T.C.

KARABÜK ÜNİVERSİTESİ

TÜRKİYE ODALAR VE BORSALAR BİRLİĞİ

TEKNİK BİLİMLER

MESLEK YÜKSEKOKULU

BİLGİSAYAR PROGRAMCILIĞI PROGRAMI

2020-2021 BAHAR YARIYILI

TBP204- SİSTEM ANALİZİ VE TASARIMI DERSİ (A Şubesi)

PROJE RAPORU

GALERİ OTOMASYONU

HAZIRLAYAN

1905113063 ENES ÖZKAN

Bilgisayar Programcılığı

Haziran 2021

İçindekiler Sayfa No

1. GİRİŞ

Bu Rapor Galeri Otomasyonu (GO) için yazılım geliştirme süreçlerinin çeşitli seviyelerde detaylandırılması ve geliştirme safhalarının belgelenmesi amacıyla düzenlenmiştir. Geliştirilmek istenen sisteme özgü gereksinimlere bağlı olarak hazırlanmıştır. Rapor, gereksinim raporunda belirtilen isteklerin gerçeğe dönüştürülmüş halini yansıtarak, yazılım geliştirici için gerekli ilişkisel veri tabanı tasarımını, tüm metotları, ana ve alt mimariyi şematik ve diyagramlar yardımıyla anlatmaktadır. Veri tabanı tasarımı ile yazılıma ilişkin verilerin saklanacağı tabloların veri bütünlüğü, veri tutarlılığı ve veri güvenliği prensiplerine uygun olarak normalizasyon kuralları çerçevesinde ilişkileri sunulmaktadır. Metotlar ve fonksiyonlar ile tüm yapı içindeki gerçekleşmesi gereken işlemleri tanımlanmaktır. Ana ve alt sistem mimarisinin diyagramları da işlemlerin nasıl yürütüldüğü kodlayıcı tarafından nasıl geliştirilmesi gerektiği hakkında bilgi vermektedir. Yazılıma ilişkin gerekli veri sözlüğü hazırlanmış, raporu oluştururken kullanılan araçlar hakkında gerekli açıklamalar yazılım geliştiricilerim bilgisine sunulmuştur.

2. TEKNİK TERİMLER

Galeri: İkinci el veya sıfır araç alıp satan işletme.

Yatırım: Bir birikimi, belli bir süre boyunca, enflasyon oranı, vade uzunluğu ve gelecekteki nakit akışlarının belirsizliği faktörlerini bertaraf edebilecek bir gelir sağlayabilmek için bugünden ayırmaktır.

Nesne Bağıntı Diyagramı (Entity/Relationship Diyagram): E-R veri modeli gerçek dünyanın nesneler ve bu nesneler arasındaki ilişkiler kümesi olarak ifade edilmesinde yararlanılır. Özellikle veri tabanı tasarımında, veri tabanının kavramsal yapısını ortaya koymak için kullanılır.

a) Varlık (Entity): Var olan ve benzerlerinden ayırt edilebilen her şeye, her nesneye bir varlık (entity) denir.

b) Varlık Kümesi (Entity Set): Aynı türden benzer varlıkların oluşturduğu kümeye varlık kümesi (entity set) denir.

c) Nitelikler (Attributes): Bir varlık kümesindeki varlıkların özelliklerini göstermek ve varlıkları birbirinden ayırt etmek için nitelikler (attributes) kullanılır.

ç) Değer Alanı (Domain): Her niteliğin bir değer alanı vardır. Değer alanı, ilgili niteliğin olurlu değerlerinin tümünü içeren bir kümedir.

d) Bağıntı (Relationship): İki ya da daha çok sayıdaki varlığın birleşmesi, bir araya gelmesi, aralarında ilişki kurulmasına bağıntı denir. Örneğin bir öğrenci ile bir ders, firma ile malzeme, kişi ile otomobil gibi. Bunlar ikili bağıntılardır. Bir isçi, ürün ve makine bir araya getirilirse bu bağıntı üçlü olur.

e) Bağıntı Kümesi (Relationship Set): Aynı tür benzer bağıntıların kümesine bağıntı kümesi (relationship set) adı verilir.

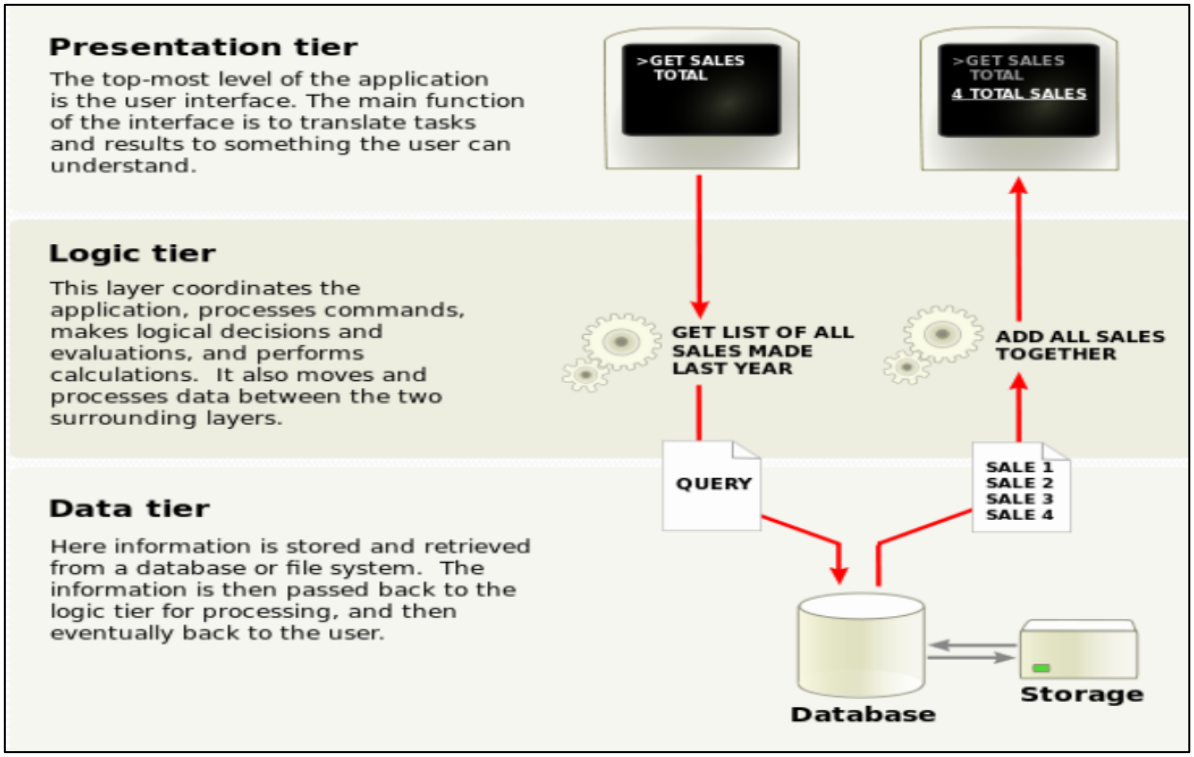
3. YAZILIM TASARIMINA AİT GENEL BİLGİLER

3.1. Yazılım Yetenekleri

- Kullanıcı, Mevcut Araç listesini görüntüleyebilir.
- İlanda olan ve olmayan araçları listeleyebilir.
- Araçlar eklendikten sonra hata varsa düzeltebilir.
- Kullanıcı, Gelir/ Gider işlemleri yapabilir.
- Etiketlemeler yapılarak verimli bir gelir/gider takibi gerçekleştirilir.
- Satılmış Araçlarını Satılan Araçlar Menüsünden görüntüleyebilir.
- Aracı Alış ve Satış fiyatına göre Kar oranını görüntüleyebilir.

3.2. Yazılım Mimarisi

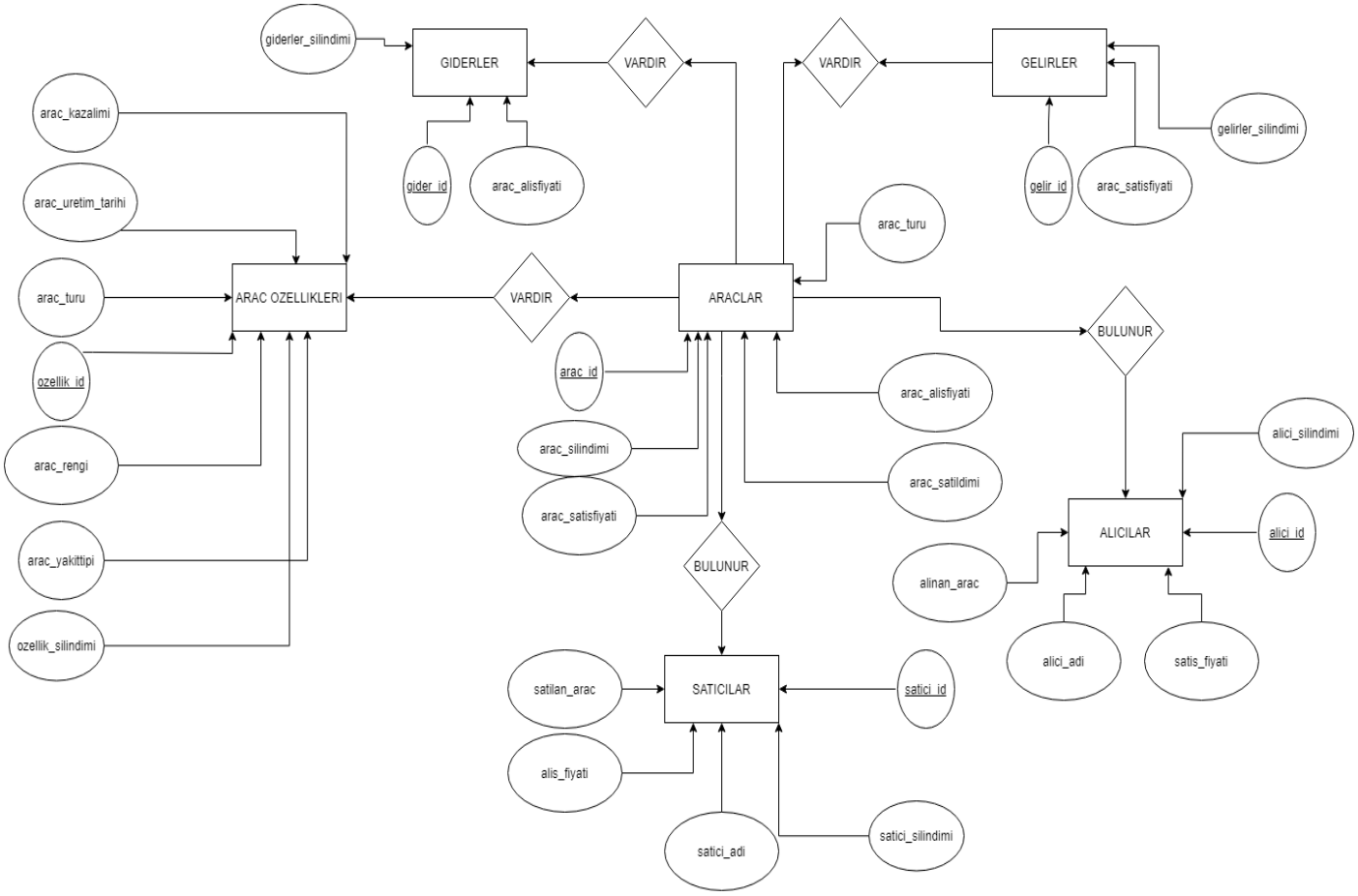
GO üç katmanlı mimari ile fonksiyonel işlem mantığı (iş kuralları), kullanıcı ara yüzü (sunum), veri saklama ve veri erişiminin bağımsız modüller olarak ve farklı platformlar üzerinde geliştirildiği ve sürdürüldüğü bir istemci -sunucu mimarisi tekniği kullanılarak tasarlanmıştır.



Şekil 1. Üç katmanlı yazılım mimarisi.

