Beyaz Eşya Bayisi Otomasyon Sistemi

\*Note: Sub-titles are not captured in Xplore and should not be used

1st Tayfun Açıcı  
*Bilişim Sistemleri Mühendisliği*  
KOCAELİ ÜNİVERSİTESİKocaeli/Türkiye2nd Artun Kara   
*Bilişim Sistemleri Mühendisliği*  
KOCAELİ ÜNİVERSİTESİKocaeli/Türkiye3rd Elif Özkan   
*Bilişim Sistemleri Mühendisliği*  
KOCAELİ ÜNİVERSİTESİKocaeli/Türkiye

***Abstract***— **By using the white goods dealer automation system we have developed, it helps us to keep detailed and dynamic inventory tracking of our products on sale. The information of our shopping customers, store employees and detailed information about our products are kept in detail in our database. Thanks to this system we have created, product tracking, customer, employee and store information are kept in detail. We built this system on three general structures: The database consists of backend and frontend parts. Technologies we generally use while creating this system: NodeJS, ReactJS, Axios, PrimeReact, TailwindCSS, NodeJS, ExpressJS, MySql**

Keywords— Beyaz eşya, javascript, müşteri, çalışan, veri tabanı, tablo, çalışan, Nodejs, back-end, front-end, diyagram,Reactjs, Nodejs,CSS

Giriş

Geliştirdiğimiz beyaz eşya bayisi otomasyon sistemini kullanarak satıştaki ürünlerimizin, stok takibini ayrıntılı ve dinamik bir şekilde tutmamıza yarar. alış veriş yapan müşterilerimizin ,mağaza çalışanlarımızın bilgileri ve ürünlerimiz ile ilgili detaylı bilgiler oluşturduğumuz veri tabanımızda detaylı olarak tutulur. Oluşturduğumuz bu sistem sayesinde ürün takibini,müşteri, çalışan ve mağaza bilgileri detaylı olarak tutulur. Bu sistemi de üç genel yapı üzerine kurduk: Veri tabanı, backend, frontend kısımlarından oluşuyor. Bu sistemi oluştururken genel olarak kullandığımız teknolojiler:NodeJS, ReactJS, Axios, PrimeReact, TailwindCSS, NodeJS, ExpressJS, MySQL olarak sıralanabilir.

# Problemin Tanımı

## Problem

Beyaz eşya bayisinde satışı yapılan, stok olarak bulunan ürünlerin, çalışanların ve müşterilerin bilgilerini kolay ve düzenli bir şekilde takip etmek.

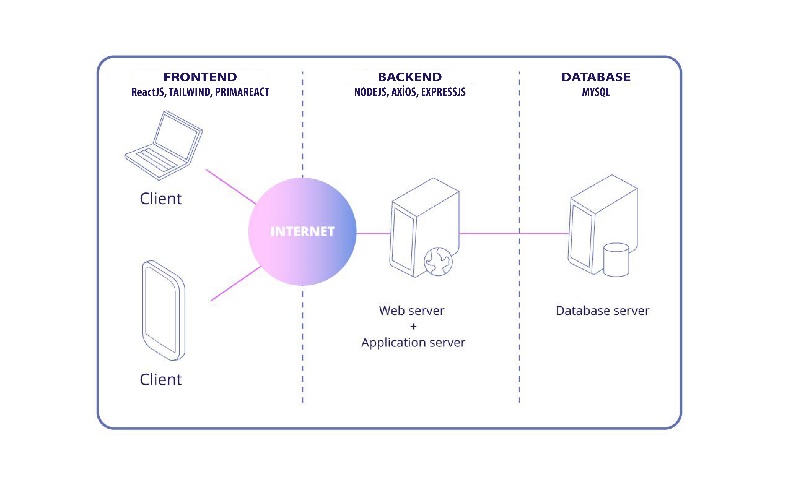
## Yapılan Araştırmalar ve Kullandığımız Teknolojiler

Projemizi geliştirirken ilk olarak kullanacağımız teknolojileri belirlemekte zorlandık. Otomasyon sistemlerinde genel olarak kullanılan teknolojileri araştırdık. Projemizin veri tabanı kısmı için Mysql Workbench kullandık. Projenin dinamik olarak çalışabilmesi için de veri tabanından çektiğimiz verilerin projenin ön yüzünde kullanılması gerekir. Projenin ön yüzüne veri çekebilmek için durağan bir yapı yaratan HTML, CSS gibi yapılar yerine tüm çalışma ortamımızı Javascript üzerine kurmak istedik ve bu ortama uygun teknolojileri araştırdık. Bu ortamda kullanmamıza uygun olan teknolojilerin ise proje ön yüzümüz için ReactJS ön yüz tasarımımız için PrimaReact ve TailwindCSS kullandık. Javascript ortamına kolayca uyarlayabileceğimiz ve proje dinamikliğini sağlamak için bu teknolojileri tercih ettik. Projenin dinamik ortamda kullanılması için http isteğinin gönderilmesi gerekiyordu. Bunu sağlamak için araştırma yaptığımızda Axios kullanmaya karar verdik. Proje ortamımız olan Javascript’i Backend kısmında kullanmak için ise Nodejs kullandık. Veri tabanındaki verileri okuyabilmek için Nodejs ile uyumlu olan bir yapıya ihtiyacımız vardı. Bunun için de Express Js kullandık.

.

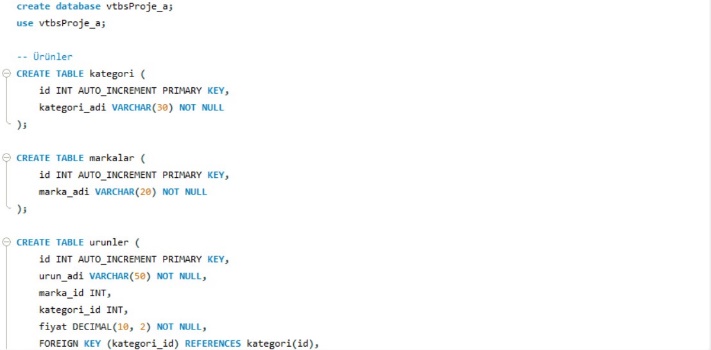
.

# Akış Şeması



## Yazılım Mimarisi

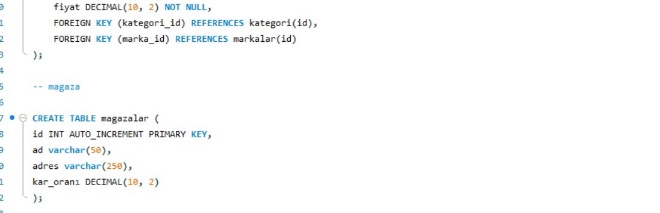
Temel olarak projemiz veri tabanı, backend, frontend kısımlarından oluşmaktadır. Bu kısımları da kendi içinde özelleşmektedir.

