KOCAELİ ÜNİVERSİTESİ

TEKNOLOJİ FAKÜLTESİ

BİLİŞİM SİSTEMLERİ MÜHENDİSLİĞİ

PROJE

OTOGALERİ OTOMASYON PROGRAMI

Selim Erdem ÇAMLIOĞLU

Tunç Berkim KÜRTÜNCÜ

Mert MAHANOĞLU

Boran AKOVA

KOCAELİ 2020

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

GİRİŞ

Gerçekleştirilen proje çalışması kapsamında bir araba galeri zincirinin belirlenen senaryolar dahilinde, müşterilerine hatasız ve kesintisiz hizmet verebileceği bir otomasyon programı üzerinde çalışılmıştır. Bu çalışma müşterilerin mobil ve masaüstü cihazlarından araç alım, satım ve arabalarını galeriye satma işlemlerini gerçekleştirebilmeleri göz önüne alınarak gerçekleştirilmiştir. Aynı zamanda masaüstü arayüzü galeride çalışan personeller ve yöneticiler içinde bilgilere özel olarak erişim sağlayabilmektedir. Problemin galeri zincirini tek bir yerden tüm müşteriler ve personel için nasıl yönetilebileceği olduğu düşünüldüğünde gerçekleştirilen çalışma testlerden başarı ile çıkmıştır. Eklenen özellikler kapsamında sektörde yer alan galerinin daha iyi hizmet vermesi ve yönetilmesi amaçlanmıştır. Mobil kapsamında geliştirilen arayüz sadece müşterilere hızlı bir hizmet sunmaktadır. Masaüstü arayüz müşterilere ve personele aktif ve etkin hizmet sunabilmektedir. Bu arayüzlerin birbirleri arasında başarılı bir şekilde bağlantı sağlayabilmeleri için sağlam bir sunucu mimarisine ihtiyaç duyulmaktadır. Anlık senkronizasyon ihtiyacı olan bir sistem inşa edilen bu durum için sunucunun da anlık çalışmayı desteklemesi ve arayüzlerin bu açıdan gerekli konfigürasyona sahip olması beklenmektedir. Bu isterler doğrultusunda mobil kullanan müşteriler için Swift ile bir uygulama, masaüstü kullanan personeller ve müşteriler için Java ile ayrı bir uygulama geliştirilmiştir. Uygulamaların veri tabanındaki gerekli yerlere bağlantı gerçekleştirmesi için özel bağlantılar tanımlanmıştır ve başarı ile gerçekleştirilmiştir. [1-3]

1. PROJE AŞAMALARI

Bu bölümde, problemin tanımından ve bu proje çalışması kapsamında yapılan geliştirmelerden bahsedilecektir. Akabinde bu geliştirmeleri gerçek zamanlı (real time) ayakta tutabilmek için kullanılması beklenen sunucu yapısından bahsedilecektir.

1.1. Problem Tanımı

Günümüzde piyasadaki galerilerin en büyük problemlerinden bir tanesi de müşteriye doğru yollarla ulaşamamaktır. Müşteriye ulaşamamak küçük birkaç satışınızı değil müşteri ile aranızdaki bütün bağlantıyı ve dolayısı ile satış rakamlarınız da etkiler. Galerilerin satışları için online komisyon bazlı sistemler kullandığının farkındayız fakat bütün gelirin kendilerine kaldığı bu sistemler galerinin elde etmek istediği farklılığı ve özel imajı yaratamamaktadırlar. Bundan dolayı galerilere özel olarak kendi ihtiyaçları doğrultusunda yeterli çalışmayı sağlayabilecek bir proje çalışması gerçekleştirdik. Bu çalışmada müşterilerin mobil ve masaüstü uygulamamızdan ulaşabildiği yöneticilerin ve personellerin ise masaüstündeki aynı uygulamadan güvenli bir şekilde satışları izleyebildiği bir çalışma gerçekleştirdik. Bu çalışma odaklanmış olduğumuz galerilerin özel bir yönetim sistemlerinin bulunmaması sorununa çözüm üretmiş oldu.

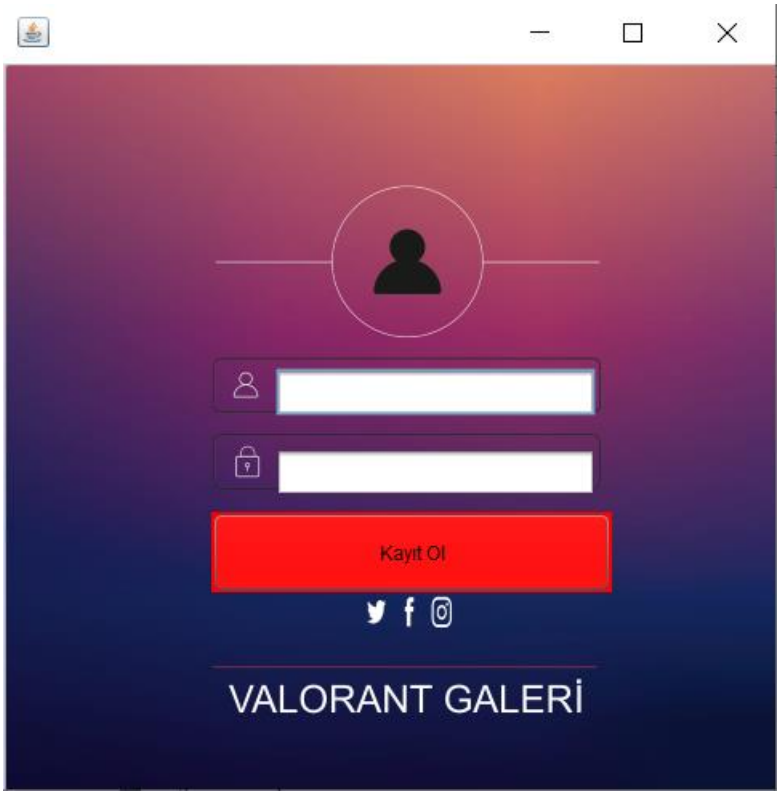
1.2. Java ile Masaüstü için Geliştirilen Uygulama

Gerçekleştirilen proje çalışması kapsamında Java Swing altyapısı kullanılarak bir masaüstü uygulaması geliştirilmiştir. Bu uygulama ile kullanıcı ve yöneticiler galeri detaylarına erişerek işlemler gerçekleştirebilmektedirler. Masaüstü kullanıcısı olan çalışanlar ve müşteriler için gerçekleştirilmiş olan uygulamadır. Özellikleri dahilinde yönetici(admin) paneline sahip olması ve müşterilerin bütün işlemlerini gerçekleştirebileceği bir uygulama olması açısından masaüstünde verimliliği arttırarak müşteri memnuniyetini yükseltebilecek bir uygulama olmuştur. [2]

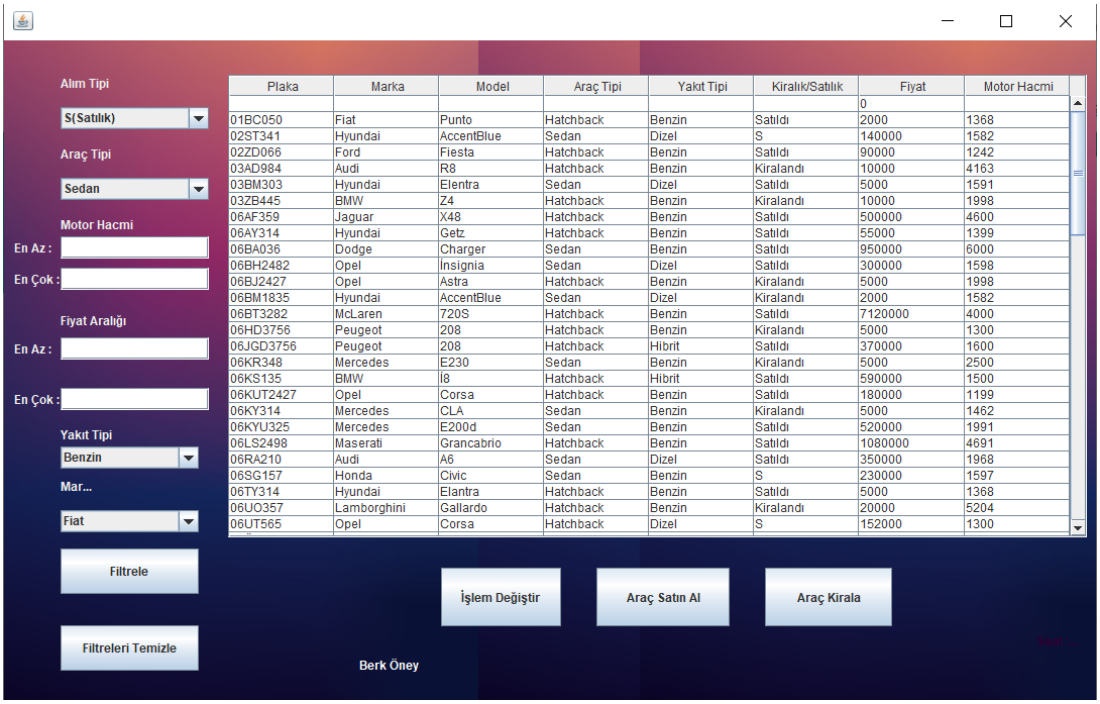
Gerçekleştirilen Galeri Otomasyon projesi kapsamında, derleyici olarak Netbeans 8.2 tercih edilmiş olup, Java Run Environment (JRE) 1.8.0_241 ve Java Development Kit