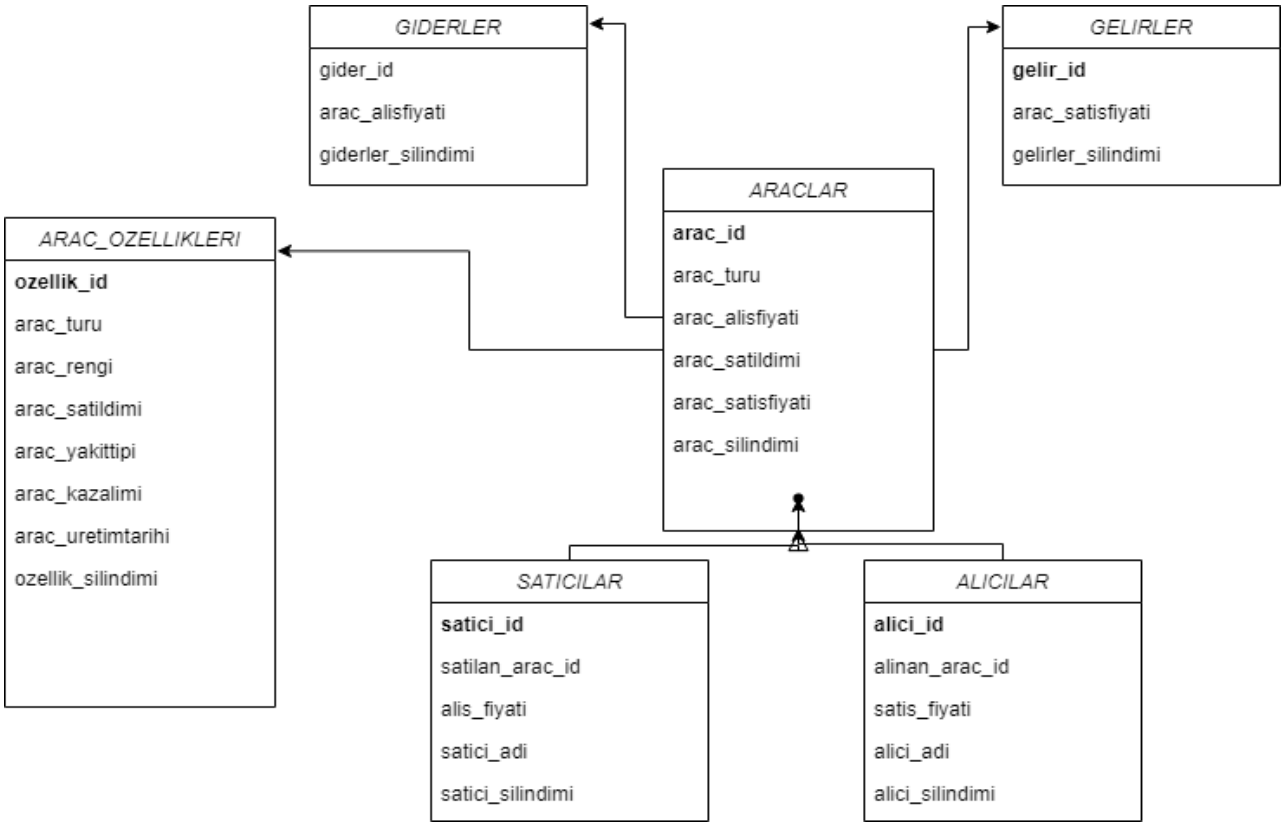
3.3. Veri Tabanı Tasarımı

3.3.1. E-R Tasarımı



Şekil 2. Entity Relation Diyagramı.

3.3.2. Veri Tabanı Tablo Tasarımı



Şekil 3. Veri Tabanı tablo tasarımı.

3.3.3. Veri Sözlüğü

Araçlar Tablosu (ARACLAR):

Bu tabloda kullanıcının araçları kategorize etmesi için veriler bulunmaktadır.

Arac_id: Tablonun primary key'idir. INT tipindedir. Otomatik olarak artmaktadır.

Arac_turu: Tablonun INT tipi araç türü değeridir.

Örn: Arac_turu = 1, etiket: “Araba”

Arac_alisfiyati: FLOAT tipinde aracın galeriye geliş fiyatıdır.

Örn: Arac_alisfiyati = 82.198.45, etiket: “Sekseniki bin Yüzdoksansekiz lira 45 Kuruş.”

Gider Türleri Tablosu (gider_tur):

Bu tabloda kullanıcının giderlerini kategorize etmesi için gider türleri tutulmaktadır.

id: Tablonun primary key'idir. INT tipindedir. Otomatik olarak artmaktadır.

etiket: Gider türleri için açıklama yapmaktadır.

Örn: id=2, etiket: 'İletişim

Gelirler Tablosu (Gelirler):

Tabloda gelirler hakkında bilgi tutulmaktadır.

id: Tablonun primary key'idir. INT tipindedir. Otomatik olarak artmaktadır.

gelir_tur_id: Gelir türleri tablosundan id'leri alır. Foreign key olarak gelir_tur tablosuna bağlıdır.

odeme_sekli_id: Odeme_sekli tablosundan id'leri alır. Foreign key olarak odeme_sekli tablosuna bağlıdır.

user_id: Kullanıcılar tablosundan id'leri alır. Foreign key olarak Kullanicilar tablosuna bağldır.

etiket: Gelirler için açıklama yapmaktadır.

tarih: Gelirin gerçekleştiği tarihi tutmak için kullanılmaktadır. DateTime tipindedir.

miktar: Gelirin miktarını belirler.

periyodikmi: Gelirin periyodik olup olmadığını tutmak için kullanılır. Bool tipindedir.

Örn: id=2, gelir_tur_id = 3, odeme_sekli_id= 2, user_id=1, etiket : 'Kasım ayı maaş', tarih : 16.11.2005, miktar = 900.00, periyodikmi : True

Giderler Tablosu (Giderler):

Tabloda giderler hakkında veriler tutulmaktadır.

id: Tablonun primary key'idir. INT tipindedir. Otomatik olarak artmaktadır.

gider_id: Gider türleri tablosundan id'leri alır. Foreign key olarak gider_tur tablosuna bağlıdır.

odeme_sekli_id: Odeme_sekli tablosundan id'leri alır. Foreign key olarak odeme_sekli tablosuna bağlıdır.

user_id: Kullanıcılar tablosundan id'leri alır. Foreign key olarak Kullanicilar tablosuna bağldıır.

etiket: Giderler için açıklama yapmaktadır.

tarih: Giderin gerçekleştiği tarihi tutmak için kullanılmaktadır. DateTime tipindedir.

miktar: Giderin miktarını belirler.

periyodikmi: Giderin periyodik olup olmadığını tutmak için kullanılır. Bool tipindedir.

Örn: id=2, gider_tur_id = 3, odeme_sekli_id= 2, user_id=1, etiket : '100 Kontör', tarih : 17.11.2005, miktar = 13.00, periyodikmi : False

Alarm Tablosu (Alarm):

Tabloda kullanıcının belirttiği hatırlatmaların bilgileri tutulmaktadır.

id: Tablonun primary key'idir. INT tipindedir. Otomatik olarak artmaktadır.

gider_tur_id: Gider türleri tablosundan id'leri alır. Foreign key olarak gider_tur tablosuna bağlıdır.

etiket: Alarmlar için açıklama yapmaktadır.

max_miktar: Uyarı verilecek miktarı belirtmektedir.

basla_tarih: Alarm için başlangıç tarihini tutar.

bitis_tarih: Alarm için bitiş tarihini tutar.

Örn: id=3, gider_tur_id=3, etiket:'İletişim masrafı 300 ytl'yi geçerse uyar', max_miktar=300, basla_tarih :10.11.2005, bitis_tarih: 10.12.2005

Borçlar Tablosu (Borc):

İleri tarihte gerçekleşecek olan giderlerin bilgilerini tutmaktadır.

id: Tablonun primary key'idir. INT tipindedir. Otomatik olarak artmaktadır.

gider_tur_id: Gider türleri tablosundan id'leri alır. Foreign key olarak gider_tur tablosuna bağlıdır.

etiket: Borçlar için açıklama yapmaktadır.

odeme_sekli_id: Odeme_sekli tablosundan id'leri alır. Foreign key olarak odeme_sekli tablosuna bağlıdır.

son_odeme_tarih: Borçlar için son ödeme tarihini belirtir.

miktar: Borcun miktarını tutmaktadır.

odendi: Borcun ödenip ödenmediğini kontrol etmek için kullanılmaktadır. Bool tipindedir.

odeme_tarih: Borcun ödeme tarihini belirtir.

id=1, gider_tur_id = 3, etiket:'Su faturası', odeme_sekli_id = 2, son_odeme_tarih:10.12.2005, miktar: 30.00, odendi : False, odeme_tarih:8.12.2005

Ödeme Şekilleri Tablosu (Odeme_Sekli):

Ödeme şekilleri ile ilgili bilgileri tutmak için kullanılmaktadır.

id: Primary key'idir. INT tipindedir. Otomatik olarak artmaktadır.

etiket: Ödeme şekilleri için açıklama yapmaktadır.

doviz_id: Dövizler tablosundan id'leri alır. Foreign key olarak Dovizler tablosuna bağlıdır.

hesap_id: Hesaplar tablosundan id'leri alır. Foreign key olarak Hesaplar tablosuna bağlıdır.

Örn: id=1, etiket:'Nakit YTL', doviz_id=1, hesap_id=2

Kullanıcılar Tablosu (Kullanici):

Programa kayıtlı olan kullanıcıları tutmak için kullanılmaktadır.

id: Primary key'idir. INT tipindedir. Otomatik olarak artmaktadır.

isim: Kullanıcı adı alanıdır.

sifre: Kullanıcıların şifresini tutar.

auth: Kullanıcının yetkili olup olmadığını belirtir. Bool tipindedir.

Örn: id=3, isim: 'Kemal', sifre:'KML', auth: True

Tasarruf Planları Tablosu (Tasarruf_Plan):

Yapılan tasarrufların planı bu tabloda tutulur.

id: Primary key'dir. INT tipindedir. Otomatik olarak artmaktadır.

tasarruf_id: Tasarruflar tablosundan id'leri alır. Tasarruflar tablosuna Foreign Key olarak bağlıdır.

tarih: Ödeme tarihini belirtir.

miktar: Ödemenin miktarını belirtir

odeme_sekli_id: Odeme_sekli tablosundan id'leri alır. Foreing Key olarak Odeme_sekli tablosuna bağlıdır.

Örn: id=2, tasarruf_id=3, tarih: 12.11.2005, miktar= 200.00, odeme_sekli_id=2

Tasarruflar Tablosu (Tasarruflar):

Kullanıcının amaçladığı finansal değere belli bir süre içinde ulaşabilmesi için, gerekli olan bilgilerin tutulduğu tablodur.

id: Primary key'dir. INT tipindedir. Otomatik olarak artmaktadır.

etiket: Açıklama amacıyla kullanılan alandır.

basla_tarih: Tasarrufun başlangıç tarihini belirtir.

bitis_tarih: Tasarrufun bitiş tarihini belirtir.

hedef_miktar: Hedeflenen tasarruf miktarını tutan alandır.

Örn: id=2, etiket:'Laptop alacağım', basla_tarih: 03.03.2005, bitis_tarih: 03.06.2005, hedef_miktar: 1200.00

Dövizler Tablosu (Dovizler):

Dövizlerin ve tiplerinin tutulduğu tablodur.

id: Primary key'dir. INT tipindedir. Otomatik olarak artmaktadır.

etiket: Döviz tipini belirtir.

birim_fiyat: Parite bilgilerini. Double tipindedir.

Örn: id=3, etiket ='Amerikan Doları', birim_fiyat = 1.58

Hesaplar Tablosu (Hesaplar):

Banka hesaplarını tutmak için kullanılır.

id: Primary key'dir. INT tipindedir. Otomatik olarak artmaktadır.

banka_id: Bankalar tablosundan id'leri alır. Bankalar tablosundan id alanına Foreign Key olarak bağlıdır.

hesap_no: Kullanıcının banka hesaplarının numaralarını tutmak için kullanılır.

user_id: Kullanıcılar tablosundan id'leri alır. Kullanıcılar tablosuna Foreign Key olarak bağlıdır.

etiket: Hesaplar için açıklama alanı olarak tutar.

kredilimi: Banka hesabının kredili olup olmadığını tutmak için kullanılır. Bool tipindedir.

Örn: id=3, banka_id=2, hesap_no=123456, user_id=2, etiket='Maaş Hesabı', kredilimi= False

Bankalar Tablosu (Bankalar):

Bankaların tutulduğu tablodur.

id: Primary key'dir. INT tipindedir. Otomatik olarak artmaktadır.

etiket: Banka isimlerinin tutulduğu alandır.

Örn: id=1, etiket:'Akbank'

Yatırımlar Tablosu (Yatirimlar):

Sahip olunan her türlü yatırımın bilgisini tutmaktadır. Örneğin : Döviz, araba, gayrimenkul.

id: Primary key'dir. INT tipindedir. Otomatik olarak artmaktadır.

yatirim_tur_id: Yatırım türlerinin id'lerini yatırım türleri tablosundan alır. Foreign key olarak Yatırım_Tur tablosunun id alanına bağlıdır.

etiket: Yatırımlar için açıklama alanı olarak kullanılmaktadır.

alis_fiyati: Yatırımın alış fiyatını tutar.

alis_tarihi: Yatırımın alış tarihini tutar.

su_anki_fiyati: Yatırımın şu anki fiyatını tutar.

satildimi: Yatırımın satılıp satılmadığını tutmak için kullanılır. Bool tipindedir.

satis_tarihi: Yatırımın satış tarhini tutmak için kullanılır.

satis_fiyati: Yatırımın satış fiyatını tutmak için kullanılır.

dovizler_id: Alış satış fiyatının döviz tipini tutmak için kullanılır.