Projemizi yaparken öncelikle veri tabanı yapısını oluşturduk. Mysql önce ER diyagramını daha sonrasında da veri tabanı yapısı oluşturduk ve ihtiyacımız olan tabloları oluşturduk ve tablolara veri girişi sağladık. Bunu sağlamak için yazdığımız kodlar şu şekildedir:

*Şekil 1*

Bu kısımda ilk olarak ihtiyacımız olan tabloları oluşturmaya başladık. Tabloları oluşturmaya veri tabanının en dış katmanında yapmaya başladık. Bu sayede tablolarımızın ilişkiselinkileri doğrultusunda oluşturduğumuz da veri tabanımızda ilgili tabloyu tanımlasın. İlk oluşturduğumuz kategori tablosundaki id bilgisini otomatik artan olarak tanımladık. Kategori adını ise varchar olarak tanımladık. Markalar tablosunu oluşturduk. İd bilgisini otomatik artan olarak tanımladık ve marka adını boş geçilmeyecek şekilde ayarladık. Ürünler tablosunu oluşturduk. Bu kısımda ise id bilgisini otomatik artan olarak tanımladık. Ürünler tablosu oluşturduk. Ürün adını varchar olarak ve boş geçilmeyecek şekilde tanımladık. Marka idbilgisini int olarak tanımladık. Kategori id bilgisi int olarak tanımladık bunları da tablo ile direkt ilişkilendirdik çünkü bu tabloların veri tabanımızda bir başka tablonun özleikleri içerisinde bulunmamaktadır ve fiyat bilgisini decimal olarak tabloda boş geçilemeyecek şekilde tanımladık. Kategori id bilgisini

Foreign Key olarak tanımladık.

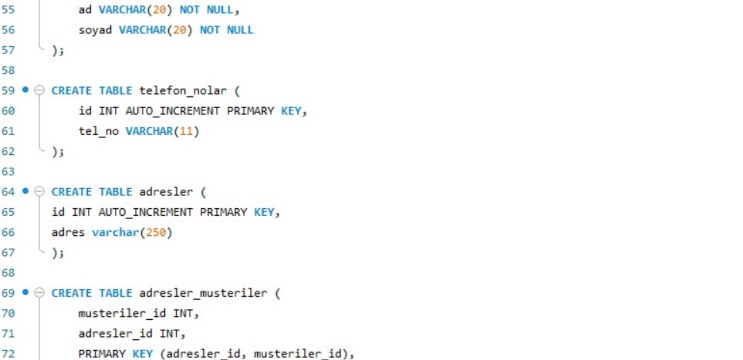
  
*Şekil 2*

Beyaz eşya mağazasının tablosunu oluşturduk. İd bilgisini otomatik artan Primary Key olarak tanımladık. Ad bilgisini ve adres bilgisini varchar olarak tanımladık. Kar oranı bilgisi Decimal olarak tanımladık

## 

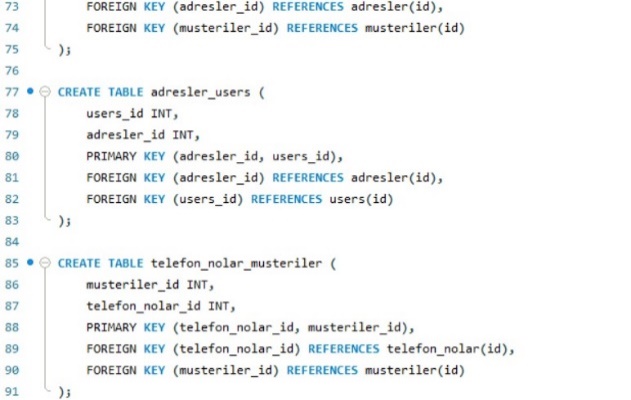
*Şekil 3*

Bu kısımda kullanıcılar, müşteriler ve unvanlar tablosunu oluşturduk. Kullanıcı adının eşsiz olmasını sağladık. Maaş kısmının, kullanıcı adı kısmını boş geçilemeyecek şekilde ayarladık. Id’nin ise Primary Key olarak ayarladık. Açıklama kısmını varchar olarak tanımladık. Username , şifre kısmını da varchar olarak tanımladık.



*Şekil 4*

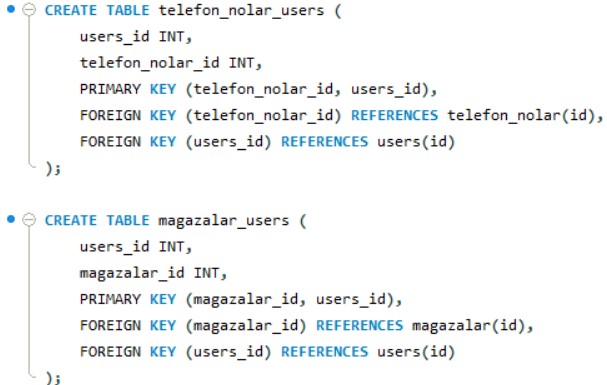
Telefon numaraları, adresler ve müşteri adresleri tablosunu oluşturduk. ID’leri otomatik artan olacak şekilde ayarladık. Telefon numarası bilgisini varchar olarak tanımladık. Adres id bilgisini otomatik artan olacak şekilde primary key olarak tanımladık. Adres bilgisini ise varchar olarak tanımladık.



*Şekil 5*

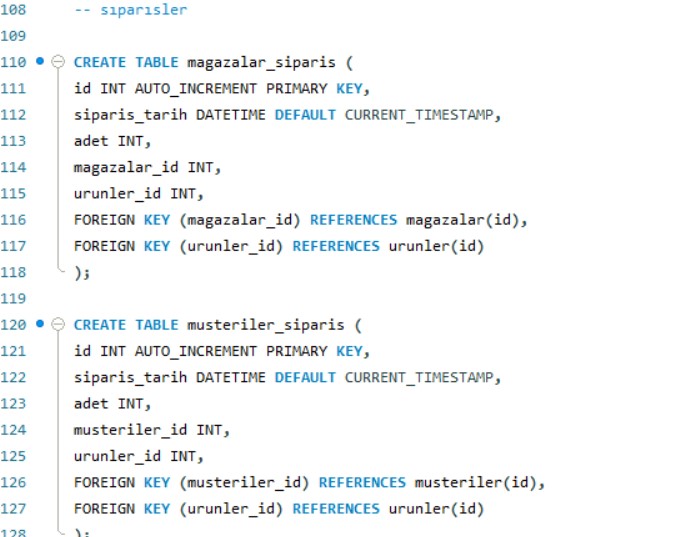
Çalışan adresleri ve müşteri telefon numaraları tablosunu oluşturduk. Telefon numara id ve müşteri id numarasını birincil anahtar olarak ayarladık. Çalışan id ve çalışanların adres id bilgilerini int olarak tanımladık. Telefon numarasıid bilgisi ve müşteri id bilgisi Foreign Key olarak tanımladık.