(JDK) 1.8.0_231 sürümleri tercih edilmiştir. Uygulama geliştirilirken program ile veri tabanını bağlayabilmek için MySQL Connector 8.0.18 kullanılmıştır. Uygulama Java ile masaüstü platformlar için geliştirilmiş olup, bu uygulamada müşterilerin olabilecek en kolay şekilde kayıt olma, araç satın alma veya kiralama, araç satma ve filtreleme gibi işlemleri gerçekleştirmesi sağlanmıştır. Müşteriler dışında veri tabanı ve galerinin kontrollerinin yönetilebileceği bir yönetici paneli tasarlanmıştır. Tasarlanan yönetici panelinin güvenliğinin sağlanması için gerekli kontroller yapılmış olup sadece yetkili kişilerin girmesine imkân sağlanmıştır. Bu yönetici panelinde veri tabanında ki tablolar kolayca listelenebilir, fatura bilgileri değiştirilebilir ve dinamik arama ile anında istenilen sonuçlar elde edilebilir. [2, 4, 5]

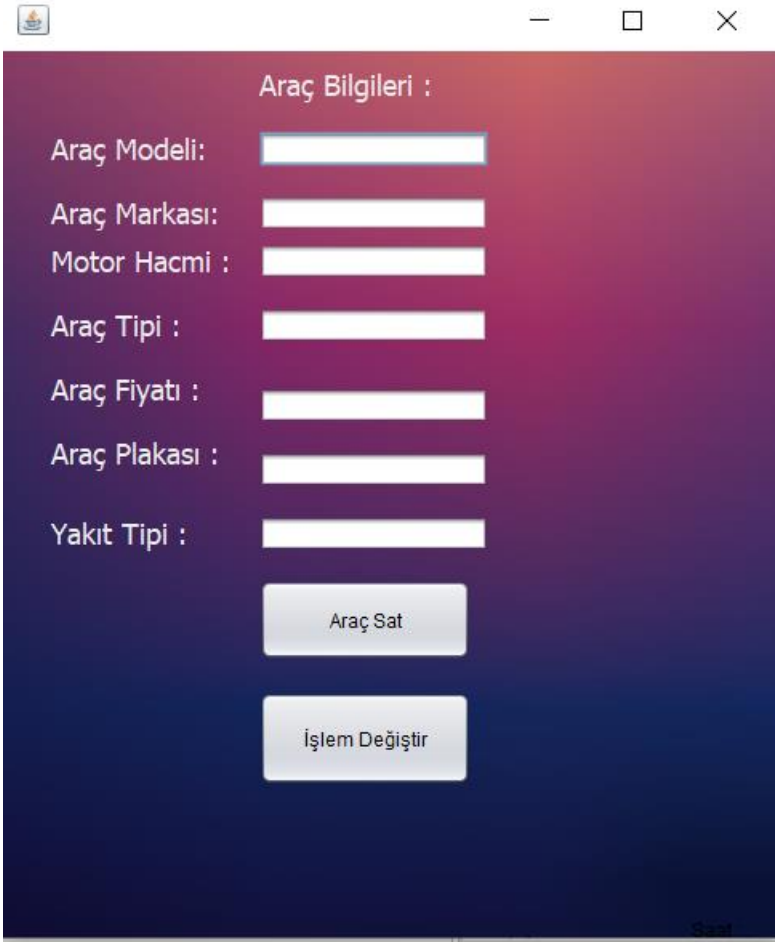
Şekil 1.2.1. Masaüstü uygulamasının müşteri giriş paneli



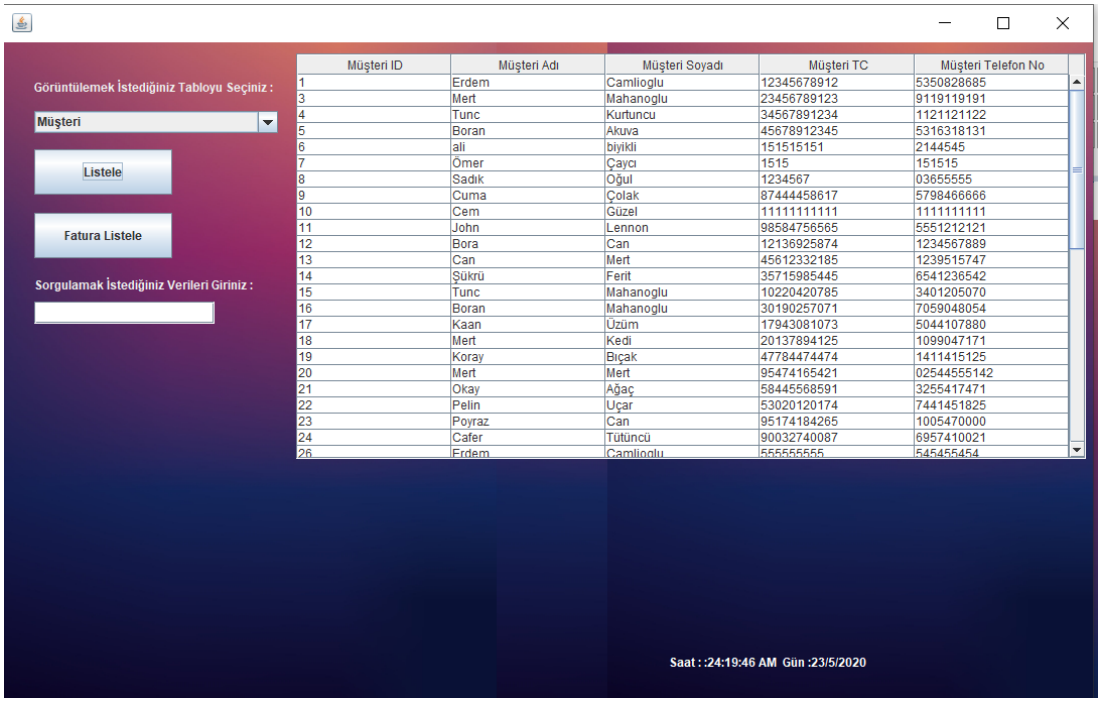
Şekil 1.2.2. Masaüstü uygulamasının müşteri kayıt paneli