Örn: id=1, yatirim_tur_id=2, etiket:'Marmaris'teki arazim', alis_fiyati: 1000, alis_tarihi: 02.03.1999, su_anki_fiyat=2000, satildimi: True, satis_tarihi: 05.06.2004, satis_fiyati=2000, dovizler_id = 3

Yatırım Türleri Tablosu (Yatirim_Tur):

Yatırım türlerini tutmak için kullanılır.

id: Primary key'dir. INT tipindedir. Otomatik olarak artmaktadır.

etiket: Yatırım türlerini belirtmek için kullanılan alandır.

Örn: id=3, etiket : 'Arazi'

4. YAZILIM MODÜLLERİ

4.1. Gelirler Modülü

Gelirler modülü kullanıcının gelirlerini ekleyerek, eklenen kayıtlar üzerinde daha sonra düzenleme, silme gibi işlemler yapabileceği bir modüldür.

4.1.1. Gelir Ekle: Kullanıcı bu form üzerinden gelirle ilgili olan etiketi seçecektir. Etiket var olmadığı takdirde etiketi oluşturup, gerekli alanları da doldurarak veritabanındaki ilgili tablolara gelir kaydını ekleyecektir.

4.1.2. Gelirler Göster: Kullanıcıya bu formda daha önce veritabanına eklediği kayıtları tablo şeklinde listeleme hizmeti sunulacaktır.

4.1.3. Gelirleri Düzenle – Sil: Bu formda kullanıcı listeden seçtiği herhangi bir gelir kaydının özelliklerini yeniden düzenleyebilecektir. Kullanıcını isteği doğrultusunda seçili gelir kaydı tümüyle veritabanından silinebilecektir.

4.2. Giderler Modülü

Bu modül kullanıcının giderlerini ekleyerek, eklenen kayıtlar üzerinde daha sonra düzenleme, silme gibi işlemler yapabileceği bir modüldür.

4.2.1. Gider Ekle: Kullanıcı bu form üzerinden giderle ilgili olan etiketi seçecektir. Etiket var olmadığı takdirde etiketi oluşturup, gerekli alanları da doldurarak veritabanındaki ilgili tablolara gelir kaydını ekleyecektir. Eğer eklenen gider periyodik(taksit) olarak gerçekleşiyorsa o zaman ileri tarihte gerçekleşecek olan giderler(sonraki taksiler) Fatura&Borç ile ilgili tabloya eklenecektir.

4.2.2. Giderleri Göster: Kullanıcıya bu formda daha önce veritabanına eklediği kayıtları tablo şeklinde listeleme hizmeti sunulacaktır.

4.2.3. Giderleri Düzenle – Sil: Bu formda kullanıcı listeden seçtiği herhangi bir gider kaydının özelliklerini yeniden düzenleyebilecektir. Kullanıcını isteği doğrultusunda seçili gider kaydı tümüyle veritabanından silinebilecektir.

4.3. Banka Hesapları Modülü

Kullanıcının sahip olduğu banka hesaplarını kaydedebileceği, kayıtlar üzerinde düzenleme ve silme gibi işlemler yapabileceği bir modüldür. Aynı zamanda kullanıcı hesaplarının takibini de bu modül üzerinden yürütebilecektir.

4.3.1. Banka Hesabı Ekle: Kullanıcı bu form üzerindeki alanları doldurarak hesaplarını kaydedebilecektir.

4.3.2. Banka Hesaplarını Göster: Kullanıcıya kayıtlı banka hesapları, bu hesapların özellikleri, ilgili hesabın bakiye bilgilerini bu formda liste şeklinde gösterilecektir. Ayrıca farklı hesap bakiyelerinin toplam miktarı da hesaplanarak kullanıcıya bu formda gösterilecektir.

4.3.3. Banka Hesaplarını Düzenle – Sil: Bu form üzerinde kullanıcı seçtiği banka hesabının bakiyesini ve diğer özelliklerini düzenleyebilecektir. Kullanıcı istediği takdirde seçtiği banka hesabını tümüyle veritabanından silebilecektir.

4.4. Fatura/Borç Modülü

Kullanıcının faturalarını ve ileri bir tarihte gerçekleşecek giderlerini kaydedebileceği ve daha sonra bu kayıtlar üzerinde düzenleme ve silme gibi işlemler yapabileceği bir modüldür.

4.4.1. Fatura&Borç Ekle: Kullanıcı bu form üzerinden ileri tarihte gerçekleşecek olan giderle ilgili olan etiketi seçecektir. Etiket var olmadığı takdirde etiketi oluşturup, ödemenin gerçekleşmesi gereken tarihi ve diğer alanları da doldurarak veritabanındaki ilgili tablolara fatura&borç kaydını ekleyecektir.

4.4.2. Fatura&Borç Göster: Kullanıcıya kayıtlı Fatura&Borçlar liste şeklinde gösterilecektir. Ayrıca kullanıcı seçtiği kaydı ödeme düğmesine basarak Fatura&Borç tablosundaki kaydı ödendi yapmış olacaktır. Aynı zamanda giderler tablosuna da direk olarak yeni bir gider kaydı yapmış olacaktır.

4.4.3. Fatura&Borç Düzenle – Sil: Bu form üzerinde kullanıcı seçtiği Fatura&Borç kaydının özelliklerini düzenleyebilecektir. Kullanıcı istediği takdirde seçtiği Fatura&Borç kaydını tümüyle veritabanından silebilecektir.

4.5. Etiketler Modülü

Etiketler modülde kullanıcı gelir ve gider türlerinin kaydedebilecektir. Daha sonra bu kayıtlar üzerinde düzenleme ve silme gibi işlemler yapabilecektir.

4.5.1. Etiket Ekle: Bu formda kullanıcı gelir veya gider için yeni bir etiket(kategori) oluşturabilecektir.

4.5.2. Etiketleri Göster: Bu formda iki adet liste olacaktır. Bir liste veritabanında kayıtlı gider türlerini gösterecektir. Diğer liste ise veritabanında kayıtlı gelir türlerini gösterecektir.

4.5.3. Etiketleri Düzenle – Sil: Bu form üzerinde kullanıcı seçtiği etiket kaydının özelliklerini düzenleyebilecektir. Kullanıcı istediği takdirde seçtiği etiket kaydını tümüyle veritabanından silebilecektir.

4.6. Raporlar Modülü

Gelir ve giderlerin kullanıcının seçtiği periyodik şekilde ayrıntılı veya özet olarak gösterilmesiyle gerçekleşecek modüldür.

4.6.1. Özet: Kullanıcının seçtiği periyota (aylık, 3 aylık, yıllık) göre kullanıcıya veritabanındaki gelir ve giderlerin toplamı gösterilecektir. Örnek olarak; yıllık gösterim için veri tabanındaki gelir ve giderlerin gerçekleştikleri yıllara göre toplamını gösteren bir yapı olacaktır.

4.6.2. Ayrıntılı: Kullanıcının seçtiği periyota (aylık, 3 aylık, yıllık) göre kullanıcıya veritabanındaki gelir ve giderlerin hepsi gösterilecektir. Örnek olarak; yıllık gösterim için veri tabanındaki gelir ve giderlerin gerçekleştikleri yıllara göre iki ayrı liste olarak hepsinin gösterilmesiyle gerçekleşecektir.

4.7. Hatırlatmalar Modülü

Bu modül kullanıcının belirlediği koşulların gerçekleşip gerçekleşmediğini kullanıcıya belirtecektir.

4.7.1. Hatırlatma Ekle: Kullanıcı bu formda kendi koşullarını belirleyerek yeni bir hatırlatma kaydı oluşturacaktır.

4.7.2. Hatırlatmaları Göster: Kullanıcı daha önce kaydettiği hatırlatmaları ve bu hatırlatmaların gerçekleşip, gerçekleşmediğini bu formda görecektir.

4.7.3. Hatırlatmaları Düzenle – Sil: Bu form üzerinde kullanıcı seçtiği hatırlatma kaydının özelliklerini düzenleyebilecektir. Kullanıcı istediği takdirde seçtiği hatırlatma kaydını tümüyle veritabanından silebilecektir.

4.8. Yatırımlar Modülü

Kullanıcının kişisel yatırımları bu modülde kaydedilecek. Bu modülde kullanıcı yatırımlarının takibini yapacaktır.

4.8.1. Yatırım Ekle: Kullanıcı gündelik hayatında yaptığı yatırımlarını veya sahip olduğu gayrimenkulleri bu formda kaydedebilecek. Bu kısımda kullanıcıdan yatırımın ilk finansal değerini ve güncel finansal değerini ilgili alanlara girmesi istenmektedir.

4.8.2. Yatırımları Göster: Kullanıcının daha önce yaptığı yatırım kayıtları bu formda listelenecektir. Kullanıcıya yatırımlardan elde ettiği kar – zarar ilişkisi bu formda gösterilecektir.

4.8.3. Yatırımları Düzenle – Sil: Bu form üzerinde kullanıcı seçtiği yatırım kaydının güncel finansal değerini ve diğer özelliklerini düzenleyebilecektir. Kullanıcı istediği takdirde seçtiği yatırım kaydını tümüyle veritabanından silebilecektir.

4.9. Finansal Tasarruflar Modülü

Kullanıcı belirlediği finansal amaç için kendi belirlediği ödeme periyotları ile ne kadar tasarruf etmesi gerektiğini bu modülle hesaplayabilecektir.

4.9.1. Tasarruf Planı Ekle: Bu formda kullanıcı hedeflenen miktarı, ödeme periyodunu ilgili alanlara girerek yeni bir tasarruf planı kaydetmiş olacaktır. Bu kayıt için bir sonuç yine bu formda kullanıcıya gösterilecektir.

4.9.2. Tasarruf Planlarını Göster: Kullanıcı daha önce kaydettiği planların listesi bu formda görebilecektir.

4.9.3. Tasarruf Planlarını Düzenle – Sil: Bu form üzerinde kullanıcı seçtiği tasarruf planı kaydının özelliklerini düzenleyebilecektir. Kullanıcı istediği takdirde seçtiği tasarruf planı kaydını tümüyle veritabanından silebilecektir.

4.10. Online İşlemler Modülü

Kullanıcı bu modülde internet bağlantısını kullanarak döviz kurlarını ve banka hesaplarının güncel durumlarını sisteme aktarabileceklerdir.

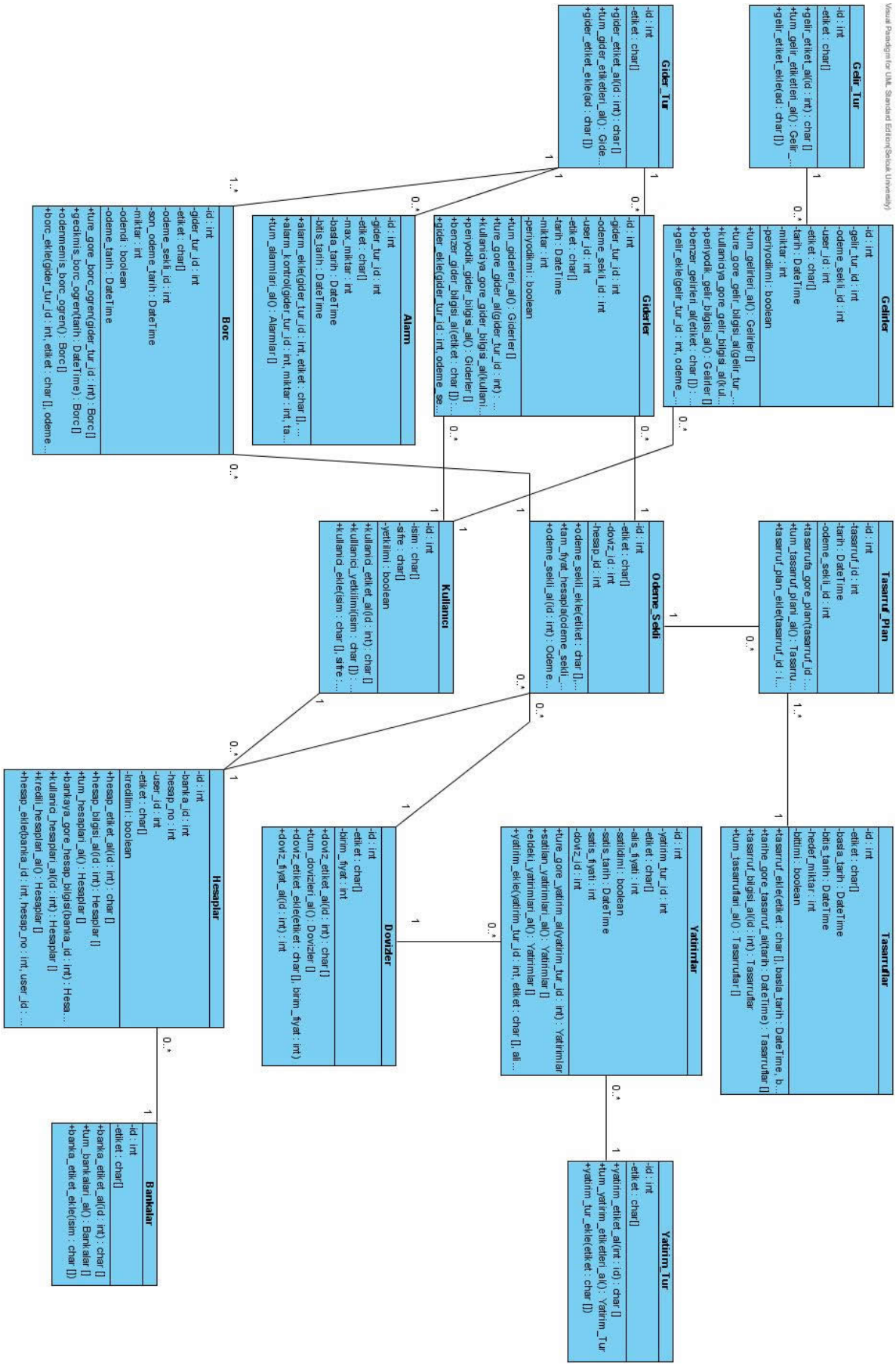
4.10.1. Kur Bilgilerini Güncelle: Günümüzde kullanılan web servisleri kullanılarak veri tabanında kayıtlı olan dövizleri birim fiyatları bu formda güncellenecektir.