Bu tabloları oluşturmamızın sebebi bu bilgilerin aynı zamanda userslaında özeliği olmasıdır.



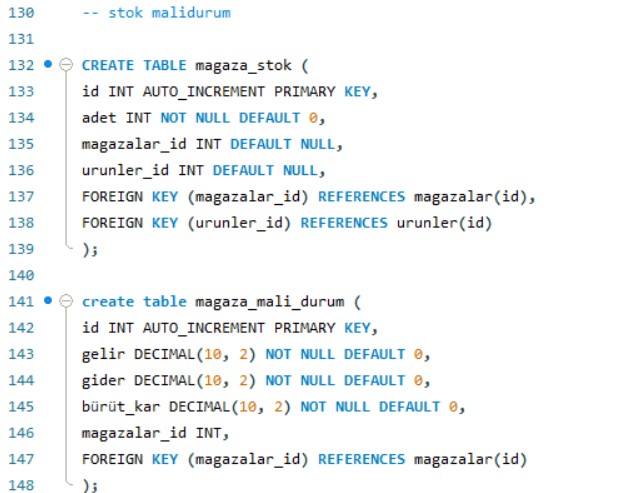
*Şekil 6*

Mağaza çalışanları ile telefon numaralarını ilişkilendirmek için ve mağaza çalışanları için ayrı tablolar oluşturduk. Mağaza id ve çalışanlar için oluşturduğumuz kullanıcı id bilgisini birincil anahtar olarak tanımladık. Çalışanlar için oluşturduğumuz kullanıcı tablosunda mağazalar id ve kullanıcı id niteliklerini Foreign Key olarak tanımladık. Telefon numaraları için oluşturduğumuz tabloda ise telefon numarası id ve kullanıcı id bilgilerini Foreıgn Key olarak tanımlayarak aralarında ilişki kurulmasını sağladık.

. 

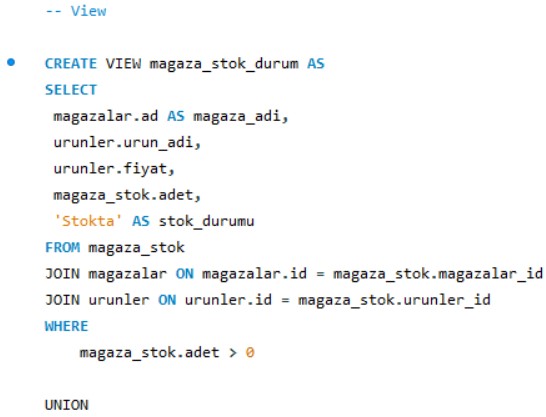
*Şekil 7*

Bu kısımda sipariş bilgilerini tutacağımız tabloları oluşturduk. Id bilgisini otomatik artacak şekilde Primary Key olarak tanımladık. Sipariş edilen tarih bilgisini ise datetime olarak tuttuk.Mağaza id, sipariş adedi bilgisini int olarak tuttuk.Mağaza id ve ürünler id kısmını ise Foreıgn Key olarak tuttuk. Bir diğer tablo ise Müşterilerin sipariş ettiği ürünler için oluşturduğumuz tablo. Bu tabloda ise müşteri id, ürünler id bilgilerini int olarak tuttuk. Müşteri id, ürünler id bilgilerini de bu tablo için Foreıgn Key olarak tanımladık.



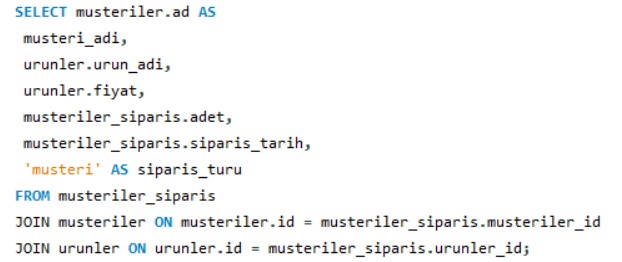
*Şekil 8*

Bu oluşturduğumuz tablolarda ise stokların mali durumunu takip etmek istedik. İlk tablomuz olan mağaza stok tablosunda id bilgisini otomatik artacak şekilde tanımladık, adet bilgisini boş geçilemeyecek şekilde tanımladık triggerlar mantıksal operasyonları yaparken ilk işlem sırasında hata vermesinin önüne geçtik. Ürünler id kısmın da aynı şekilde boş geçilemiyor. Mağaza mali durum tablosunda ise id bilgisi otomatik artan olarak tanımlanmıştır. Gelir,gider ,brüt kar bilgisi boş geçilemeyecek şekilde tanımlanır ve 0 olakalarak tanımlanır ilk değer triggerlar mantıksal operasyonları yaparken ilk işlem sırasında hata vermesinin önüne geçtik. Mağaza id bilgisi int olarak tanımladık. Mağaza id bilgisi Foreıgn Key olarak tanımlanır ve bu şekilde tablolar arasında ilişki kurulur.



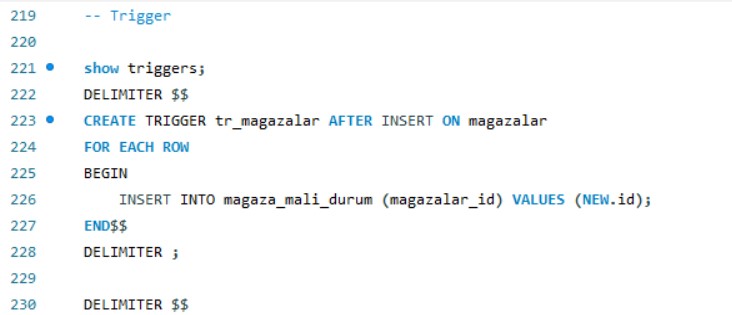
*Şekil 9*

Tablolar arasında daha kolay iletişim kurabilmek için view yapısı kullandık. Ürünler id yapısı ile mağaza stok id yapısını ilişkilendirdik. Mağazaların id numarası ile mağaza stok id numaralarını ilişkilendirdik. Bu ilişkilendirmeye mağazada tutulan stokların adedinin sıfırdan büyük olma koşulunu ilave ettik.