Şekil 1.2.3. Masaüstü uygulamasının müşteri için araç alım ve kiralama paneli

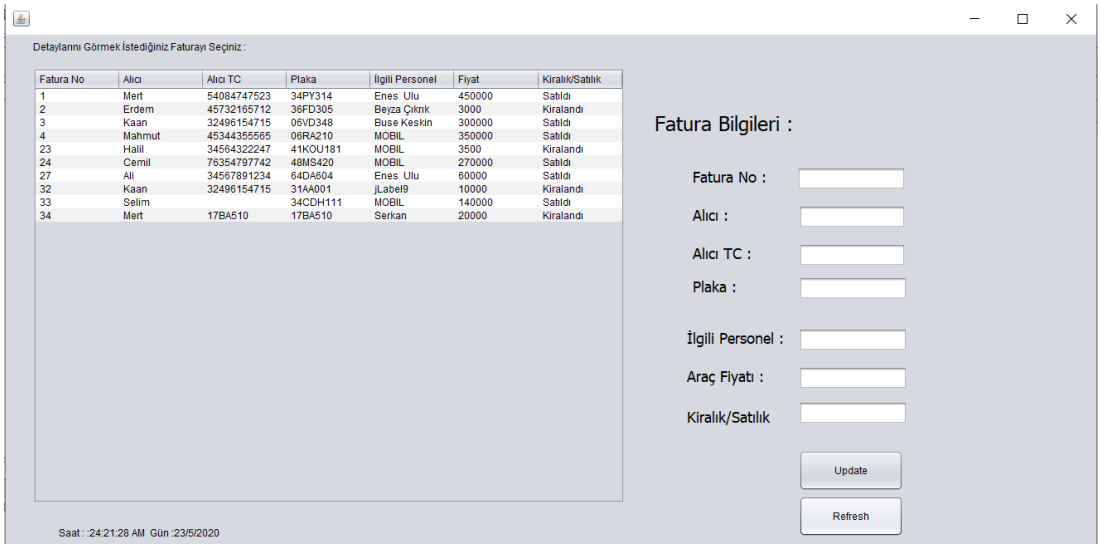


Şekil 1.2.4. Masaüstü uygulamasında müşterinin araç satma paneli



Müşteri ID	Müşteri Adı	Müşteri Soyadı	Müşteri TC	Müşteri Telefon No
1	Erdem	Camlioglu	12345678912	5350828685
3	Mert	Mahanoglu	23456789123	9119119191
4	Tunc	Kurtuncu	34567891234	1121121122
5	Boran	Akuva	45678912345	5316318131
6	ali	biyikli	151515151	2144545
7	Ömer	Çaycı	1515	151515
8	Sadık	Oğul	1234567	03655555
9	Cuma	Çolak	87444458617	5798466666
10	Cem	Güzel	11111111111	1111111111
11	John	Lennon	98584756565	5551212121
12	Bora	Can	12136925874	1234567889
13	Can	Mert	45612332185	1239515747
14	Şükrü	Ferit	35715985445	6541236542
15	Tunc	Mahanoglu	10220420785	3401205070
16	Boran	Mahanoglu	30190257071	7059048054
17	Kaan	Üzüm	17943081073	5044107880
18	Mert	Kedi	20137894125	1099047171
19	Koray	Bıçak	47784474474	1411415125
20	Mert	Mert	9547416542l	02544555142
21	Okay	Ağaç	58445568591	3255417471
22	Pelin	Uçar	53020120174	7441451825
23	Poyraz	Can	95174184265	1005470000
24	Cafer	Tütüncü	90032740087	6957410021
26	Erdem	Camlioglu	555555555	545455454

Şekil 1.2.5. Masaüstü uygulamasında yöneticinin bilgi görüntüleme paneli



Fatura No	Alıcı	Alıcı TC	Plaka	İlgili Personel	Fiyat	Kiralık/Satılık
1	Mert	54084747523	34PY314	Enes Ulu	450000	Satıldı
2	Erdem	45732165712	36FD305	Beyza Çıkrık	3000	Kiralandı
3	Kaan	32496154715	06VD348	Buse Keskin	300000	Satıldı
4	Mahmut	45344355565	06RA210	MOBIL	350000	Satıldı
23	Halil	34564322247	41KOU181	MOBIL	3500	Kiralandı
24	Cemil	76354797742	48MS420	MOBIL	270000	Satıldı
27	Ali	34567891234	64DA604	Enes Ulu	60000	Satıldı
32	Kaan	32496154715	31AA001	jLabel9	10000	Kiralandı
33	Selim		34CDH111	MOBIL	140000	Satıldı
34	Mert	17BA510	17BA510	Serkan	20000	Kiralandı

Şekil 1.2.6. Masaüstü uygulamasında yöneticinin faturaları inceleyeceği panel

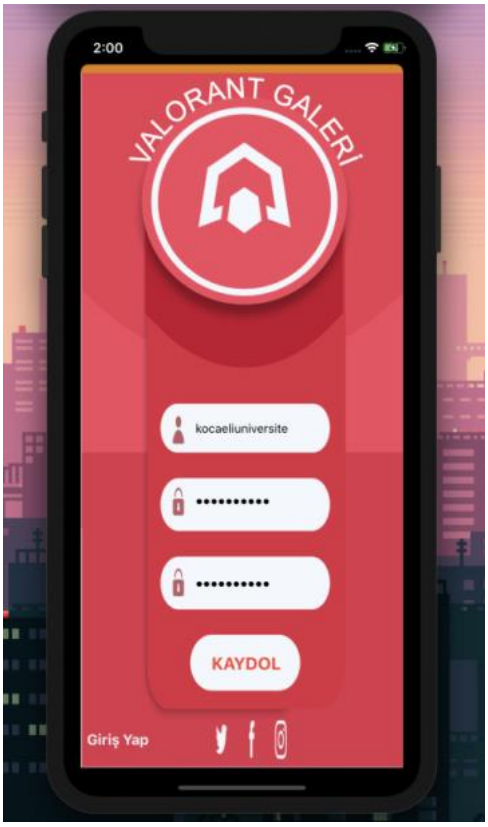
1.3. Swift ile IOS Platformu için Geliştirilen Uygulama

Gerçekleştirilen proje çalışması kapsamında, mobil müşterilerine özel olarak hazırlanan bu arayüz için geliştirme safhasında MacOS Catalina geliştirme ekipmanı ile Xcode 11.1 üzerinde Swift 5.3 dil geliştirmeleri ile IOS uygulaması gerçekleştirilmiştir. Bu geliştirme süresince müşterilerin mağdur olmadan giriş yapabileceği, kayıt olabileceği, stokumuzda bulunan araçları görüp kiralık araçları

kiralayabileceği, satılık araçları satın alabileceği ve kendi aracını galerimize satabileceği bir platform oluşturulmuştur. Platform kapsamında bulunan giriş ekranı Şekil 1.3.1.'de, kayıt ol ekranı Şekil 1.3.2.'de, araçların listelendiği ekran Şekil 1.3.3.'de, araçların müşteriler tarafından satın alınabileceği ve kiralanabileceği ekran Şekil 1.3.4.'de, müşterinin kendi aracını galeriye bilgilerini girerek satacağı ekran ise Şekil 1.3.5.'de belirtilmiştir. [3, 4, 6]

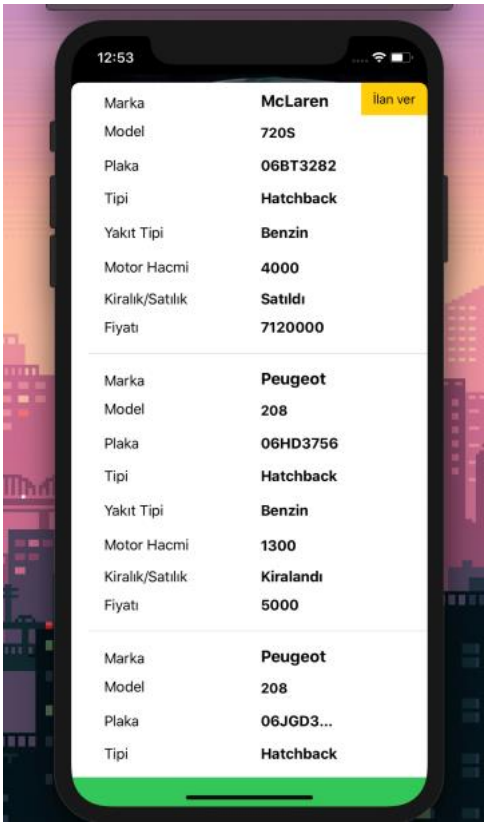
Şekil 1.3.1. IOS uygulaması müşteri giriş ekranı

Müşteri bu ekranda (Şekil 1.3.1.) giriş bilgilerini yani kullanıcı adı ve şifresini kullanarak güvenli bir şekilde sisteme giriş sağlar.