4.10.2. Banka Hesabını Güncelle: Bankaların müşterileri için sunduğu dosyalama desteğiyle kullanıcı seçtiği banka hesabının bakiyesini otomatik olarak buradan güncelleyebilecektir.

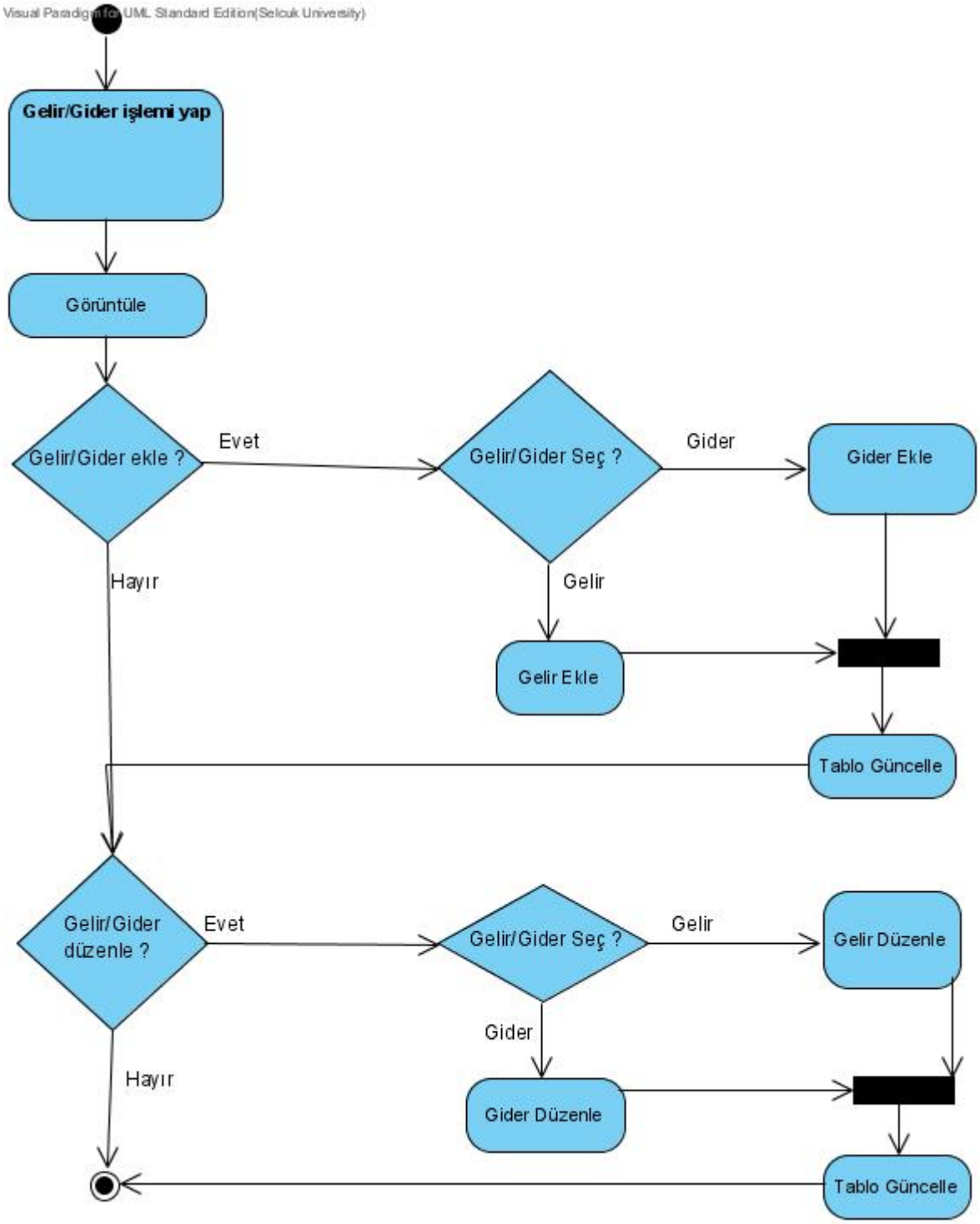
5. UML DİYAGRAMLARI

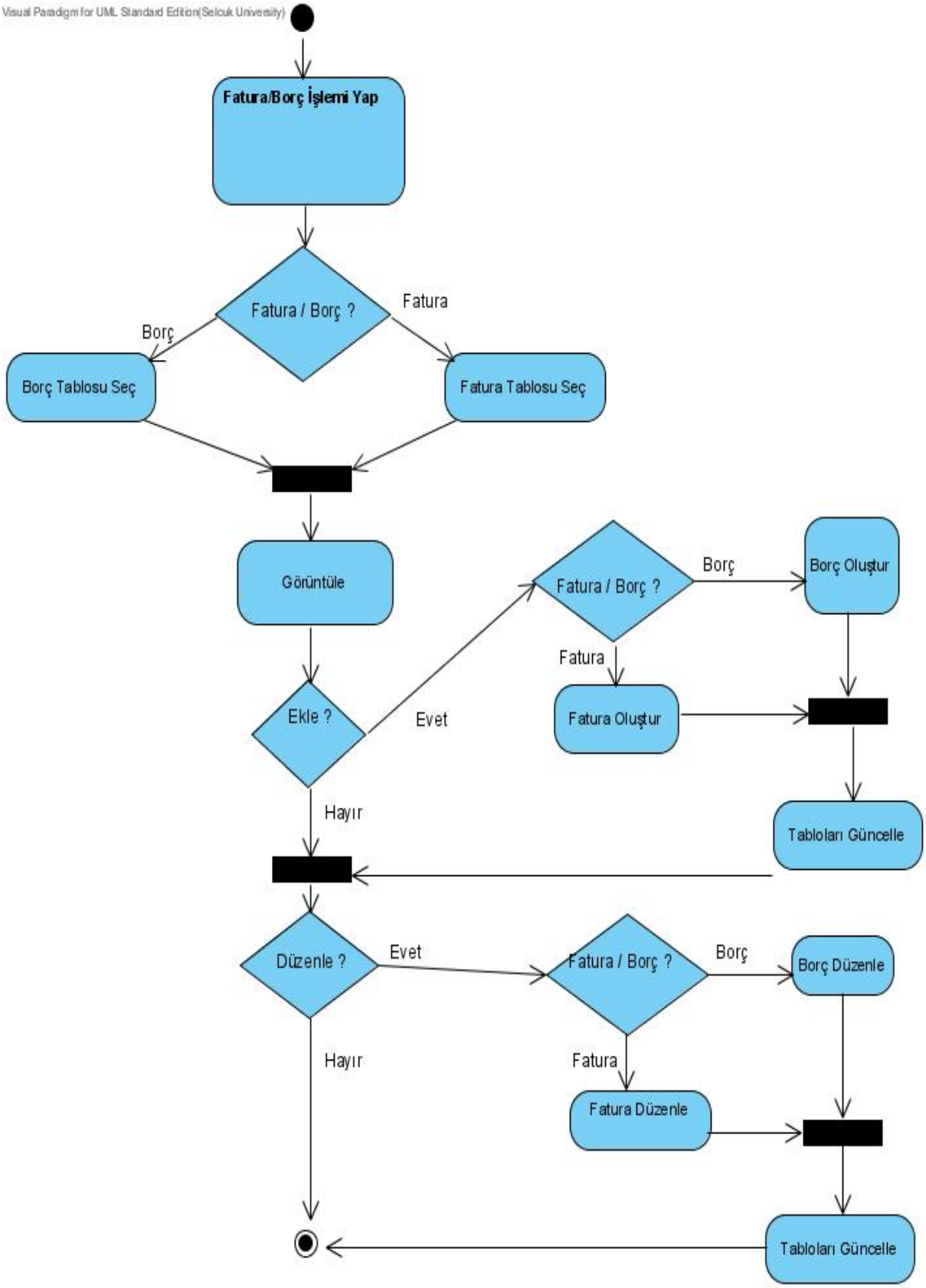
5.1. Use-Case Diyagramı

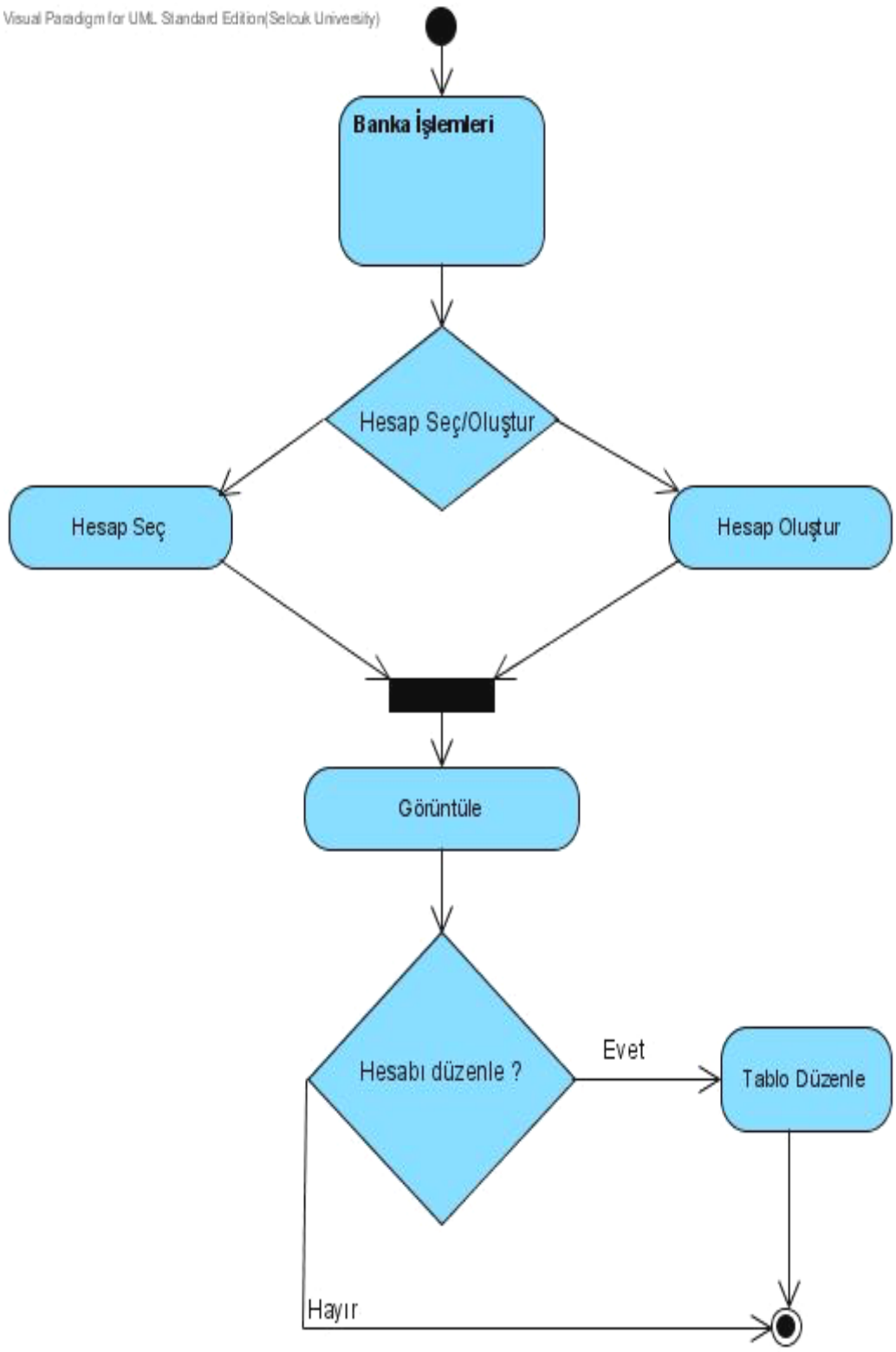
5.2. Class Diyagramı (Burası yok)