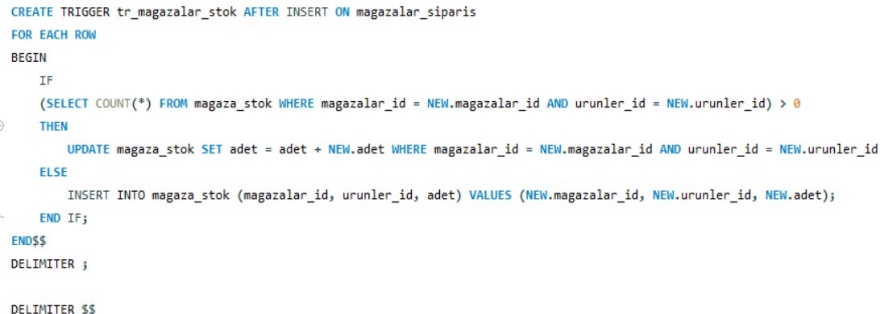
*Şekil 9*

Kodun bu kısmında çekmek istediğimiz verinin başlığını AS yapısı ile değiştirdik. Müşterilerin sipariş tablolarındaki id numaraları ile Müşterilerin sipariş tablolarındaki id numarası ile müşteriler tablosundaki id numara yapısı arasında ilişki kurduk. Sipariş tablosundaki ürün id numarası ile ürünler tablosundaki ürün id numarası arasında ilişki kurduk.



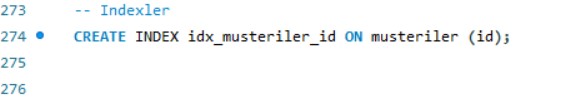
*Şekil 10*

Tablomuzda yapmak istediğimiz ekleme işlemi yaptığımızda mağaza mali durum tablosunda da da bu işlemin senkronize şekilde gerçekleşmesi için trigger tasarladık.



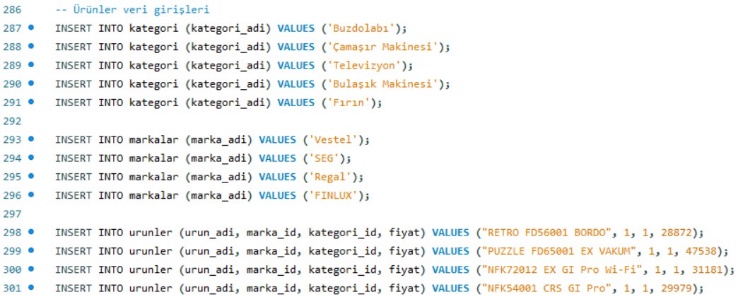
*Şekil 11*

Tasarladığımız bir diğer trigger ise mağazalardaki stok ve sipariş bilgisi için tasarladık. Sipariş bilgisi girildikten sonraki durum koşula bağlanmıştır. Ürün adedi arttıkça tablolarda da eş zamanlı olarak güncelleme yapılacak aksi durumda ise mağaza id bilgisi ürünlerin id bilgisi ve adet bilgisi getirilmesi üzerine koşullanmıştır.



*Şekil 12*

Kodumuzun bu kısmında Müşteri bilgilerinin id numaralarına göre öncelikli sıralayabilmek için index tasarımı yaptık.



*Şekil 13*

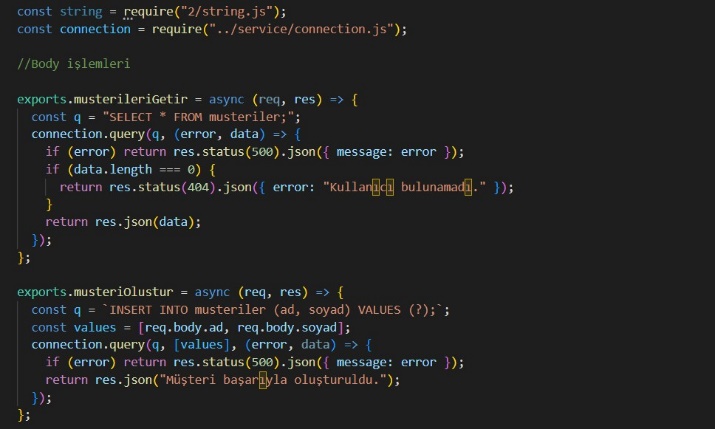
Kodun bu kısmında tablolarımıza veri girmeye başladık. Yukarı kısmında ürünler için yaptığımız veri girişi görülmektedir.

**BackEnd Kısmı**

## Controllerlar

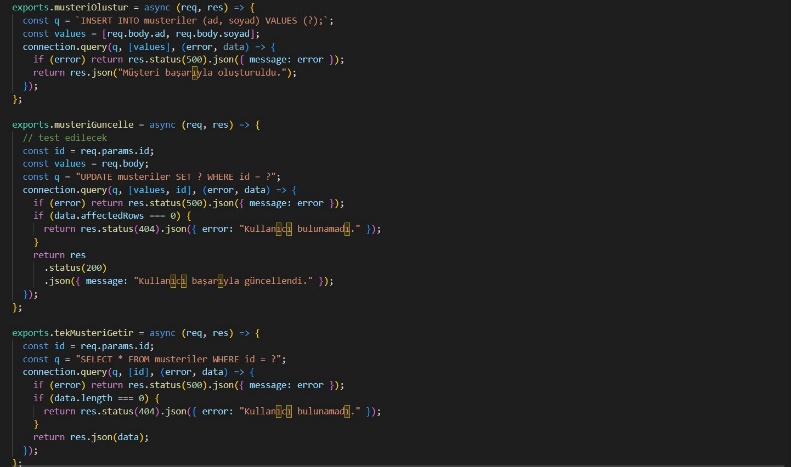
**Müşteri Controller Yapısı**

Projemizde ekleme, silme ve güncelleme yapabilmek için tanımladığımız bir yapıdır. Müşteri Controller yapımızda biz müşterilerimiz ile ilgili ekleme silme ve güncelleme işlemleri yapmayı amaçladık.