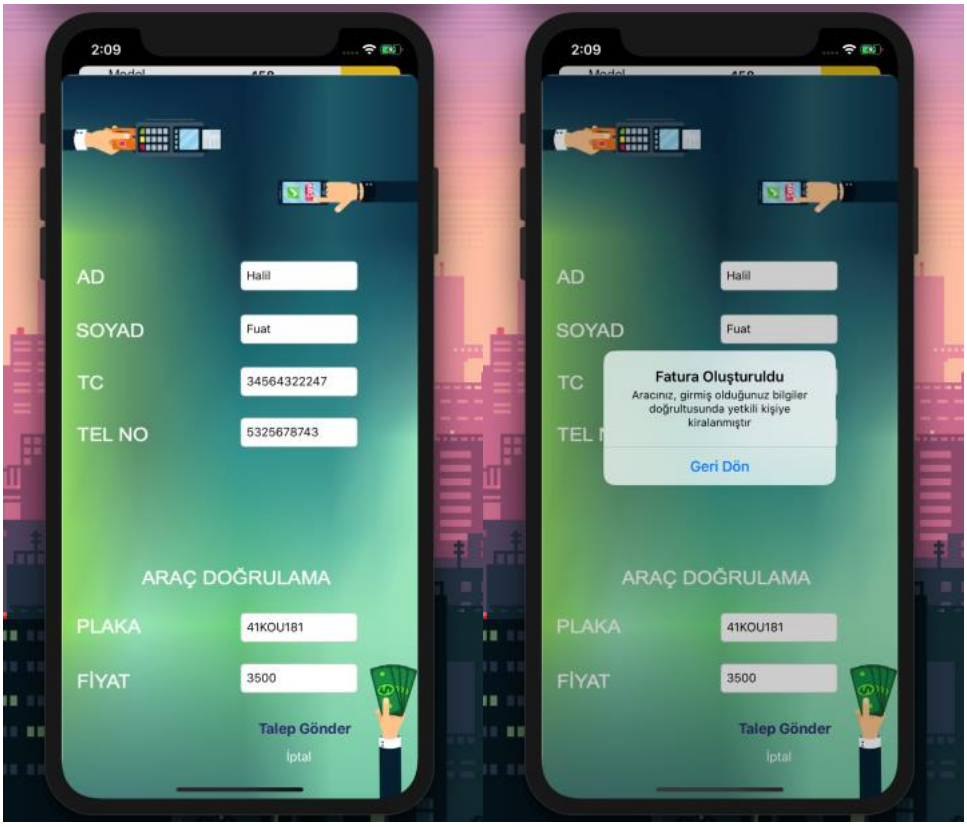
Şekil 1.3.2. IOS uygulaması müşteri kayıt ekranı

Müşteri bu ekranı (Şekil 1.3.2.) kullanarak sisteme kullanıcı adı şifre ve şifre onay bilgileri ile kayıt olmaktadır. Bu aşamada müşterinin girdiği bilgilerin sistemde olan bilgiler ile çakışması veya girdiği şifrelerin birbiri ile uyuşmaması ihtimaline karşı sistemde çift yönlü hata koruma mekanizması vardır.



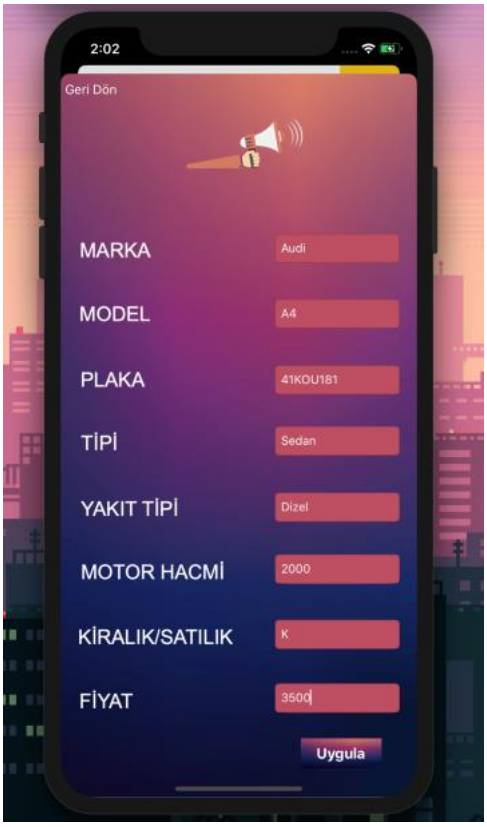
Şekil 1.3.3. IOS uygulaması araç listeleme ekranı

Müşteri bu ekranda (Şekil 1.3.3.) listelenmiş araçları inceleyerek kiralayabileceği araçları veya satın alabileceği araçların listelenmiş hallerini görüp tercihini gerçekleştirip satın alım veya kiralama ekranına gidebilmektedir.



Şekil 1.3.4. IOS uygulaması araç kiralama ve satın alma ekranı

Müşteri bu ekranı (Şekil 1.3.4.) kullanarak seçtiği aracın kiralama işlemini veya satın alma işlemini gerçekleştirir.



Şekil 1.3.5. IOS uygulaması araç satma ekranı

Müşteri bu ekrandaki (Şekil 1.3.5.) boşlukları satmak istediği araç bazında doldurarak aracının satım işlemini gerçekleştirir.

1.4. MySQL Veri Tabanı ve Gerçekleştirilen Bağlantı

Bu denli kapsamlı bir projede verilmiş isterler sebebi ile MySQL kullanılması gerekmiştir. Bu kullanım kapsamında projenin sunucu kısmında MySQL çalıştırılmış ve bağlantılar MySQL komutları ile gerçekleştirilmiştir. Projenin Java ile gerçekleştirilen masaüstü uygulaması kısmında bağlantı her ne kadar dahili konektör ile sağlanmış olsa da Swift ile gerçekleştirilen IOS Native uygulamasında bağlantı MySQL'in çok eski bir mimariye sahip olması nedeni ile direkt olarak gerçekleştirilememiştir. Bu aşamada yetersiz kalan MySQL'in konektör bağlantı eksikliğini PHP ile geliştirdiğimiz kendi RESTFULL API'ımız ile tamamlamış bulunmaktayız. Bu API, IOS uygulamamız için gerekli sorgulamaları ve veri tabanı işlemlerini gerçekleştirebilecek gelişmişliğe sahip olarak tasarlanmış ve işleme alınmıştır. Bu çalışma kapsamında oluşturulan sunucuya API'dan özel izin ile şifreli bir şekilde bağlantı sağlanmış ve istekler yanıtsız bırakılmamıştır. İki tabanda

geliştirilen bir uygulamada real time etkileşim olması gerektiğinden sunucu tarafında senkron bağlantı sağlanması için istekleri gönderen iki kullanıcı yaratılmıştır. Bu kullanıcılar istekleri kullandıktan sonra kendilerini kapalı konumuna alan erişim sağlayıcıları olarak nitelendirilebilir. Swift ve Java ile geliştirilen bu uygulamaların en az gecikme ile birlikte çalışabilmesi için veri tabanı erişimleri dikkatlice incelenerek bir sıra elde edilmektedir. Bu sıra kapsamında gecikme en aza indirilip müşterinin mağdur olmaması sağlanmıştır. [4, 7]

1.4.1. PHP ile Swift uygulaması için gerçekleştirilen Restfull API (Application Programming Interface) çalışması

Çalışma kapsamında bir önceki bölümde de bahsedildiği üzere Swift ile geliştirilen IOS uygulamasının MySQL gibi eski bir mimari ile bağlantısının sağlanması için PHP dilinde proje çalışmasına öz bir API çalışması gerçekleştirilmiştir. Bu API kapsamında sadece müşterilerin kullanımı için hazırlanan mobil uygulamanın veri tabanı ile anlık bağlantısı güvenli bir şekilde sağlanmış olmaktadır. Gerçekleştirilen API, proje çalışmasının iki birimde de anlık olarak doğru bir şekilde işlemde kalması gerekçesi nedeni ile kiralanan sunucunun uygun adresinde depolanmakta ve 7 gün 24 saat erişime açık bir şekilde saklanmaktadır. Herhangi bir hata veya arıza durumunda teknik ekibin ulaşabilmesi için istendiği takdirde sunucunun kendi ara birimi veya uzaktan bağlantı kullanılabilmektedir. [7, 8]

1.5. Sunucu ve Veri Tabanı Mimarisi

Sunucu olarak Linux tabanlı bir sunucu kullanıp üzerinde MySQL veri tabanı sağlanmış ve çalıştırılmıştır. Web üzerinde aktif gözlem ve hata çözümünde sağlanan hızdan dolayı phpMyAdmin kullanılmıştır. Geliştiren ekibin bilgilere rahatça ulaşması, düzenlemesi ve izlemesi bakımından geliştiricilere özel erişim hesapları ile MySQL Workbench bağlantıları sağlanmıştır. Bu bağlantılar sonucunda projede yapılan değişikliklere rahatça ulaşılmıştır. Gerçekleştirilen arayüzlerin anlık olarak sağlam bir şekilde çalışması ve iletişimdeki paketlerin çakışmaması için API ve Java uygulamasında ayrı kullanıcılar ile erişim verilmiştir. Müşterilerin alım emirlerinin çakışmaması için sunucuda zaman bazlı erişim sıralaması mevcuttur. [4, 7-10]

Örnek olarak gerçekleştirilen çalışmada veri tabanı altında sunucu adından bir veri tabanı oluşturulmuş ve test tabloları ile denemeler gerçekleştirildikten sonra gerçek veri tabanına geçilmiştir. Veri tabanında sürümlerden başlayarak karakter tipine kadar çakışmaya engel olmayı sağlayacak birçok aşamadan yararlanılmıştır. Bu aşamalar sayesinde sunucuda ve isteklerde çakışma gerçekleşmeden iletişim sağlanmıştır.