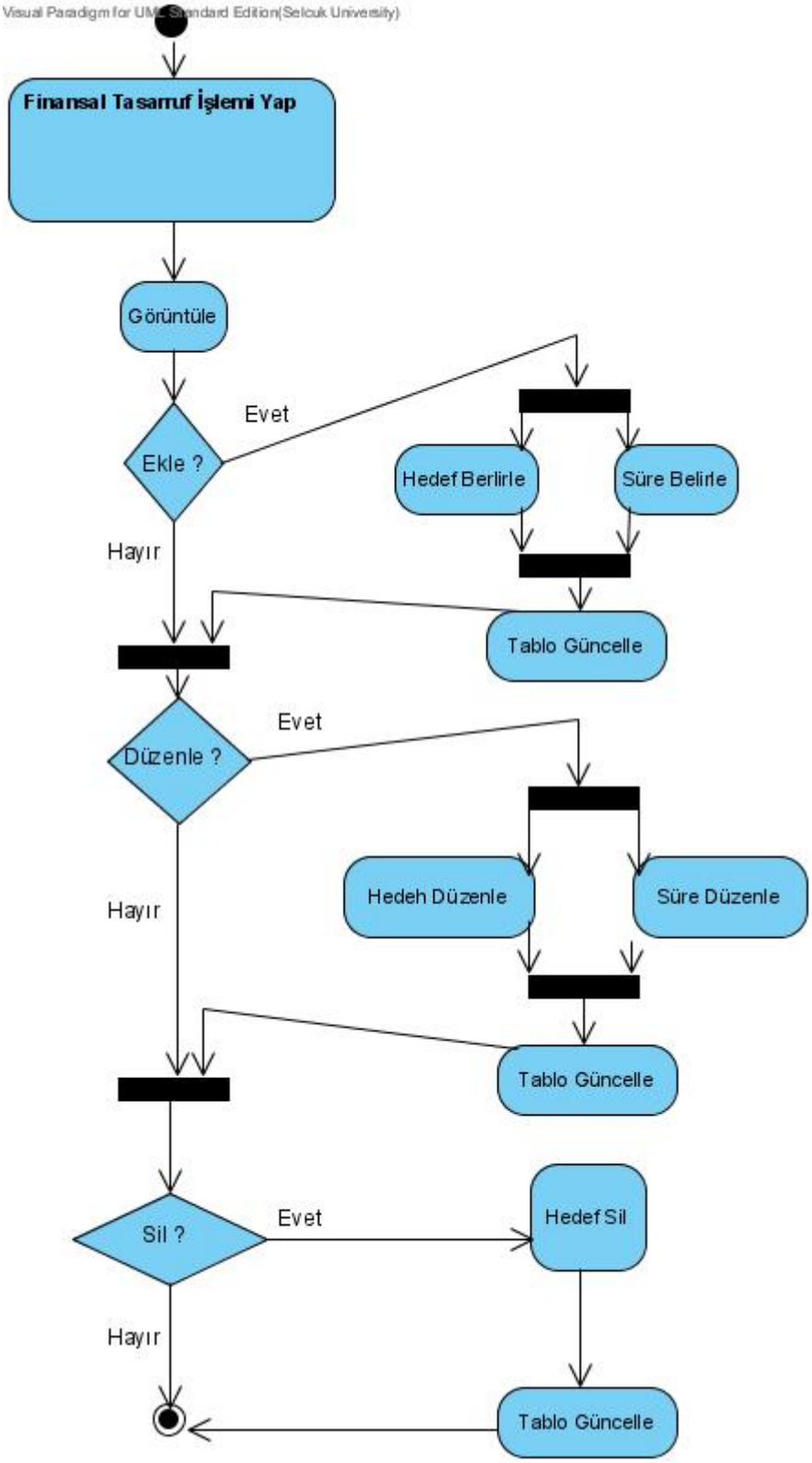


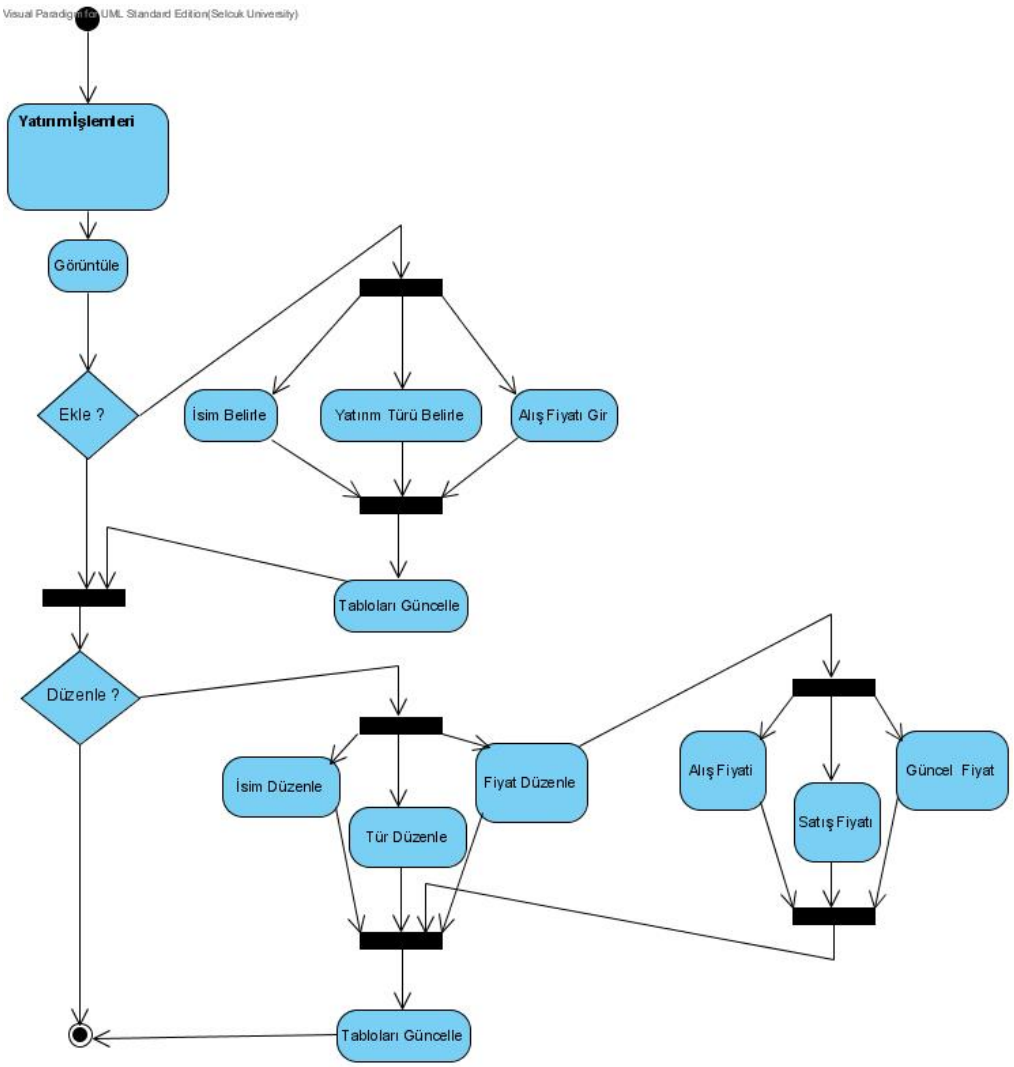
5.3. Activity Diagramları

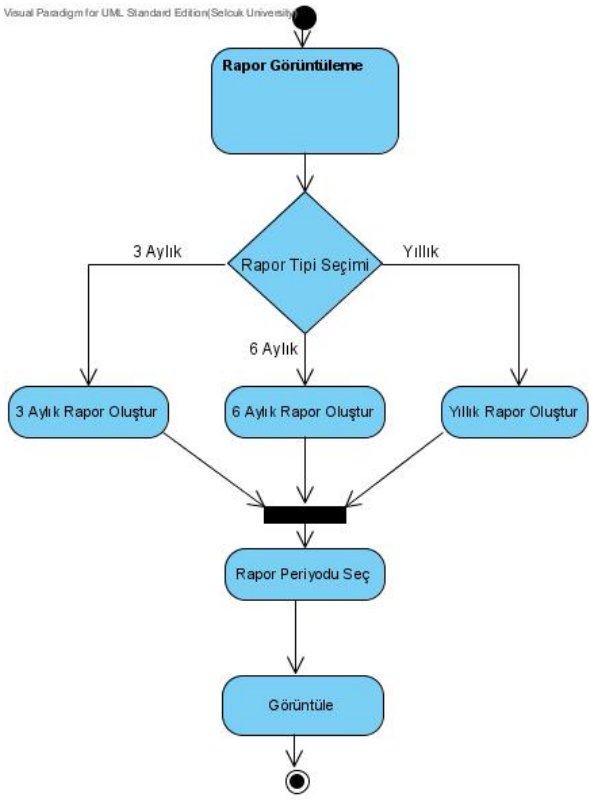



Visual Paradigm for UML Standard Edition(Selcuk University)
Fatura/Borç İşlemi Yap
Fatura / Borç ?
Borç
Fatura
Borç Tablosu Seç
Fatura Tablosu Seç
Görüntüle
Ekle ?
Evet
Hayır
Fatura / Borç ?
Borç
Borç Oluştur
Fatura
Fatura Oluştur
Tabloları Güncelle
Düzenle ?
Evet
Hayır
Fatura / Borç ?
Borç
Borç Düzenle
Fatura
Fatura Düzenle
Tabloları Güncelle

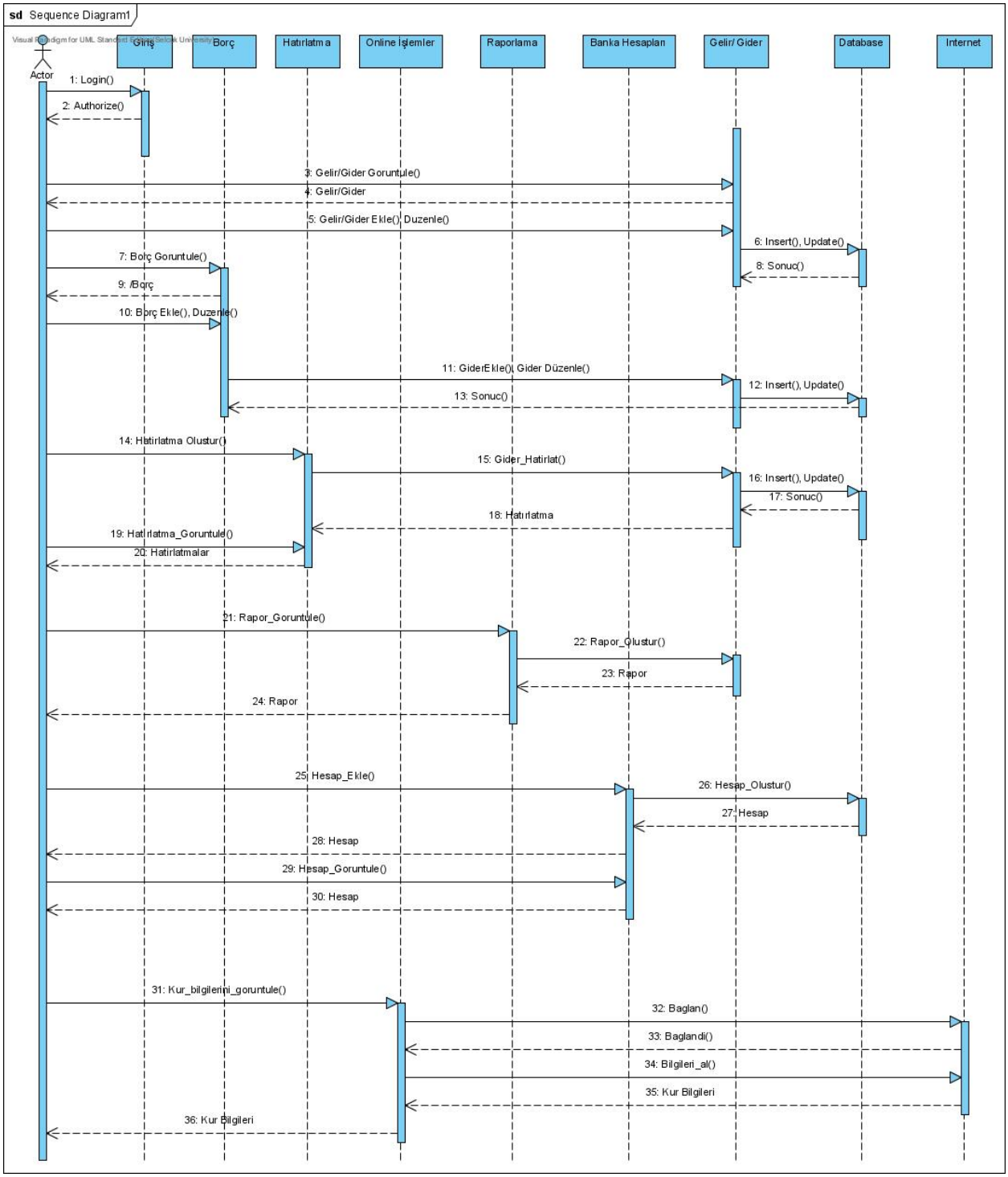

Visual Paradigm for UML Standard Edition(Selcuk University)
Banka İşlemleri
Hesap Seç/Oluştur
Hesap Seç
Hesap Oluştur
Görüntüle
Hesabı düzenle ?
Evet
Tablo Düzenle
Hayır