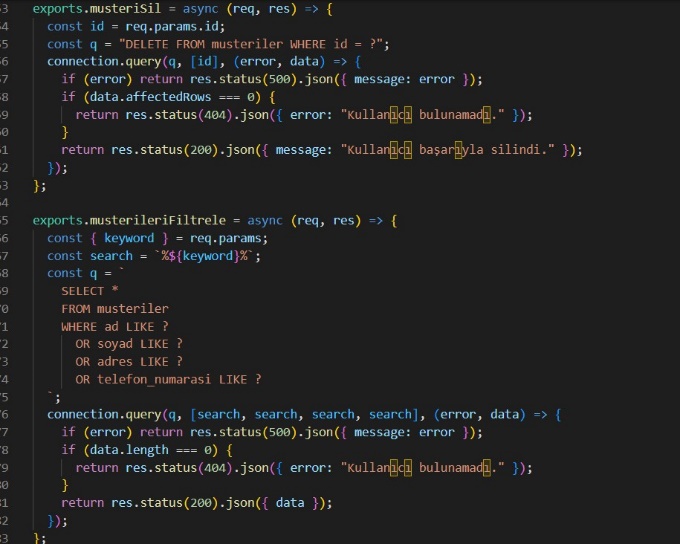
*Şekil 15*

Kodumuzun bu kısmında tablolarımızdan aldığımız SQL sorgularını kullanarak uygun verileri oluşturmamızı, eklememizi ve güncellememizi sağlar.



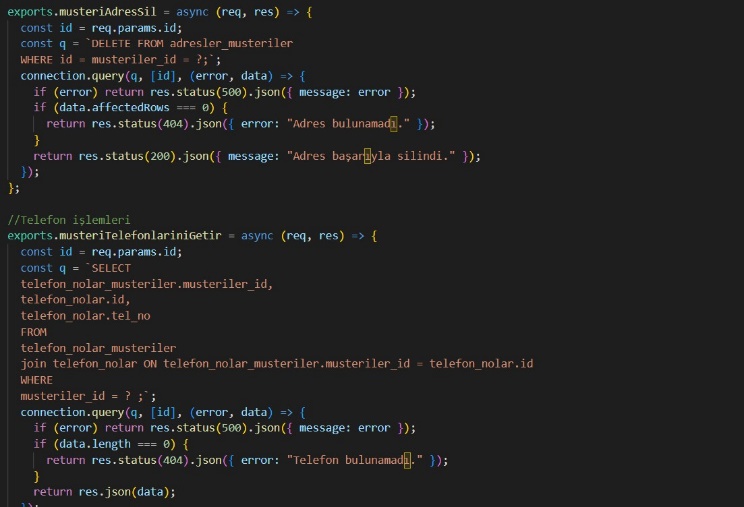
*Şekil 16*

Kodumuzun bu kısmında sorgular sırasında tablodan bilgi güncellemesi yapmaya çalışıyoruz aynı zamanda veri girişi olmadığı takdirde ise hata almamızı sağlar bu kısmı. Aynı zamanda tanımladığımız fonksiyonu asenkron tanımlayarak aynı zaman diliminde farklı fonksiyonların da çalışmasını sağladık.



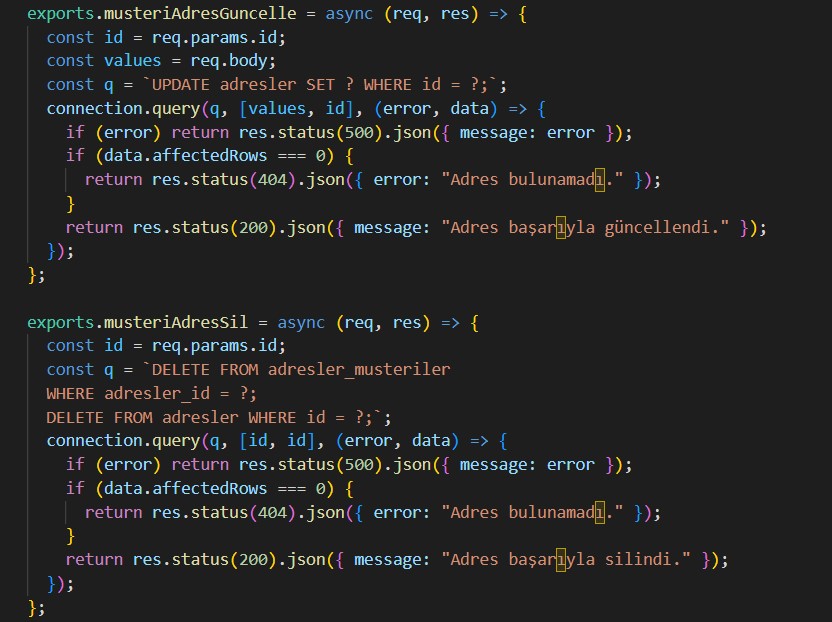
*Şekil 17*

Kodun bu kısmında müşetiSil adlı asenkron fonksiyonumuzu tanımlayarak sunucuya bir http isteği gönderdik.Aldığı isteklerle req parametresi olarak aldığı isteklerle veritabanından id parametresi alır ve bu id ile eşleşen müşteri bilgisini siler. Silme işlemi başarılı olursa istemciye 200 OK yanıtı gönderir. Kullanıcıyı bulamazsa sunucuda 404 hata kodu döner. İkinci fonksiyonumuzda ise musterileriFiltrele adı verilen bir asenkron fonksiyondur. Bu fonksiyon, HTTP istekleri alır Veri tabanımıza bağlantı kurarak "musteriler" tablosundan müşterileri arar. Fonksiyon, "req" parametresi olarak gelen HTTP isteğinin "keyword" parametresini alır ve bu anahtar kelimeye göre veritabanında müşterileri arar. Fonksiyon, müşterilerin "ad", "soyad", "adres" veya "telefon\_numarasi" sütunlarından herhangi biri "keyword" parametresini içeriyorsa, müşteriyi döndürür. Bulunan müşteriler "data" nesnesi içinde JSON formatında istemciye gönderilir. Eğer müşteri bulunamazsa, istemciye bir "404 Not Found" yanıtı gönderilir.



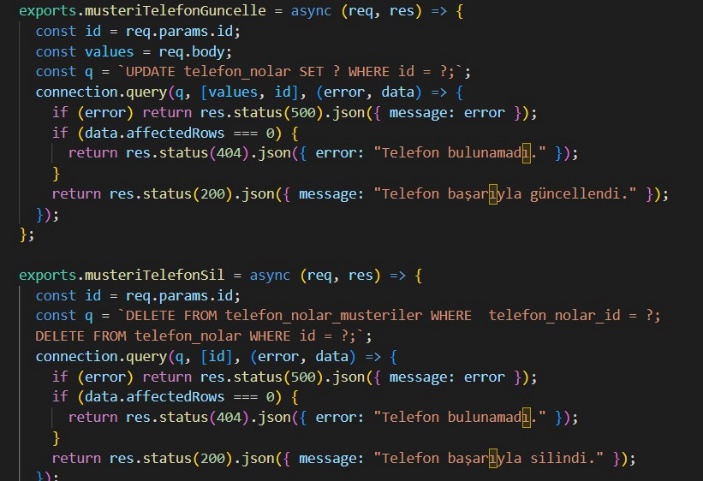
*Şekil 18*

İlk fonksiyonumuz müşetri adres bilgisini silmektedir. Adresler müşteriler tablosunda ilgili adres alanı adresler id alanına gör silinmekte işlemler başarılı olursa 200 kodu döner adres başarıyla silindi mesajı verilir. Aksi durumda 404 hata kodu döner. İkinci fonksiyonumuz olan müşteri telefonlarını getirmeye yarar. Telefon\_nolar\_müşteriler ve telefon\_nolar tablolarından müşteriler\_id ve telefon\_nolar\_id alanlarına göre join edilerek yürütülmektedir. İşlem başarılı olursa veriler json formatında döndürür.



