T.C.
BEYKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
MATEMATİK BİLGİSAYAR ANABİLİM DALI
BİLGİ TEKNOLOJİLERİ BİLİM DALI

SILVERLIGHT TABANLI BİR İNTERNET BANKA ŞUBESİ UYGULAMASI

(Yüksek Lisans Tezi)

Tezi Hazırlayan : **Barış BALDAN**

İstanbul, 2011

T.C.
BEYKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
MATEMATİK BİLGİSAYAR ANABİLİM DALI
BİLGİ TEKNOLOJİLERİ BİLİM DALI

SILVERLIGHT TABANLI BİR İNTERNET BANKA ŞUBESİ UYGULAMASI

(Yüksek Lisans Tezi)

Tezi Hazırlayan : **Barış BALDAN**
Öğrenci No : 080862001

Danışman :
Yrd.Doç.Dr. Turhan KARAGÜLER

İstanbul, 2011

YEMİN METNİ

Yüksek Lisans Tezi olarak sunduğum “Silverlight Tabanlı Bir İnternet Banka Şubesi Uygulaması” başlıklı bu çalışmanın, bilimsel ahlak ve geleneklere uygun şekilde tarafımdan yazıldığını, yararlandığım eserlerin tamamının kaynaklarda gösterildiğini ve çalışmamın içinde kullanıldıkları her yerde bunlara atıf yapıldığını belirtir ve bunu onurumla doğrularım. 12/07/2011

Aday: Barış BALDAN

T.C.
BEYKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZ SAVUNMA SINAVI SONUÇ TUTANAĞI

Beykent Üniversitesi
Fen Bilimleri Enstitüsü Müdürlüğü'ne,

Aşağıda tez adı belirtilen yüksek lisans öğrencisi ....0808.62001..... no'lu Baris.....Baldan.....'ın 25/8/2011 tarihinde yapılan tez savunma sınavı[1] sonucunda ...70.....dakika süreyle sunduğu ve savunduğu tezi hakkında[2] oybirliğiyle/oyçokluğuyla, Kabul/Red/Düzeltme(.....ay içinde) kararı verilmiştir.

Bilgilerinize saygılarımızla arz ederiz.

Anabilim Dalı : Matematik - Bilgisayar
Programı : Bilgi Teknolojileri
Tez Başlığı[3] : Silverlight Tabanlı Bir İnternet Banka Şubesi Uygulaması

Tez Sınav Jürisi	**Öğretim Üyesi**	**İmza**
Danışman	: Yrd. Doç. Dr. Turhan Karagöz	
Üye	: Yrd. Doç. Dr. Gökhan Silahtaroğlu	
Üye	: Prof. Dr. Esat Hamzaoğlu	

[1] Jüri üyeleri söz konusu tezin kendilerine teslim edildiği tarihten itibaren en geç bir ay içinde toplanarak öğrenciyi tez savunma sınavına alır. Belirlenen günde yapılamayan jüri toplantısı, katılanların hazırladığı bir tutanakla enstitü yönetimine bildirilir. Bu durumda jüri en geç onbeş gün içinde toplanarak adayı tez savunma sınavına alır. Tez savunma sınav süresi en az 45 dakikadır. Yüksek lisans tez savunma sınavı, tez çalışmasının sunulması ve bunu izleyen soru-yanıt bölümlerinden oluşur ve dinleyiciye açıktır. (Beykent Lisansüstü eğitim ve Öğretim Yönetmeliği-Madde30-3)

[2] Tez sınavının tamamlanmasından sonra jüri, tez hakkında "kabul", "düzeltme" veya "red" kararı verir. Jüri başkanı, jüri üyelerince imzalanmış sınav tutanağını, tez sınavını izleyen üç gün içinde ilgili enstitü yönetimine teslim eder. Tezi başarısız bulunan öğrencinin Enstitü ile ilişiği kesilir. Tezi hakkında düzeltme kararı verilen öğrenci en geç üç ay içinde gerekli düzeltmeleri yaparak ve yönetmelikte belirtilen usullere uygun olarak tezini aynı jüri önünde yeniden savunur. Bu savunma sınavında da tezi kabul edilmeyen öğrencinin enstitü ile ilişiği kesilir. (Beykent Lisansüstü eğitim ve Öğretim Yönetmeliği-Madde30-4)

[3] İleride doğabilecek aksaklıkların engellenmesi için tezin başlığının yazılması gerekmektedir.

SILVERLIGHT TABANLI BİR İNTERNET BANKA ŞUBESİ UYGULAMASI

Tezi Hazırlayan : **Barış BALDAN**

Özet

Günümüzde bankacılık işlemlerinin çoğu elektronik bankacılık hizmetleri ile gerçekleştirilmektedir. Elektronik bankacılık sayesinde banka müşterileri şubelere gitmeden işlemlerini her yerden, istedikleri zamanda yapabilmektedirler. Bankalar müşterilerine daha iyi hizmet verebilmek için İnternet Şube uygulamalarını geliştirmektedirler. Görsel yönü gelişmiş zengin içeriğe sahip internet şube uygulamaları geliştirmek için son yıllara kadar Flash ve Flesk gibi Adobe uygulamaları kullanılmaktaydı. Oysa web uygulamaları artık masaüstü uygulamalarına benzemeye başlamıştır. Masaüstü programların çalışma rahatlığıyla web üzerinden çalışabilmek yine Silverlight gibi teknolojilerle mümkün olmuştur.

Bu çalışma da Silverlight teknolojisi kullanılarak bir internet banka şubesinin görsel olarak nasıl kolay ve hızlı bir biçimde zenginleştirilebileceği ve intenet şubenin nasıl daha verimli hale getirilebileceği tez kapsamında geliştirilen bir yazılım aracılığıyla gösterilmiştir.