Visual Paradigm for UML Standard Edition(Selcuk University)
Finansal Tasarruf İşlemi Yap
Görüntüle
Ekle ?
Evet
Hayır
Hedef Berlirle
Süre Belirle
Tablo Güncelle
Düzenle ?
Evet
Hayır
Hedeh Düzenle
Süre Düzenle
Tablo Güncelle
Sil ?
Evet
Hayır
Hedef Sil
Tablo Güncelle


Visual Paradigm for UML Standard Edition(Selcuk University)
Yatırım İşlemleri
Görüntüle
Ekle ?
İsim Belirle
Yatırım Türü Belirle
Alış Fiyatı Gir
Tabloları Güncelle
Düzenle ?
İsim Düzenle
Tür Düzenle
Fiyat Düzenle
Alış Fiyati
Satış Fiyatı
Güncel Fiyat
Tabloları Güncelle


Visual Paradigm for UML Standard Edition(Selcuk University)
Rapor Görüntüleme
Rapor Tipi Seçimi
3 Aylık
6 Aylık
Yıllık
3 Aylık Rapor Oluştur
6 Aylık Rapor Oluştur
Yıllık Rapor Oluştur
Rapor Periyodu Seç
Görüntüle

5.4. Sequence Diyagramları



6. Class İşlemleri burası da yok

6.1. Gelir Tür Sınıfı

```
public class Gelir_Tur
{
    private Gelirler[] Gelirlers;

    private readonly int id;
    private char[] etiket;

    public virtual char[] gelir_etiket_al(int id) {
        return null;
    }
    public virtual Gelir_Tur[] tum_gelir_etiketleri_al() {
        return null;
    }
    public virtual void gelir_etiket_ekle(char[] ad) {

    }

}
```

6.2. Gider Tür Sınıfı

```
public class Gider_Tur
{
    private Giderler[] Giderlers;
    private Alarm[] Alarms;
    private Borc[] Borcs;

    private int id;
    private char[] etiket;

    public virtual char[] gider_etiket_al(int id) {
        return null;
    }
    public virtual Gider_Tur[] tum_gider_etiketleri_al() {
        return null;
    }
    public virtual void gider_etiket_ekle(char[] ad) {

    }

}
```

6.3. Gelir Sınıfı

```
public class Gelirler
{
    private Kullanıcı Kullanıcı;
```

```
private Gelir_Tur Gelir_Tur;

private int id;
private int gelir_tur_id;
private int odeme_sekli_id;
private int user_id;
private char[] etiket;
private DateTime tarih;
private int miktar;
private boolean periyodikmi;

public virtual Gelirler[] tum_gelirleri_al() {
        return null;
}
public virtual Gelirler[] ture_gore_gelir_bilgisi_al(int gelir_tur_id) {

        return null;
}
public virtual Gelirler[] kullaniciya_gore_gelir_bilgisi_al(int kullanici_id) {

        return null;
}
public virtual Gelirler[] periyodik_gelir_bilgisi_al() {
        return null;
}
```

```
        public virtual Gelirler[] benzer_gelirleri_al(char[] etiket) {
            return null;
        }
        public virtual void gelir_ekle(int gelir_tur_id, int odeme_sekli_id, int kullanici_id, char[] etiket, DateTime tarih, int miktar, boolean periyodikmi) {
    }
    }
```

6.4. Gider Sınıfı

```
    public class Giderler
    {
        private Gider_Tur Gider_Tur;
        private Kullanıcı Kullanıcı;
        private Odeme_Sekli Odeme_Sekli;

        private int id;
        private int gider_tur_id;
        private int odeme_sekli_id;
        private int user_id;
        private char[] etiket;
        private DateTime tarih;
        private int miktar;
        private boolean periyodikmi;

        public virtual Giderler[] tum_giderleri_al() {
            return null;
```

```
        }
        public virtual Giderler[] ture_gore_gider_al(int gider_tur_id) {

            return null;
        }
        public virtual Giderler[] kullaniciya_gore_gider_bilgisi_al(int kullanici_id) {

            return null;
        }
        public virtual Giderler[] periyodik_gider_bilgisi_al() {
            return null;
        }
        public virtual Giderler[] benzer_gider_bilgisi_al(char[] etiket) {

            return null;
        }
        public virtual void gider_ekle(int gider_tur_id, int odeme_sekli_id, int user_id,
char[] etiket, DateTime tarih, int miktar, boolean periyodikmi) {

        }

    }
```

6.5. Bankalar Sınıfı

```
    public class Bankalar
    {
```

```
        private Hesaplar[] Hesaplars;

        private int id;
        private char[] etiket;

        public virtual char[] banka_etiket_al(int id) {
                return null;
        }
        public virtual Bankalar[] tum_bankalari_al() {
                return null;
        }
        public virtual void banka_etiket_ekle(char[] isim) {

        }

}
```

6.6. Hesaplar Sınıfı

```
public class Hesaplar
{
        private Kullanıcı Kullanıcı;

        private Bankalar Bankalar;
        private Odeme_Sekli[] Odeme_Seklis;
```

```
private int id;
private int banka_id;
private int hesap_no;
private int user_id;
private char[] etiket;
private boolean kredilimi;

public virtual char[] hesap_etiket_al(int id) {
        return null;
}
public virtual Hesaplar[] hesap_bilgisi_al(int id) {
        return null;
}
public virtual Hesaplar[] tum_hesaplari_al() {
        return null;}
public virtual Hesaplar[] bankaya_gore_hesap_bilgisi(int banka_id) {
return null;
}
public virtual Hesaplar[] kullanici_hesaplari_al(int id) {
        return null;
}
public virtual Hesaplar[] kredili_hesaplari_al() {
        return null;
}
```

```
        public virtual void hesap_ekle(int banka_id, int hesap_no, int user_id, char[] etiket, boolean kredilimi) {
        }
    }
```

6.7. Borç Sınıfı

```
    public class Borc
    {
        private Gider_Tur Gider_Tur;
        private Odeme_Sekli Odeme_Sekli;

        private int id;
        private int gider_tur_id;
        private char[] etiket;
        private int odeme_sekli_id;
        private DateTime son_odeme_tarih;
        private int miktar;
        private boolean odendi;
        private DateTime odeme_tarih;

        public virtual Borc[] ture_gore_borc_ogren(int gider_tur_id) {

            return null;
        }
        public virtual Borc[] gecikmis_borc_ogren(DateTime tarih) {
```

```
                return null;
        }
        public virtual Borc[] odenmemis_borc_ogren() {
                return null;
        }
        public virtual void borc_ekle(int gider_tur_id, char[] etiket, int odeme_sekli_id,
DateTime son_odeme_tarih, boolean odendimi, DateTime odeme_tarih) {
                }
}
```

6.8. Yatırımlar ve Yatırım Türleri Sınıfı

```
public class Yatirim_Tur
{
        private Yatirimlar[] Yatirimlars;
        private int id;
        private char[] etiket;
        public virtual char[] yatirim_etiket_al(id int) {
                return null;
        }
        public virtual Yatirim_Tur tum_yatirim_etiketleri_al() {
                return null;
        }
        public virtual void yatirim_tur_ekle(char[] etiket) {
        }
}
```

```
public class Yatirimlar
{
    private Yatirim_Tur Yatirim_Tur;
    private Dovizler Dovizler;
    private int id;
    private int yatirim_tur_id;
    private char[] etiket;
    private int alis_fiyati;
    private boolean satildimi;
    private DateTime satis_tarih;
    private int satis_fiyati;
    private int doviz_id;
    public virtual Yatirimlar ture_gore_yatirim_al(int yatirim_tur_id) {
    return null;
    }
    public virtual Yatirimlar[] satilan_yatirimlari_al() {
        return null; }
    public virtual Yatirimlar[] eldeki_yatirimlari_al() {
        return null;}
    public virtual void yatirim_ekle(int yatirim_tur_id, char[] etiket, int alis_fiyati,
boolean satildimi, DateTime satis_tarih, int satis_fiyati, int doviz_id) {
    }
}
```

6.9. Tasarruflar ve Tasarruf Planları Sınıfları

```
public class Tasarruflar
{
    private Tasarruf_Plan[] Tasarruf_Plans;

    private int id;
    private char[] etiket;
    private DateTime basla_tarih;
    private DateTime bitis_tarih;
    private int hedef_miktar;
    private boolean bittimi;

    public virtual void tasarruf_ekle(char[] etiket, DateTime basla_tarih, DateTime
bitis_tarih, int hedef_miktar, boolean bittimi) {

    }
    public virtual Tasarruflar[] tarihe_gore_tasarruf_al(DateTime tarih) {

        return null;
    }
    public virtual Tasarruflar tasarruf_bilgisi_al(int id) {
        return null;
    }
```

```
        public virtual Tasarruflar[] tum_tasarruflari_al() {
            return null;     }}
    public class Tasarruf_Plan
    {
        private Odeme_Sekli Odeme_Sekli;
        private Tasarruflar Tasarruflar;

        private int id;
        private int tasarruf_id;
        private DateTime tarih;
        private int odeme_sekli_id;

        public virtual Tasarruf_Plan[] tasarrufa_gore_plan(int tasarruf_id) {

            return null;
        }
        public virtual Tasarruf_Plan[] tum_tasarruf_plani_al() {
            return null;
        }
        public virtual void tasarruf_plan_ekle(int tasarruf_id, DateTime tarih, int
odeme_sekli_id) {

        }

    }
```

6.10. Alarm Sınıfı

```
public class Alarm
{
    private Gider_Tur Gider_Tur;

    private int id;
    private int gider_tur_id;
    private char[] etiket;
    private int max_miktar;
    private DateTime basla_tarih;
    private DateTime bitis_tarih;

    public virtual void alarm_ekle(int gider_tur_id, char[] etiket, int max_miktar,
DateTime basla_tarih, DateTime bitis_tarih) {

    }
    public virtual boolean alarm_kontrol(int gider_tur_id, int miktar, DateTime
tarih) {
        return false;
    }
    public virtual Alarmlar[] tum_alarmlari_al() {
        return null;
    }
```

```
}
```

6.11. Ödeme Şekli Sınıfı

```
public class Odeme_Sekli
{
    private Hesaplar Hesaplar;

    private Tasarruf_Plan[] Tasarruf_Plans;
    private Dovizler Dovizler;
    private Borc[] Borcs;
    private Giderler[] Giderlers;

    private int id;
    private char[] etiket;
    private int doviz_id;
    private int hesap_id;

    public virtual void odeme_sekli_ekle(char[] etiket, int doviz_id, int hesap_id) {

}
    public virtual int tam_fiyat_hesapla(int odeme_sekli_id, int miktar) {
    return null;
    }
    public virtual Odeme_Sekli odeme_sekli_al(int id) {
```

```
            return null;}}
```

6.12. Kullanıcılar Sınıfı

```
using System;


public class Kullanıcı
{
    private Gelirler[] Gelirlers;
    private Giderler[] Giderlers;
    private Hesaplar[] Hesaplars;


    private int id;
    private char[] isim;
    private char[] sifre;
    /// <summary>
    /// Sadece 1 kullanıcı yetkili olabilir. Yetkisiz kullanıcılar ailenin diğer üyeleri gibi düşünülebilir.
    /// </summary>
    private boolean yetkilimi;


    public virtual char[] kullanici_etiket_al(int id) {
        return null;
    }
    public virtual boolean kulllanici_yetkilimi(char[] isim) {
        return false;
```