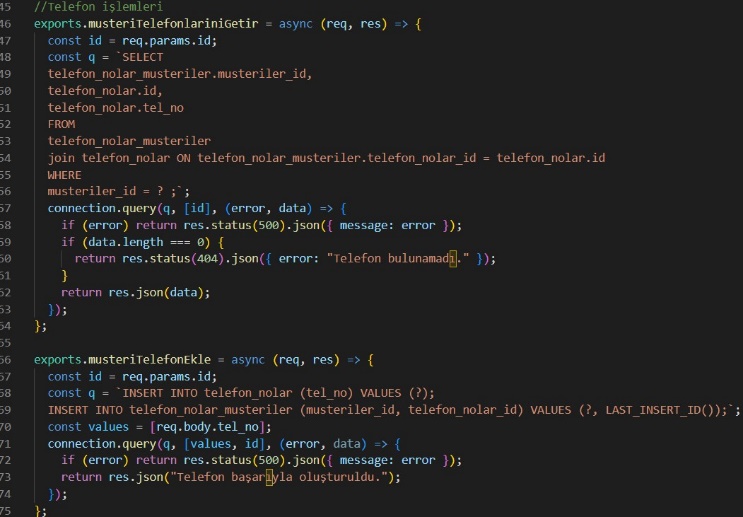
*Şekil 19*

Kodumuzun bu kısmında müşteri adresleriyle CRUD işlemleri gerçekleştirmek istedik.Müşteri adresi güncellemek için kullanılır ilk fonksiyonumuz. http PUT isteği gönderdiğinde id parametresiyle belirtilen adres değeri istek gövdesindeki values tarafından değiştirilir. Ardından adresler tablosunda ilgili satırın güncellenmesi için UPDATE sorgusu çalıştırılır ve http yanıt döndürür. Güncellenen satır yoksa 404 hatası döndürülür. İkinci fonksiyonumuz ise hem adresler\_müşteriler tablosundan hem de adreslerden bilgilerin silinmesi için kullanılır. DELETE sorgusu ile adresler\_müşteriler tablosundan ilgili satır silinir ardından aynı adresler tablosundaki satırı da siler.Silinen satır yoksa 404 hatası döner.



*Şekil 20*

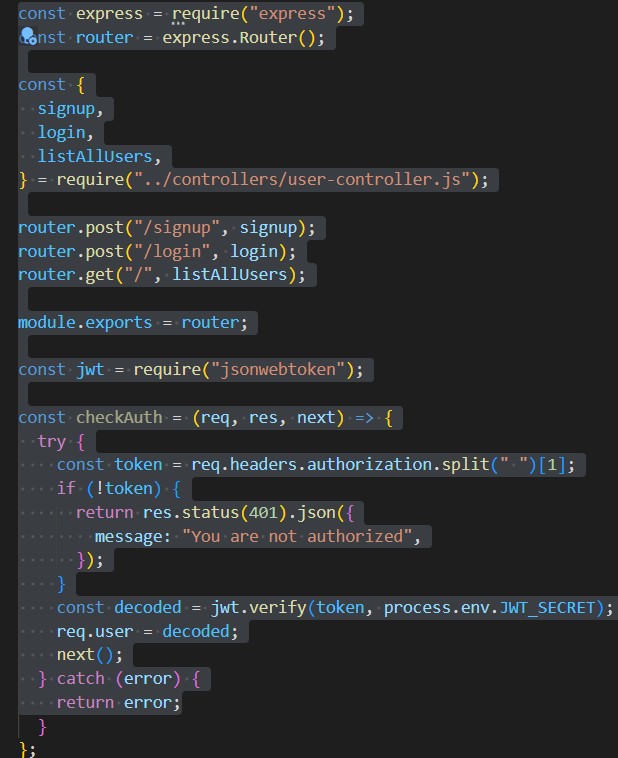
Kodun bu kısmında telefon numaralarını güncelleme ve silme işlemi yapılmaktadır. İlk olarak id değişkeni aracılığı ile hangi verilerin güncelleneceği belirlenir. Sonra güncellenen veriler values değişkeni aracılığı ile alınır. Bu veriler telefon\_nolar tablosunda ilgili id numarasına sahip satırın verileri ile güncellenir. Başarısız işlem olursa 404 hatası döndürülür. İkinci fonksiyonumuzda belli tekefon numarası ve ona bağlı veriler silinir. İlk olarak id ile hangi verilerin silineceği belirlenir. Daha sonra telefon nolar müşteriler tablosundan telefon numarası ile ilişkili tüm veriler silinir. Son olarak belli telefon nolar tablosundan belli telefon numarası silinir. İşlem başarısız olursa hata döner



*Şekil 21*

İlk fonksiyonumuz belli bir müşterinin tüm telefon numaralarını veri tabanından çeker.Ve bir JSON nesnesi olarak döndürür. Bu işlemler telefon\_nolar tablosunda depolandığı ve müşteri telefon numaraları arasındaki ilişkinin telefon\_nolar\_müşteriler tablosunda depolandığı bir SQL sorgusu gerçekleştirir. İkinci fonksiyon ise telefon\_nolar tablosunda yeni kayıt ekleyerek telefon\_nolar\_müşteriler tablosuna müşteri ve yeni telefon numarası ekleyerek gerçekleştirilir.Yeni telefon numarası http isteğinin gövdesindeki tel\_no alanından alınır. Her iki sorgu sonucu http yanıtı döner.