2. YAPILAN ARAŞTIRMALAR

Bu bölüm kapsamında proje çalışması süresinde gerçekleştirilen uygulamaların yapımı esnasında kaydedilmiş sorunlardan, yapılmış çalışmalardan, optimum çözümlerden ve optimizasyon problemlerinin nasıl aşıldığından bahsedilmiştir.

2.1. Java ile Yapılan Masaüstü Uygulamasının Geliştirilmesi Esnasında Gerçekleşen Araştırmalar, Sorunlar ve Çalıştırılmış Çözümleri

Oluşturulmuş olan otomasyon programında mevcut olan bütün bilgiler veri tabanında saklanmaktadır ve masaüstü uygulaması üzerinden gerçekleştirilen işlemler ile bu bilgilerin görüntülenmesi ve gerektiğinde değiştirilip, veri eklenmesi istenmektedir. Proje için oluşturulan veri tabanı bir sunucu üzerinden alınmaktadır ve bu nedenle yapılan projede bir bağlantı sınıfı oluşturulup, bu sınıf içerisinde sunucuya erişimi olan kullanıcı adı, şifre, veri tabanının ismi, HOST ip ve URL belli değişkenlere atanıp yönlendirilerek mevcut sunucu ile bağlantı sağlanmıştır. [11, 12]

Yapılan projede yönetici ve müşterinin farklı yetkileri olması gerekmektedir ve bu nedenle uygulamanın girişinde bir giriş ekranı oluşturulmuştur. Bu yetkilerin gerekli kişilere verilmesi amacıyla girilen bilgiler ile veri tabanındaki bilgiler karşılaştırılarak verilerin eşleşmesi halinde kullanıcılara yetkiler verilmelidir. Bu kapsamda belirlenen açıklamalar ile bağlantı gerçekleştirilip yetkiler verilmiştir.

Veri tabanından çekilen bütün veriler listelenmesine rağmen tek tek kontrol edilememektedir. Kontrolün sağlanamaması sonucunda bir filtreleme mekanizmasına ihtiyaç duyulmaktadır. Oluşan bu sorunun önüne geçmek amacıyla listelenen tablodaki değişkenlere göre “combobox”, “textfield” gibi bileşenler kullanılarak bu bileşenlere girilen değerler ile veri tabanındaki değerler karşılaştırılmıştır. [13]

Proje sona yaklaştıkça programda olan belirli hatalar göze çarpmıştır. Bu hatalardan örnek vermek gerekirse kullanıcının veri girmesi gereken “textfield” bileşenine veri girmeden yapılacak işlemin butonuna tıklanması programın çalışmamasına ve hata vermesine yol açacaktır. Belirli kontrol mekanizmaları ile bu açıklar kapatılmıştır.

Veri tabanında bir bilgi aranırken en kolay şekilde o bilgiye ulaşmak amaçlanır. Veri tekrarı ne kadar artarsa bir veriye ulaşmak da o kadar zorlaşacaktır. Veri tekrarını önlemek amacıyla kullanıcının girdiği verinin, aynı olması halinde mevcut kaydın üzerine eklenmeden o kayıt üzerinden işlem yapılması sağlanmıştır. [14]

2.2. Swift ile Yapılan Mobil Uygulamasının Geliştirilmesi Esnasında Gerçekleşen Araştırmalar, Sorunlar ve Çalıştırılmış Çözümleri

Verilerin tabloya yüklenme sırasında sürekli boş (nil) değerlerinin bulunduğuna dair hatalar almaktaydım. Bu yüzden mevcut sınıf-nesne yapısından ziyade "Araclar" sınıfını bir yapı (struct) şekline getirerek o yapıya ait bir nesne üretip tabloların gerekli indeks yerlerinde kullanılması için türettim. Türetilen nesne doğrudan "Arac" bilgilerini teker teker seçip gerekli satırlara koymamı kolaylaştırdı. Bu sayede sorun başarıyla çözülmüş oldu.

Tablodan satın alma ya da kiralama ekranlarına iletim gerçekleştirildiğinde geri dön butonları tabloyu yenileyemiyordu bu yüzden veri tabanında güncellenen fakat tabloda değişmeyen verilere tekrar işlem yapıldığında stüdyo tarafından tehditler algılanıyordu. Geri dön butonlarına ait yenileme (refreshing) işlevi tanıdım. Bildiriden sonra geri dönüldüğünde yapılan işlem hem veri tabanında hem de tablo ekranında güncelleşmeyi sağlamıştır. Bu sayede yeni işlem hataları meydana gelmeyip mevcut düzeni etkilememiştir.

Oluşturulan yeni Swift sınıflarına, halihazırda çalışan diğer dosyalardaki işlemlerin üst üste gelerek tanımlanma hatalarının meydana gelmesi kritik sorun oluşturmaktaydı. Oluşturulan textfield, label, button, image başka değişken isimleriyle değiştirilip mevcut sistemi etkilemeden yeni eklemeler yapılmasına olanak sağlamıştır.

Textfieldların boş döndürülmesi halinde sistem boş yanıt döndürüldüğüne dair kritik hatalar vermekteydi. Etkili buton aksiyonunun, alanların boş olması durumunda pasifleştirilerek mevcut sisteme beklenmedik tehditler oluşturması engellenmiştir.

2.3. Swift Uygulamasının Bağlantısı için PHP API'ın Geliştirilmesi Esnasında Gerçekleşen Araştırmalar, Sorunlar ve Çalıştırılmış Çözümleri

İlk bölümlerinde MySQL'in eski bir veri tabanı mimarisi olmasından dolayı Swift'in kullandığımız yeni sürümü ve yenilikçi geliştirmelerimiz ile çalışmadığından bahsetmiştik. Bunun akabinde veri tabanı bağlantısını gerçekleştirmek için kendi geliştirdiğimiz bir bağlantı aracını kullandık. Bu araç sunucuda veri tabanı bağlantısı için çalışan ve Swift uygulamasını yolladığı sorgulara anlık cevap verebilen bir araç.

Karşılaştığımız ilk sorunlardan birisi bu araç ile veri tabanı bağlantısını gerçekleştirememek oldu. Bu sorun karşısında aldığımız ilk aksiyon veri tabanından farklı bir port açığa çıkartıp bu port ile kendi oluşturduğumuz onaylı kullanıcıyı API'ın erişim sağlayıcı olarak kullanmasını sağlamaktı. Bunun için birkaç izin gerçekleştirdikten sonra bağlantıyı ayağa kaldırmayı başardık.

Karşılaştığımız bir diğer sorun ise veri tabanından gelen istek sonucunu tekrar Swift uygulamasına geri nasıl döndüreceğimizdi. HTTP protokolleri kapsamında zaten onaylı ve gerçekleşmiş bir emir için "Status Code" alabiliyorduk. Fakat gerçekleşen çalışmanın yanlış olması veya istenmeyen bir durum olması halinde birkaç senaryo belirledik ve bir data döndürmeye karar verdik. Bu datanın ufak boyutlu olması ve hızlı çözümlenebilmesi edilebilmesi bizim için önemli bir detaydı. Geri dönüş senaryolarında karar verilen sorunlar için önce API tabanında bazı karşılaştırmalar yazdık ve bunlara uygun datalar belirledik akabinde bu dataları Swift uygulamasında başarılı bir şekilde okuyup gerekli uyarı mesajları ile doğru bir kullanıcı deneyimi yaşatmayı hedefledik.

Karşılaştığımız bir diğer sorun ise tablodan doğru komutla MySQL verilerini çekememek oldu. Bu sorunun da temel sebebi PHP'nin sunucu tarafında çalışan sürümünün "mysql" kütüphanesinin kullanımının artık geçerli olmayışı. Bir diğer değişle kullanacağımız komutların yenilenmiş kütüphane olan "mysqli" tarafında çalışmasını sağlamak için tekrardan bütün kod parçalarımızı yeniden düzenledik. [7]

Bir diğer sorun ise hazırlanan PHP dosyasına Swift uygulamasının doğru adrese istek göndermesine rağmen ulaşamaması oldu. Sunucunun bazı kısımlarında oluşan adresleri kullanıcılara açarak ve bu kullanıcılara bu adreslerden cevaplar döndürerek

bu hatayı da gidermiş olduk. Fakat güvenlik açığı oluşmaması için kritik isteklerde ip erişim iznini aramayı hala bazı sebeplerden dolayı kapalı tutmaktayız. Erişim iznini bütün kullanıcılara açabilmek için çift taraflı denetim gerekmekte.

