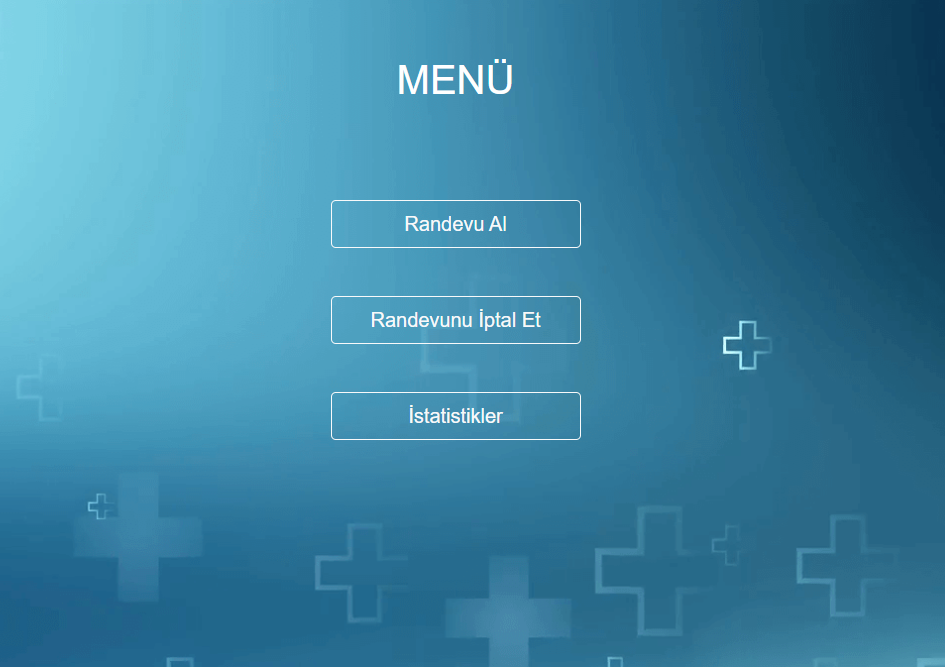
SELECT COUNT(\*) FROM Appointment

--- Tüm doktorları getirir---

SELECT \* FROM Doctor

**3.PROJE TANITIMI**

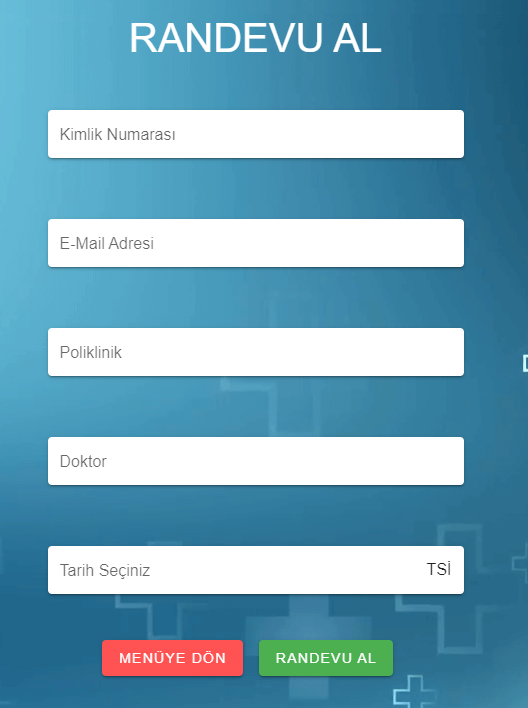
Paint++ Hastane Randevu Otomasyonu, bir hastanede bulunan üç poliklinik ve bu polikliniklerde bulunan altı doktora randevu almayı sağlayan bir veri tabanı ve web arayüzüdür. Temel olarak Randevu Alma, Randevu İptal ve İstatistik kısımlarından oluşmaktadır.

**Şekil 3.1** *Randevu Otomasyonu Ana Menü*

**3.1 Randevu Alma**

Bu menüde hasta TC Kimlik Numarası ve mail adresini girdikten sonra Poliklinik, Doktor ve Randevu Saati seçilip ‘Randevu Al’ tuşuna basılarak randevu alınır. Eğer yanlış menüye girildiyse ‘Menüye Dön’ tuşuna basılarak ana menüye dönülür.

Randevu başarıyla alındıysa bir bildirim ile gösterilir.



**Şekil 3.1.1** *Randevu Alma Ekranı*

**3.2 Randevu İptali**

Bir randevu iptal edilmek istenirse ana menüden ‘Randevunu İptal Et’ kısmına girilir. TC Kimlik Numarası yazıldıktan sonra ‘Randevuları Getir’ butonuna basılarak o numaraya ait randevular gösterilir. İptal edilmek istenilen randevu seçildikten sonra ‘Randevuyu İptal Et’ butonuna basılarak randevu iptal edilir.

**Şekil 3.2.1** *Randevu İptal Etme Ekranı*

**3.3 İstatistikler**

İstatistikler ekranında yer alan seçeneklere basılarak sonuçlar görüntülenebilir.

**Şekil 3.3.1** *Randevu Sayısına Göre Tarihlerin Sıralanması*

**3.4 Kullanılan Teknolojiler**

**Database :** Microsoft SQL Server

**Frontend :**

Vue.js

Vue.js [kullanıcı arayüzleri](https://tr.wikipedia.org/wiki/Kullan%C4%B1c%C4%B1_aray%C3%BCz%C3%BC) ve [tek sayfa uygulamalar](https://tr.wikipedia.org/wiki/Tek_sayfa_uygulamas%C4%B1) (single-page applications) inşa etmek için kullanılan [açık kaynak](https://tr.wikipedia.org/wiki/A%C3%A7%C4%B1k_kaynak) Javascript Framework’üdür.

**Backend :**

.NET 5

.NET, Windows, Linux ve macOS işletim sistemleri için ücretsiz ve açık kaynaklı, yönetilen bir bilgisayar yazılımı çerçevesidir.

Dapper

Dapper, Microsoft .NET platformu için bir nesne-ilişkisel haritalama ürünüdür: nesneye yönelik bir etki alanı modelini geleneksel bir ilişkisel veritabanına eşlemek için bir çerçeve sağlar. Amacı, geliştiriciyi ilişkisel veri kalıcılığı ile ilgili programlama görevlerinin önemli bir kısmından kurtarmaktır.

**3.5 Proje Linki**

<https://github.com/osmanyasircankaya/Hospital>

**3.6 Hazırlayanlar**

Bu proje aşağıda bilgileri verilen İstanbul Medeniyet Üniversitesi Bilgisayar Mühendisliği 3. Sınıf öğrencileri tarafından hazırlanmıştır.

Numara İsim - Soyisim

18120205020 Hüseyin Yılmaz

18120205030 Osman Yasir Çankaya

18120205041 Fatih Dursun Üzer

19120205058 Emirhan Aktürk