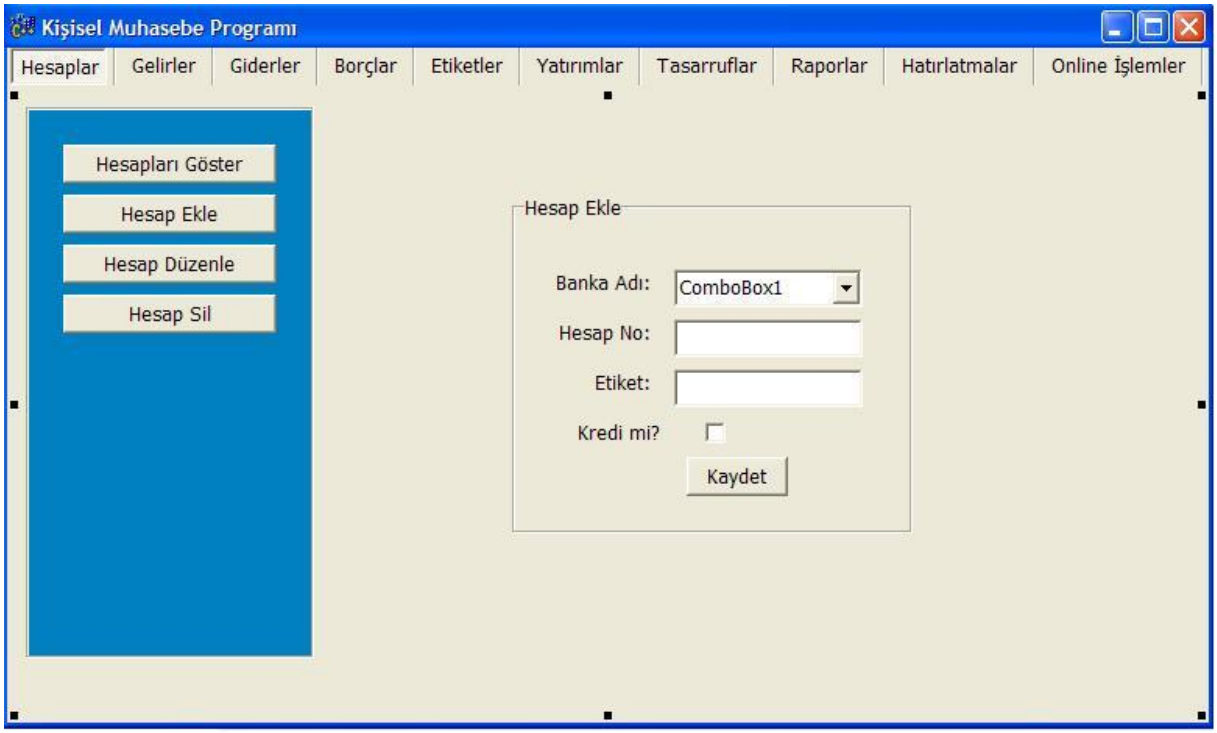
```
}public virtual void kullanici_ekle(char[] isim, char[] sifre, boolean auth) {
}}
```

6.13. Dövizler Sınıfı

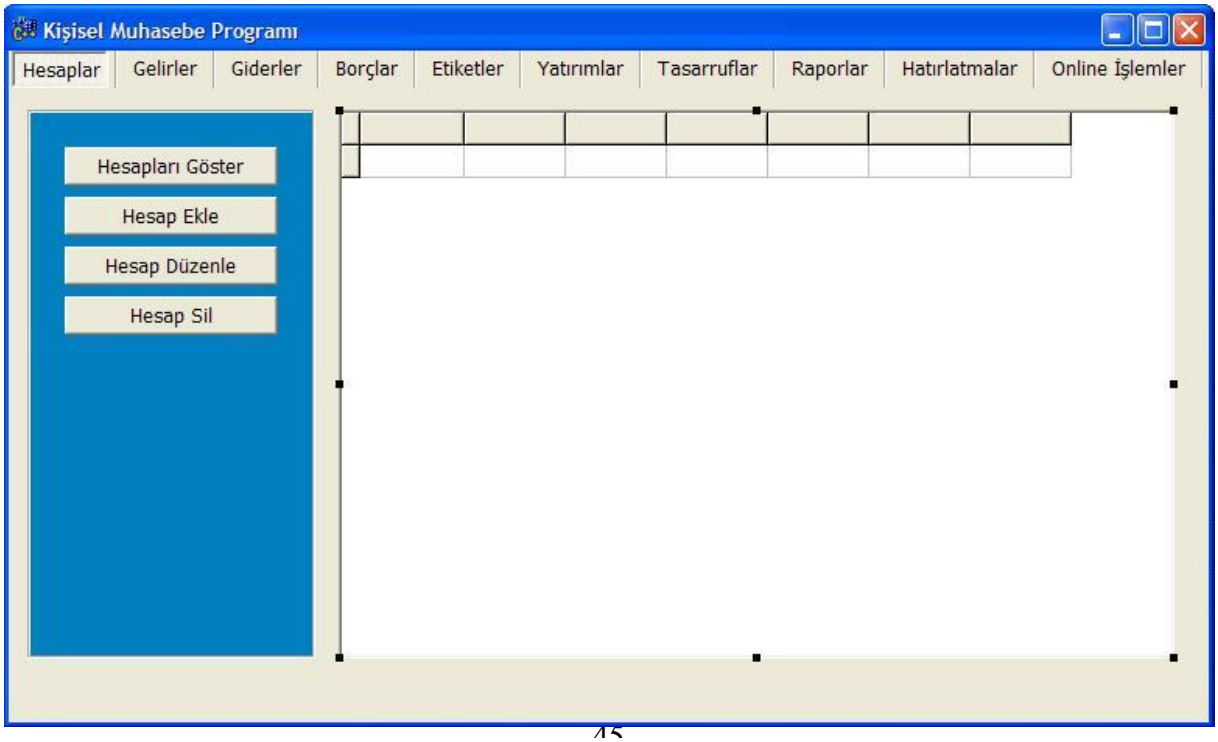
```
public class Dovizler
{
    private Odeme_Sekli[] Odeme_Seklis;
    private Yatirimlar[] Yatirimlars;

    private int id;
    private char[] etiket;
    private int birim_fiyat;

    public virtual char[] doviz_etiket_al(int id) {
        return null;
    }
    public virtual Dovizler[] tum_dovizleri_al() {
        return null;
    }
    public virtual void doviz_etiket_ekle(char[] etiket, int birim_fiyat) {

}
    public virtual int doviz_fiyat_al(int id) {
        return null;    }}
```

7. Veri tabanı Create Kodları. (İsteniyor)

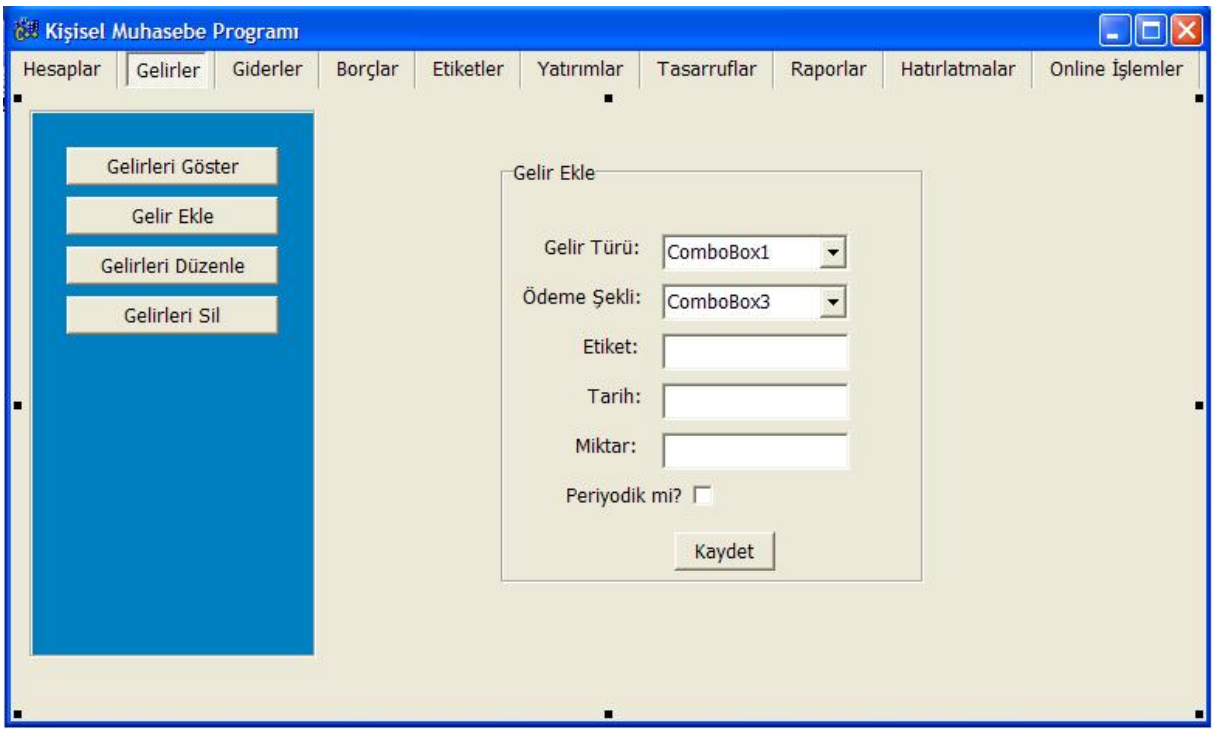
8. EKRAN ÇIKTILARI



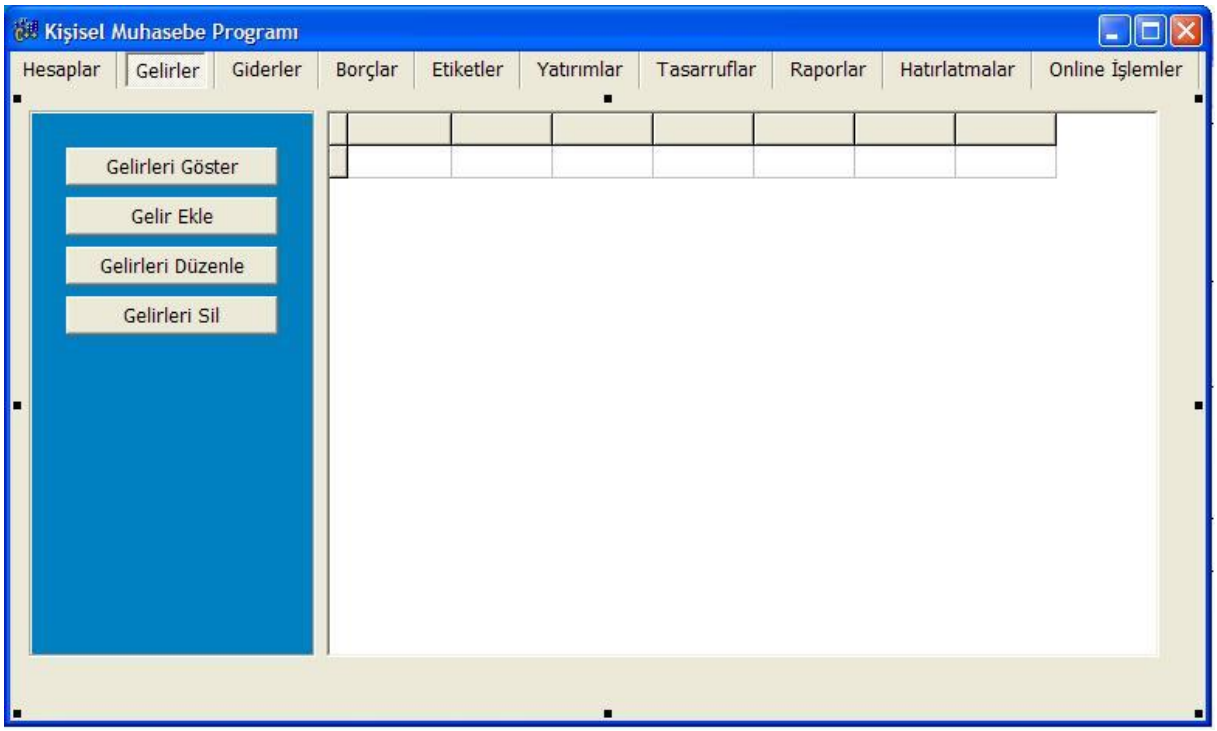
Şekil 4. Kullanıcının yeni hesaplar ekleyebileceği ara yüz.