2.4. Sunucu ve Veri Tabanı Hakkında Gerçekleştirilen Araştırma Çalışmaları

Sunucu tarafında tablolarda "foreign key" atamalarının bir kısmı başarısız oldu. Bu sorunu çözmek için birkaç yöntem denendikten sonra tablo motor yapısını değiştirmek işe yarayan en mantıklı yöntem oldu. Bu yöntem dahilinde hata alınan tabloların yapı motorları InnoDB ile değiştirildi ve sorun başarı ile çözülmüş oldu. [15, 16]

Sunucu ilk etapta iki arayüze de tekil yönlendirme yapılacak şekilde çalıştırılmaktaydı fakat bir süre sonra verilerin ve alım emirlerinin çakıştığını fark ettik. Bu sorunu çözmek içinse yaptığımız mimaride birkaç ayarı değiştirmeye karar verdik ve yeni bir veri tabanı ile kullanıcı sıralamasını ve arayüzleri kendilerine özel bir dizilimle oluşturulmuş kullanıcılar ile çalıştırmaya başladık. Ayrık kullanıcılar ve ayrık ip adreslerinden gelen emirler doğrultusunda bir daha çakışma sorunu yaşanmamıştır.

3. TASARIM

Bu bölümde proje çalışması kapsamında masaüstü ve mobil uygulamaların arayüz tasarımlarında dikkat edilen noktalar ve tasarım aşamaları anlatılmıştır. Uygulamanın bütün arayüz grafikleri CorelDRAW 2020 kullanılarak hazırlanmıştır.[17]

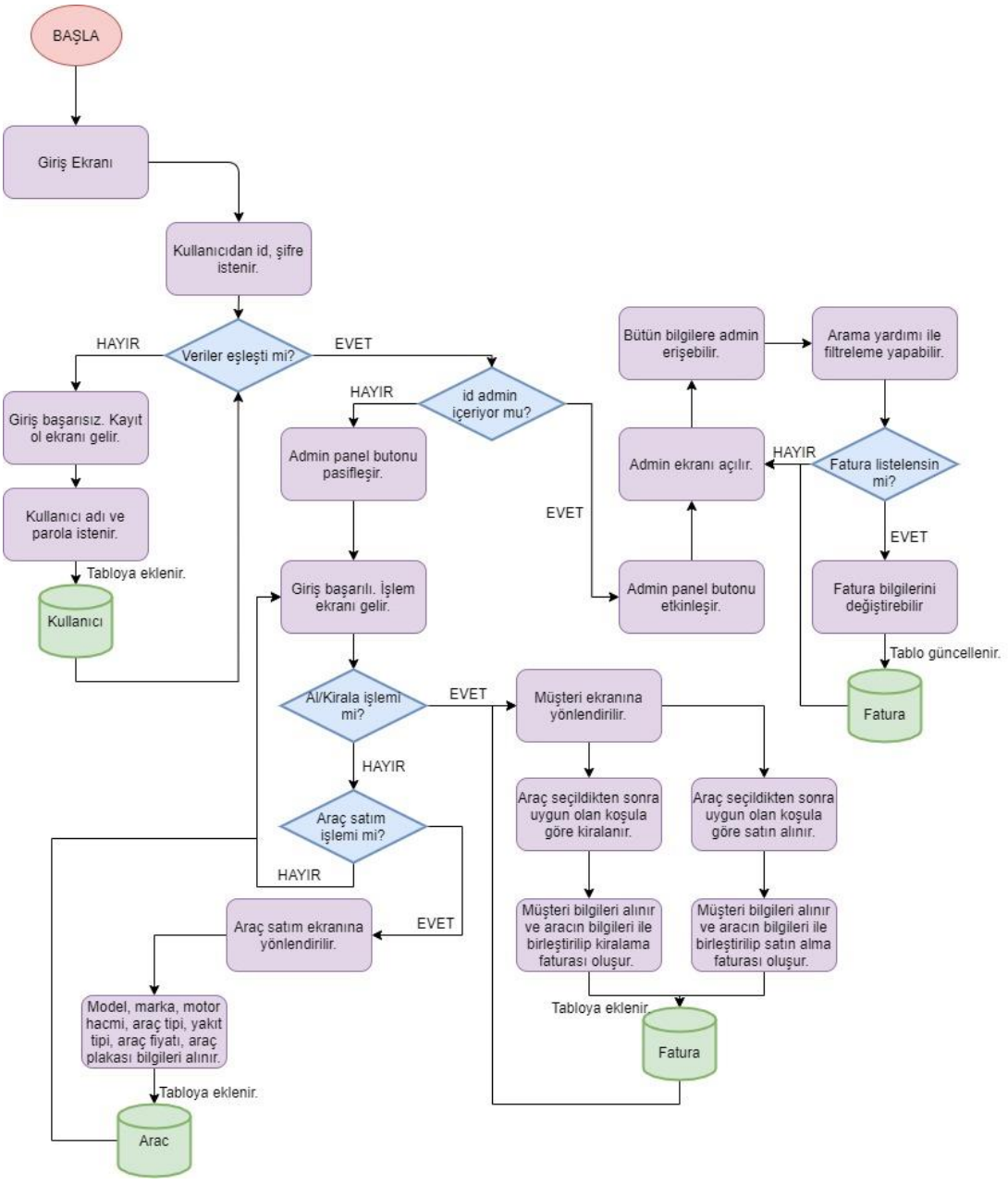
3.1. Masaüstü Uygulamasının Tasarım Aşamaları

Bu projenin masaüstü uygulaması tasarlanırken müşterilerin gözlerini yormayacak ve alımlarındaki adımları görmeyi zorlaştırmayacak bir UX planlanmıştır. Sadelik ve minimal bir tasarım anlayışı kullanılarak bir arayüz oluşturulmuştur. Kullanıcı ve yöneticilerin ulaşmak istedikleri şeye ulaşabilmeleri sağlanmıştır.

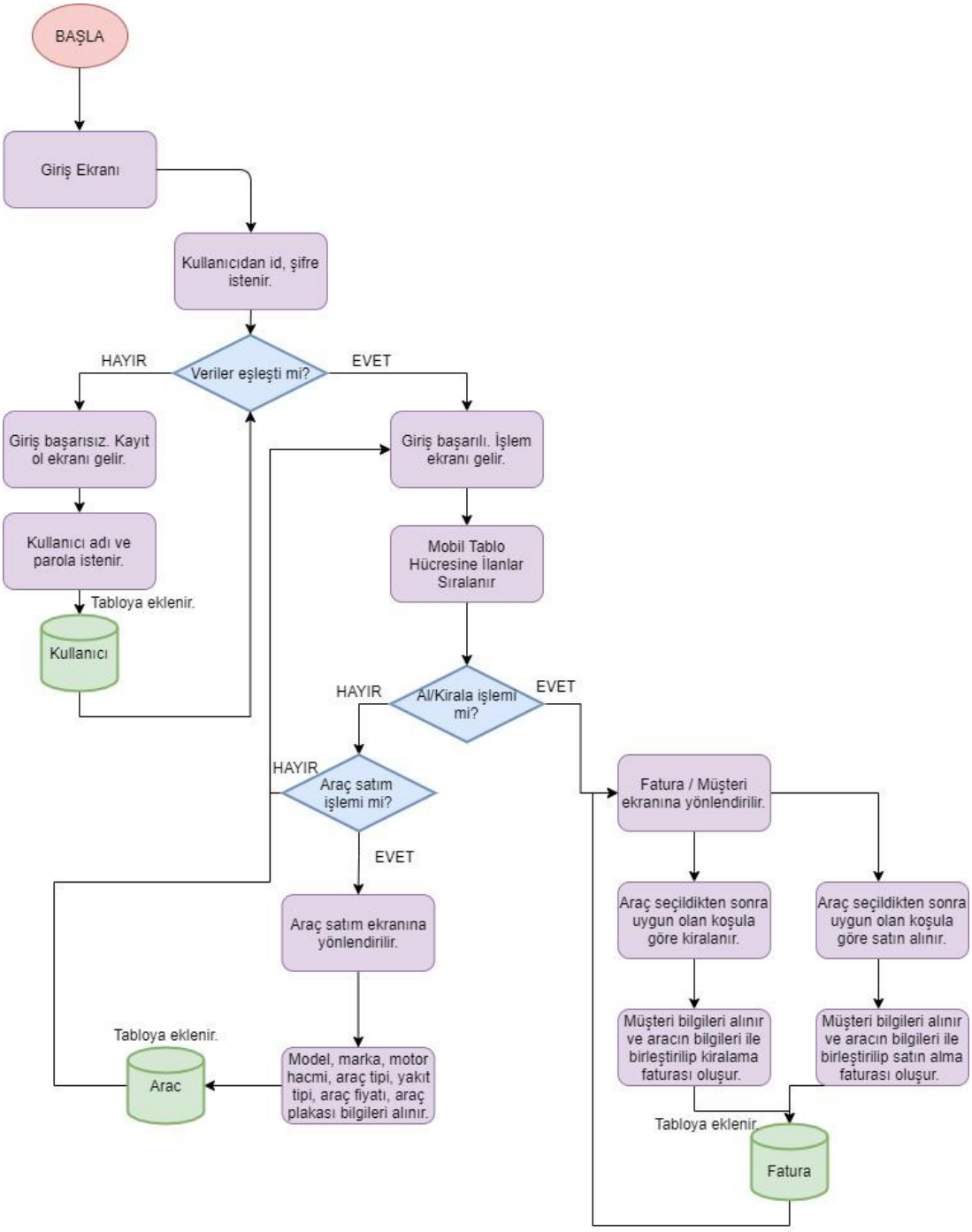
3.2. Mobil Uygulamanın Tasarım Aşamaları