**Log-Auth**

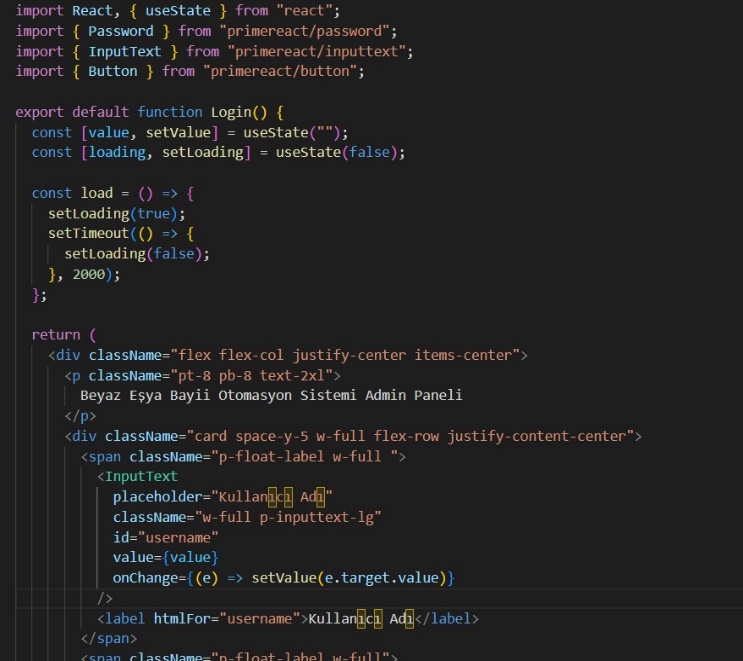


*Şekil 22*

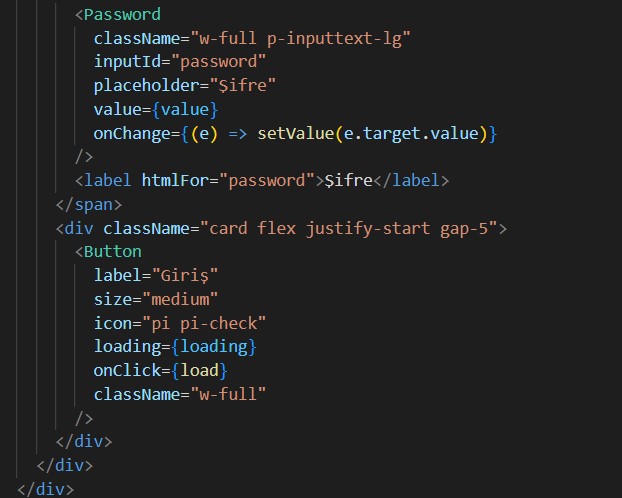
Bu fonksiyon bir http isteği aldığında kimlik doğrulaması yapmak için kullanılacak JWT öğesini kontrol eder. Fonksiyon isteğin Authhorization başlığındaki JWT öğesini alır ve bunu doğrular bu doğrulama başarılı olursa sonraki next() fonksiyonunu çalıştırır. JWT çözümlenmiş sürümünü req.user niteliğine yerleştirir. Doğrulama başarısız olursa 401 hatası döner.

**Front End Kısmı**

Front End kısmımızı oluştururken öncelikle login paneli kısmından başladık. Bu kısmı oluştururken ReactJs ve Javascript kullandık stil sayfalarımızı oluştururken de tailwindcssden yardım aldık.



*Şekil 23*



*Şekil 24*

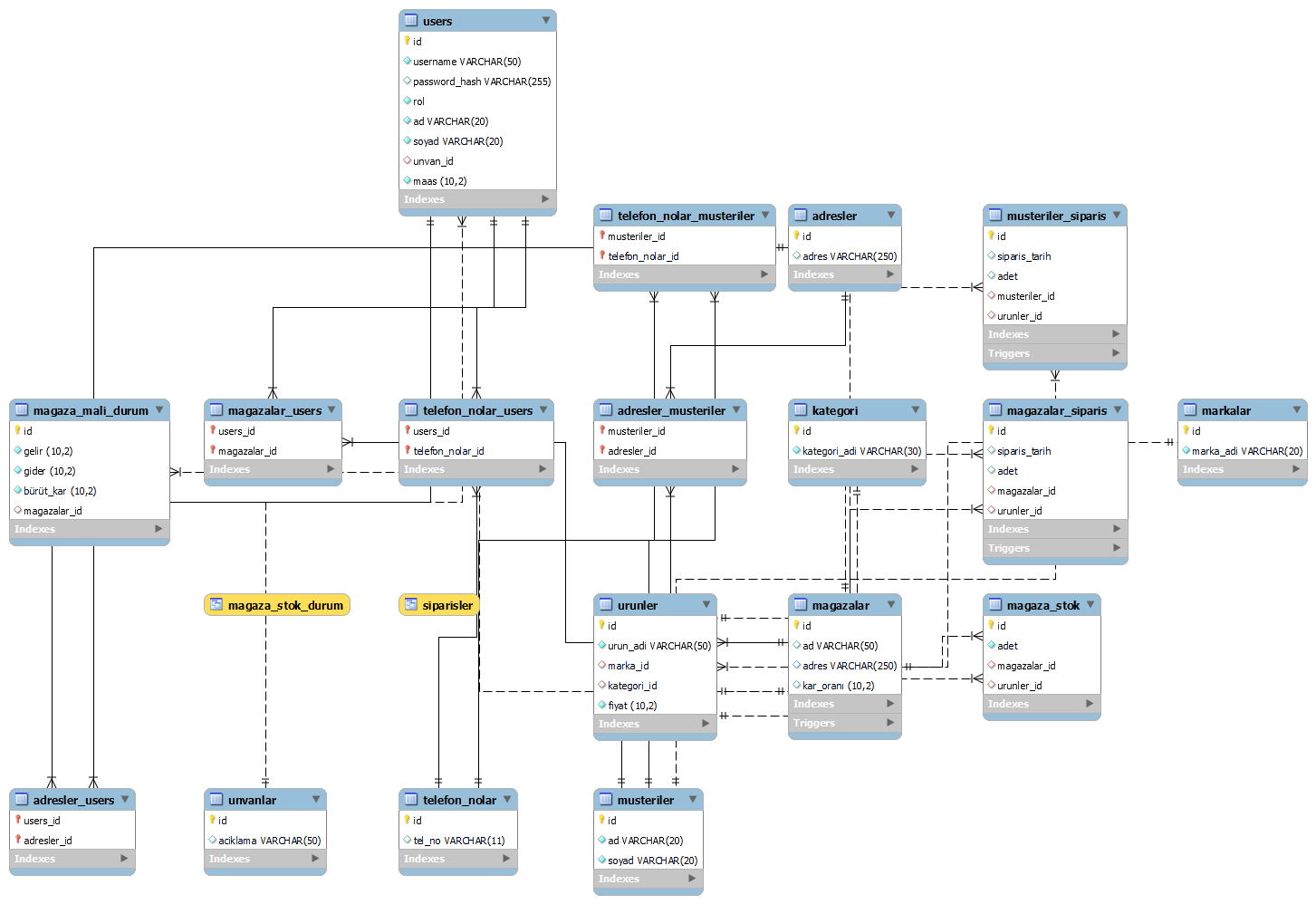
Kodumuzun bu bölümünde bir kullanıcı adı ve giriş sayfasından oluşan bir login paneli tasarlamak istedik. Sayfada bir giriş formu yer alır ve kullanıcı adı ve şifre bilgisi girilir. Bu formda primareact kütüphanesini kullandık. Bu kütüphaneden Password ve Input Text bileşenlerini kullandık. Formda bir de giriş butonu vardır. Bu butona tıklandığında kullanıcının giriş bilgilerinin doğruluğu kontrol edilir. Ve bir http isteği yapılır. Asenkron bir işlem gerçekleşmez Bunun yerine setLoading fonksiyonu çalıştırılır ve iki saniye boyunca buton yükleniyor olarak gösterilir. İşlem bittikten sonra fonksiyon tekrar çağrılır ve yükleniyor olmaktan çıkar.