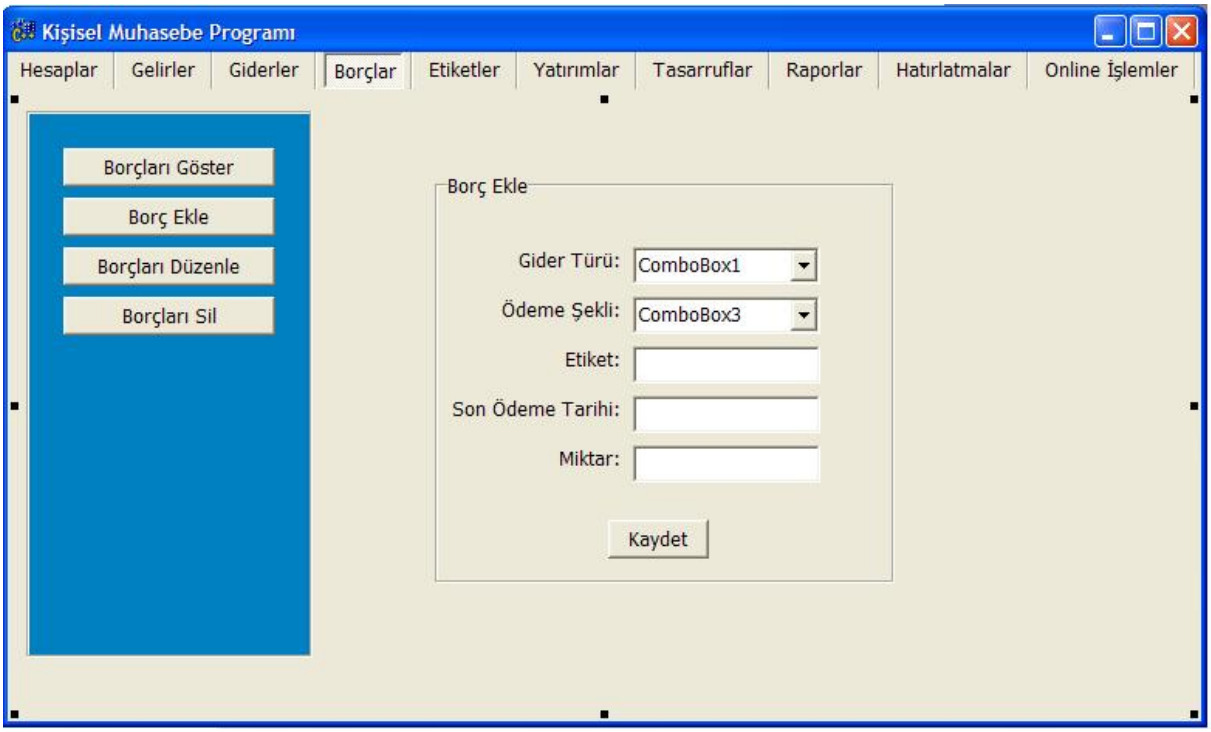
Şekil 5. Kullanıcının kayıtlı hesaplarını görebileceği ara yüz.



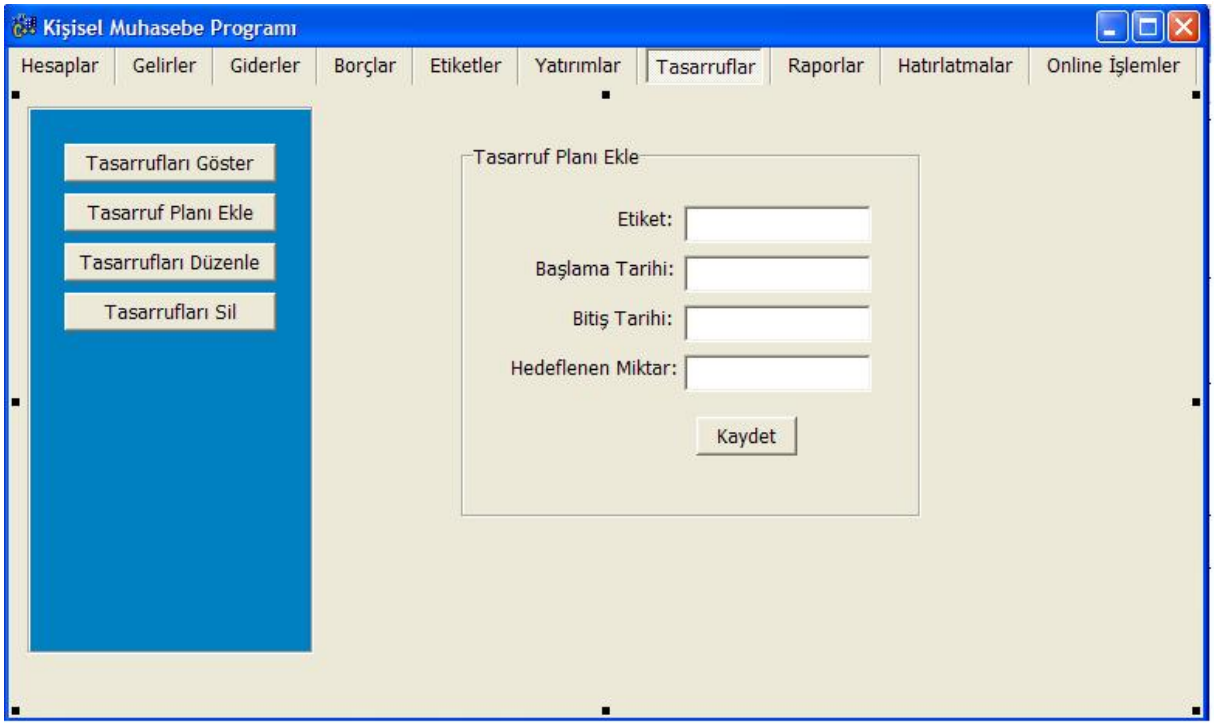
Şekil *6. Yeni gelir* eklemek *için kullanılan ara yüz.*



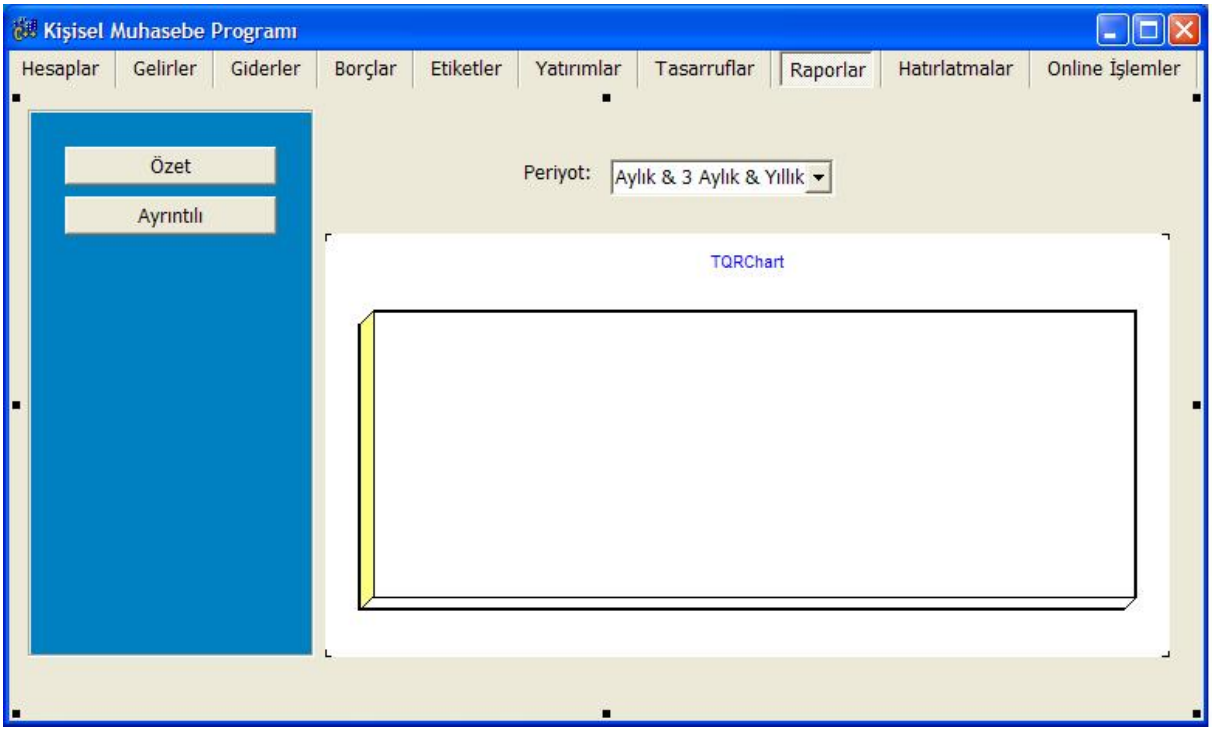
Şekil 7. Kullanıcın kayıtlı gelirlerinin gösterildiği ara yüz.



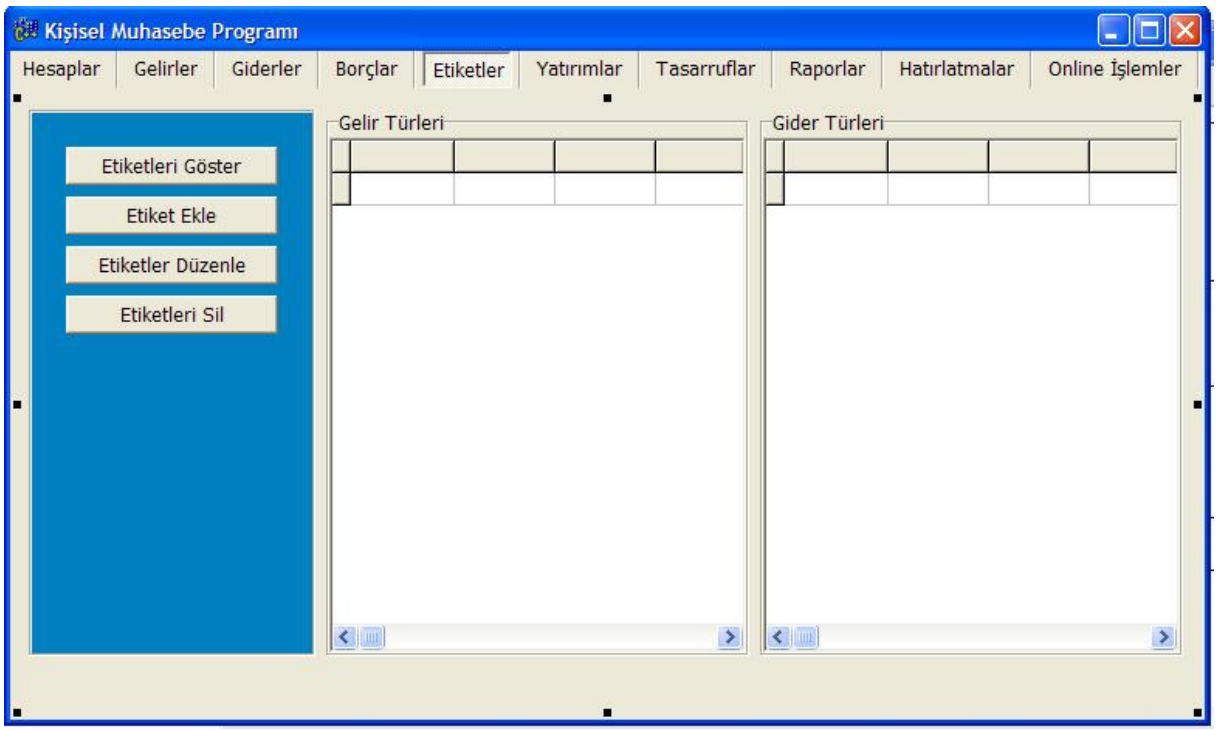
Şekil 8. Kullanıcının yeni borçlar oluşturabileceği ara yüz.



Şekil 9. Kullanıcının tasarruf planları ekleyebileceği ara yüz.



Şekil 10. Kullanıcının aylık, 3 aylık ve yıllık raporlarını ayrıntılı veya özet bir şekilde, grafik çıktı olarak görüntüleyebileceği ara yüz.



Şekil 11. Kullanıcının gelir ve gider türleri için oluşturduğu etiketleri görüntüleyebileceği ara yüz.

8. GEÇERLEME VE BAKIM PLANI

Mevcut yazılım, herhangi bir zaman dilimi içinde, program üzerinde güncel, diğer programlara ve platformlara uyumlu hale getirilecek düzeyde olmalıdır. Böylece program güncelliğini kaybetmeyecektir. Tasarlanan program kodunun geçerleme değerleri veya herhangi bir verisinde değişikliği olduğunda kolayca düzeltilip doğru ve kararlı bir şekilde çalışması sağlanmalıdır. Bu sayede gelen isteklere hızlı bir şekilde cevap verilebilecektir.