Bu projenin mobil uygulaması tasarlanırken müşterilerin görmek istediklerini tek seferde görebilecekleri bir tasarım kullanılmıştır. Uyarı kutucukları ile de adım adım müşteriye bir yol gösterme mekanizması oluşturulmuştur. Seçilen grafiklerin sadeliği koruması ve göz yormaması da önceliklerimizden biri olarak belirlenmiştir.

4. AKIŞ ŞEMASI



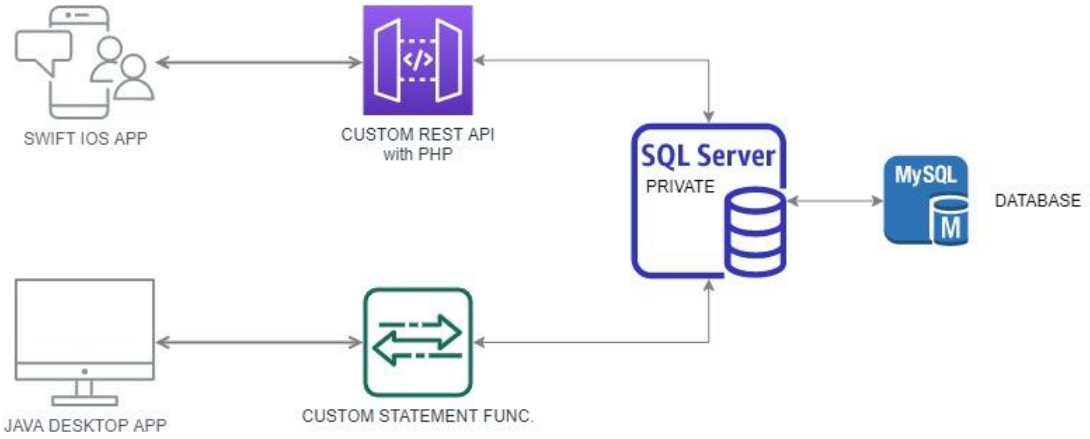
Şekil 4.1. Java ile geliştirilen uygulamaya ait akış şeması



Şekil 4.2. Swift ile geliştirilen uygulamaya ait akış şeması

5. YAZILIM MİMARİSİ

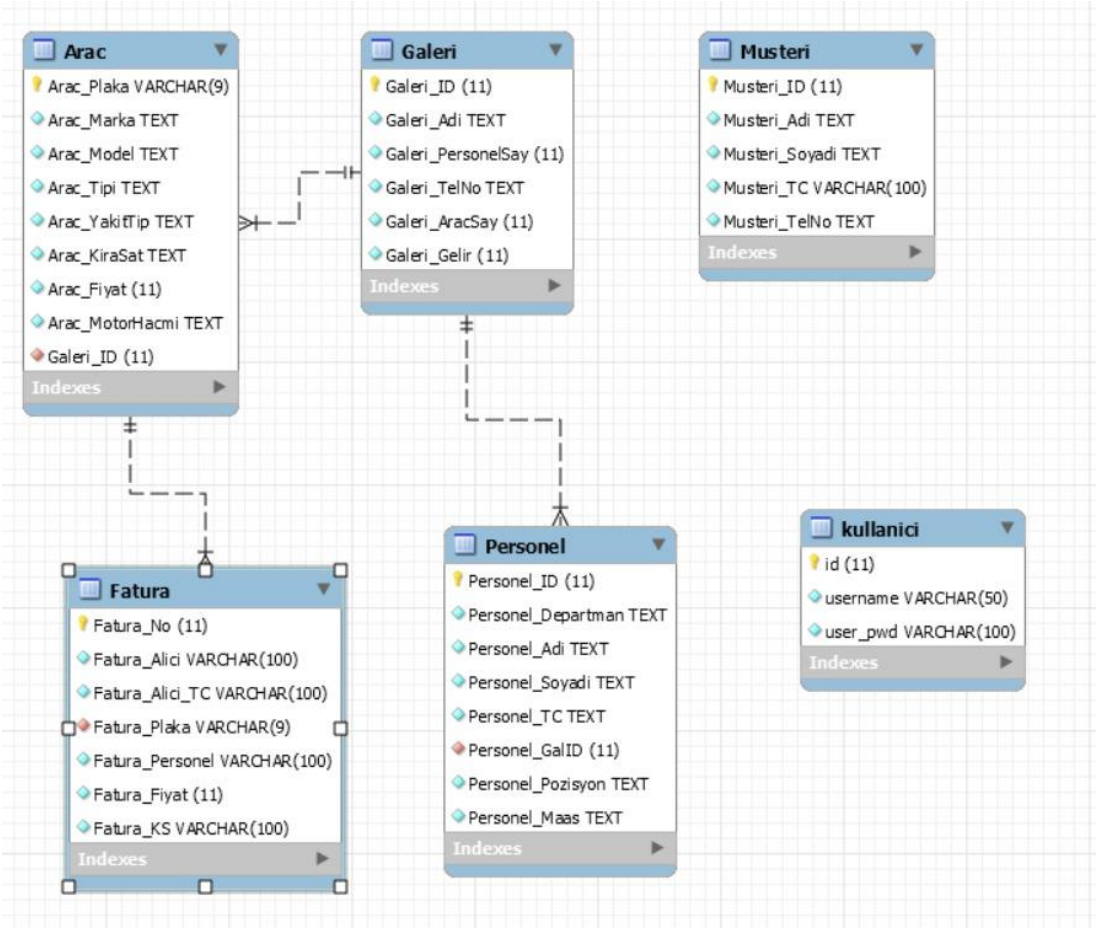
Proje çalışması kapsamında gerçekleşen sistemin yapısını tanımlayan biçimsel modeldir. Bizim geliştirdiğimiz sistemin modeli ise Şekil 5.1.'de gösterilmektedir.