*Şekil 25*

Kodun bu kısmında “pages ” kısmında “Login.jsx” dosyasında login içeren bir dosya oluşturulur. App bileşeni “h-screen” sınıfını kullanarak ekranın tamamını kaplayan bir öge oluşmasını sağlar. Flexbox ile yatay ve dikey olarak ortalanmış içerikleri içerir. Kod çalıştırıldığında kullanıcının adı ve şifresini girmesi için form sunar.

**Veri Tabanı Diyagramı**



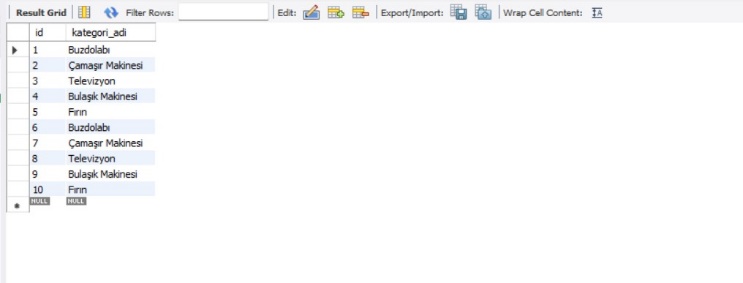
*Şekil 10*

**Genel Yapı**

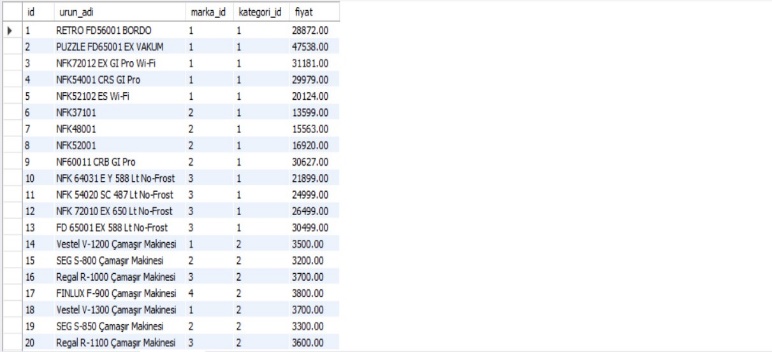
Projemiz genel olarak backend,frontend ve veri tabanı kısmından oluşmaktadır. Veri tabanı kısmında müşterilerimize, mağaza çalışanlarımıza, ürünlerimize, elimizde bulunan ürün stoklarımıza ve satış yaptığımız mağazalara ait bilgileri detaylı olarak farklı tablolarda tutuyoruz. Sistemdeki genel yapıya göre sistemde ürünlerimizin marka bilgisi, kategori bilgisi ve fiyat bilgisi bir tabloda kategori ve marka bilgisi ayrı tablolarda tutulmaktadır. Ürünlerin satıldığı mağazaların ise adı, adresi ve kar oranı ayrı bir tablo olarak mağazadaki stok bilgilerinin daha kolay takip edilebilmesi için mağaza stok bilgisi için adet mağaza id ve ürün id ile farklı bir tablo oluşturduk. Mağazadan alışveriş yapan müşterilerin bilgilerini daha düzenli tutabilmek için müşterilerin ad, soyad ve id bilgisini tuttuk. Telefon numaraları adresleri için ayrı tablolar belirledik. Mağazaların mali durumunu takip edebilmek için gelir gider bilgisi brüt kar ve mağaza id için ayrı alan tanımladık. Mağaza çalışanları bilgileri detaylı olarak tutabilmek için ayrı bir tablo oluşturduk. Mağaza çalışanlarını unvanlarına göre ayırdık. Çalışanların adreslerini ve telefonlarını da ayrı bir tabloda tuttuk.



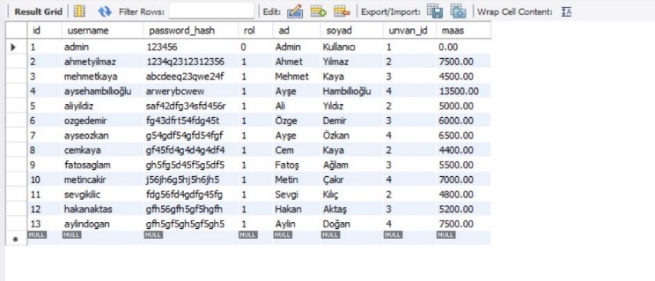
*Şekil 11*

**

*Şekil 12*

**

*Şekil 13*

**

*Şekil 14*

Bundan sonraki kısım projenin backend kısmı. Veri tabanı ile bağlantımızı controllerlar aracılığı ile sağlıyoruz. Müşteri bilgileri, çalışan bilgileri kullanıcı bilgilerine istek yollayabilmek için ayrı ayrı fonksiyonlar oluşturduk. Ve veri tabanı ile bağlantı kurulmasını sağlayarak istediğimiz işlemleri yapmaya çalıştık. Tanımladığımız fonksiyonları asenkron olarak tanımladık bu şekilde kurduğumuz yapıya aynı anda farklı işlemler yapabilme özelliği kazandırmış olduk. Oluşturduğumuz Log Auth yapısı ile kimlik doğrulaması yapılmasını sağladık.

##### Kaynakça

http://www.baskent.edu.tr/~gmemis/courses/introductiontodatabase/Db10.pdf

<https://dev.mysql.com/doc/refman/8.0/en/dynindex-statement.html>

<https://dev.mysql.com/doc/refman/8.0/en/view-syntax.html>

https://dev.mysql.com/doc/refman/8.0/en/trigger-syntax.html