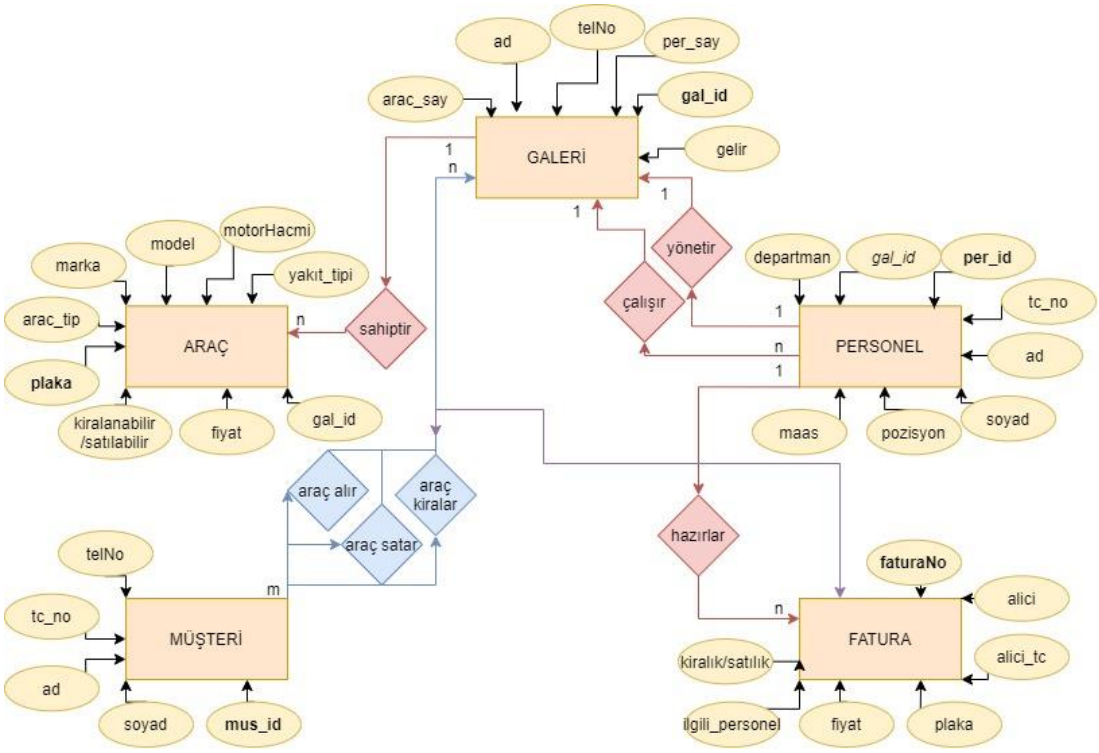
Şekil 5.1. Sistemin hem mobil hem de masaüstü kısmının bir arada rol aldığı mimari

6. VERİ TABANI DİYAGRAMI

Proje çalışması kapsamında geliştirilen modüllerin çalıştırıldığı ve kullanıldığı veri tabanının ER diyagramı Şekil 6.1.'de gösterilmektedir. Veri tabanını çalıştırılabilir boyuta getirmek için kılavuz olarak draw.io üzerinde hazırlamış olduğumuz diyagram ise Şekil 6.2.'de gösterilmektedir. [18]



Şekil 6.1. MySQL Workbench üzerinden alınmış ER diyagramı



Şekil 6.2. Draw.io üzerinde hazırlanmış planlanan veri tabanı diyagramı

7. GENEL YAPI

Bu bölümde geliştirilen mekanizmaların anlık çalışma yapısından ve gelişmelerinden bahsedilmiştir.

7.1. Mobil ve Masaüstü Modüllerinin Anlık Çalışma Yapısı

Mobil modüllerin tasarımında ve gelişiminde veri tabanına sadık bir yapı kurularak müşterinin kullanımının kolaylaşması amaçlanmıştır. Bu kapsamda mobilden bir alım gerçekleştiren müşteri için masaüstü uygulamasında yönetici giriş yaparak oluşturulmuş işleme ait faturayı anlık olarak görebilmektedir. Bu sistemi gerçekleştirirken iki yapıya ihtiyaç duyduk. Gerçek zamanlı destek ve anlık cevap modeli. Bu modelleri yaratmak için masaüstü uygulamamızda dahili erişim modüllerine yer verdik. Bu modüller anlık olarak platform için oluşturulan kullanıcının izni ile verilere müdahale edebilecekti. Mobil kısımda ise bu iletişimi ayrıca oluşturduğumuz bir modül ile sağladık. Bu modül için de tıpkı masaüstü uygulamasında yaptığımız gibi ayrı bir kullanıcı ile asenkron bir iletişim sağladık. Bunu sağlarken PHP ile ayrık bir API geliştirdik. Mobil müşteri işlemlerine özel tanımlanmış bu API, bizim için yeterli sayıda yönlendirme ile işlerimizi gerçekleştirmemizi sağladı. İşlemin bu noktasında gerekli linke erişimi sadece mobil modüller ile sağladık. Yani bir iletişim çakışmasını önlemek için masaüstü uygulamamız ile veri tabanına erişmek için aynı yolu kullanmalarını önledik. Bu sayede de düşük gecikmeli senkron işlem yapısı oluşturmuş olduk. [19]

7.2. Veri Tabanı ve Bağlantı Modüllerinin Anlık Çalışma Yapısı

Veri tabanı tasarımı yapılırken eşlenik değerlerin birbiri arasında bağlantı olacağı bir tasarım çizgisinde ilerlenerek yapılmıştır. Bu ilgili verinin hatasız ve en doğru bir şekilde tablolar arasında transferinin sağlanabilmesi için önemli bir özellik olmuştur. Veri tabanı oluşturulurken 5N kuralına dikkat edilmiştir. Faturaların bir firma için hayati öneme sahip olduğunu düşündüğümüzde herhangi bir verinin hatalı olması veya yanlış bir bilgi içermesi büyük bir sorun teşkil etmekte olduğundan bu mekanizma çok önemlidir. Veri tabanında belirlenen bağıntılar ve bu bağıntıların eksik oluşturulmasına izin verilmemesi için gerekli araçlar üzerinden seçilen kriterler ile sorunsuz bir veri tabanı yapısı sağlanmıştır. Her uygulamanın bilgi getiren modülün

farklı bağlantılar üzerinden sağlanması ise veri tabanında karşılaşılabilecek anlık sorunların önüne geçmiştir. Bu bizim için gerçekten olmazsa olmaz bir özellik oldu. Akabinde veri tabanına sorunsuz bir bağlantı sağlanması için gerekli güvenlik işlemlerinden bazıları anlık olarak test sunucu yapımızda çalışmaktadır. Bazı özellikler için ise daha fazla zaman gerekmektedir. Fakat şimdilik sistem sorunsuz bir biçimde mobil ve masaüstü müşterilere hizmet vermekte ve yönetici portalından izlemeye izin vermektedir.

KAYNAKLAR

[1] https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html (Ziyaret tarihi: 18 Mayıs 2020)

[2] https://docs.oracle.com/javase/7/docs/api/ (Ziyaret tarihi: 16 Mayıs 2020)

[3] https://swift.org/ (Ziyaret tarihi: 20 Mayıs 2020)

[4] https://dev.mysql.com/doc/refman/8.0/en/ (Ziyaret tarihi: 10 Mayıs 2020)

[5] https://docs.oracle.com/netbeans/nb82/netbeans/index.html (Ziyaret tarihi: 12 Mayıs 2020)

[6] https://developer.apple.com/documentation/xcode/ (Ziyaret tarihi: 22 Mayıs 2020)

[7] https://www.php.net/manual/tr/book.mysqli.php (Ziyaret tarihi: 18 Mayıs 2020)

[8] https://www.php.net/manual/en/index.php (Ziyaret tarihi: 18 Mayıs 2020)

[9] https://docs.phpmyadmin.net/en/latest/ (Ziyaret tarihi: 10 Mayıs 2020)

[10] https://dev.mysql.com/doc/workbench/en/ (Ziyaret tarihi: 12 Mayıs 2020)

[11] https://www.javatpoint.com/example-to-connect-to-the-mysql-database (Ziyaret tarihi: 16 Mayıs 2020)

[12] https://stackoverflow.com/questions/2839321/connect-java-to-a-mysql-database (Ziyaret tarihi: 16 Mayıs 2020)

[13] https://stackoverflow.com/questions/2494868/rowfilter-regexfilter-multiple-columns (Ziyaret tarihi: 17 Mayıs 2020)

[14] https://stackoverflow.com/questions/30674612/get-selected-cell-value-in-jtable/30674750 (Ziyaret tarihi: 20 Mayıs 2020)

[15] https://dev.mysql.com/doc/refman/8.0/en/innodb-storage-engine.html (Ziyaret tarihi: 23 Mayıs 2020)

[16] https://docs.cpanel.net/cpanel/ (Ziyaret tarihi: 5 Mayıs 2020)

[17] https://www.coreldraw.com/en/product/coreldraw/ (Ziyaret tarihi: 20 Mayıs 2020)

[18] https://draw.io/ (Ziyaret tarihi: 16 Mayıs 2020)

[19] https://developer.okta.com/blog/2019/03/08/simple-rest-api-php (Ziyaret tarihi: 14 Mayıs 2020)

[20] Date, C. J., The New Relational Database Dictionary: Terms, Concepts, and Examples., "O'Reilly Media, Inc.", p. 163., (December 21, 2015).