Anahtar Kelimeler: İnternet Bankacılığı, İnternet Şube, Elektronik Bankacılık, Silverlight, RIA(Rich Internet Applications)

AN EXAMPLE OF DEVELOPED INTERNET BANKING APPLICATION BY USING SILVERLIGHT

Presented by: **Barış BALDAN**

Abstract

Today, most banking transactions are mainly carried out by mean of electronic banking services. Bank customers from anywhere and at any time without going to a bank, can make their transactions by using electronic banking. Advanced visual aspect to develop web applications with rich content until now, used in Adobe applications such as Flash and Fles. But web applications are now becoming indistinguishable from the desktop applications. Convenience of using desktop programs is now possible for web applications by using new technologies such as Silverlight.

In this study, how improve an internet banking service by employing Silverlight technology is examined via a developed program named as *beykentbank*.

Key Words: Internet Banking, Internet Branch Office, Electronic Banking, Silverlight, RIA (Rich Internet Applications)

İÇİNDEKİLER

Sayfa

ŞEKİLLER LİSTESİ

KISALTMALAR

3D	**:**	Three Dimension
CAPTCHA	**:**	Completely Automated Public Turing test to tell Computers and Humans Apart
CLR	**:**	Common Language Runtime
EFT	**:**	Elektronik Fon Transferi
HTML	**:**	Hyper Text Markup Language
IDE	**:**	Integrated Development Environment
IP	**:**	Internet Protocol Address
OGS	**:**	Otomatik Geçiş Sistemi
RIA	**:**	Rich Internet Application
SQL	**:**	Structured Query Language
SSMS	**:**	SQL Server Management Studio
SHA1	**:**	Secure Hash Algorithm 1
SOA	**:**	Service Oriented Architecture
SP	**:**	Stored Procedure

WPF	**:**	Windows Presentation Foundation
WPF/E	**:**	Windows Presentation Foundation/Everywhere
WCF	**:**	Windows Communication Foundation
WS	:	Web Service
VB	**:**	Visual Basic
VS	**:**	Visual Studio

1. GİRİŞ

Bu tezde bir banka İnternet Şubesi'nin Silverlight ile geliştirilmesi amaçlanmıştır. Uygulamanın hem görsel olarak zengin olması hemde mevcuttaki uygulamalardan daha hızlı olması ve fonksiyonel kullanımı hedeflenmiştir.

Bilgi teknolojileri alanındaki gelişmeler, şüphesiz bankacılık sektörünün değişiminede neden olacaktır. Elektronik bankacılık, bu dönüşümün önemli bir parçasıdır. Elektronik bankacılık geleceğin değişim teknolojisi olup, internet, telefon veya diğer elektronik dağıtım kanalları (Nsouli ve Schaechter, 2002) yoluyla, işlemlerin kolaylığı ve maliyeti açısından tüketiciye büyük faydalar sağlayacaktır. İnternet bankacılığı, bankacılık işlemleri yapmak için her zaman ve her yerde daha hızlı işlem yapma imkanı sağlamakta ve geleneksel banka şubesine göre daha düşük işlem ücretlerini mümkün kılmaktadır. İnternet bankacılığı, bankalara müşteri hizmetleri kalitesinin artırılması ve operasyonel maliyetlerin azaltılması gibi rekabet avantajı sağlar [1].

Küresel rekabet kavramı ilk kez 1980'lerin sonlarında Türkiye'de kendini özel bankacılık hizmetleri ve ATM (Otomatik Para Çekme Makineleri) de göstermiştir. Türkiye, Telefon Bankacılığı'na 1995 yılında geçti ve rekabet İnternet bankacılığı nedeniyle artmaktadır. İnternet abonelik sistemi ve internet üzerinden alışveriş alışkanlığı ise 2000'li yılların başında başlamıştır [2].

Adobe firmasının Flash teknolojisi yerine Microsoft firmasının Silverlight teknolojisinin tercih nedeni, Silverlight'ın Visual Studio ile entegre oluşu ve programlama dillerini kullanıyor olmasıdır. Şimdiye kadar hiçbir bankanın mevcut internet şubesi Silverlight teknolojisi ile yapılmamıştır. Eğer Flash teknolojisi tercih edilmiş olsaydı, Adobe Flash teknolojisinin uygulama arayüzü kullanılmalı ayrıca ActionScript dilinin öğrenilmesi ve geliştirilen uygulamanın(application) manuel olarak sayfaya entegre edilmesi gerekirdi. Oysa Silverlight uygulaması kullanıldığında, sadece Visual Studio ortamı tüm yazılım işlemleri için yeterli olmaktadır. Proje oluşturulduğunda Visual Studio Silverlight uygulamasını doğrudan sayfaya entegre etmekte ve uygulamayı geliştirmek için başka bir dil öğrenmeye

gerek bırakmamaktadır. Örneğin sadece C# veya VB gibi yazılım dillerinden bir tanesinin bilinmesi yeterlidir.

Günümüzde gelişen internet kullanım alışkanlıkları web sayfalarını durağan bir görüntüden çıkararak yüksek işlevsellik ve kullanım kolaylığını bir arada bulundurma ihtiyacını getirmiştir. Yaygın olarak sunucu merkezli uygulamalar yazılmaktadır. Yani web sayfasında bir düğmeye basma işlemi yapıldığında bilgiler sunucuya gönderilir ve işlendikten sonra geri getirilir. Oysa görsellik, hızlı çalışan uygulamalar ve animasyonlar hazırlamak gibi amaçlar olursa uygulamanın sunucu tarafında değil tarayıcı tarafında çalışması gerekir. Bu da Java Applet, AJAX, Adobe Flash ve Microsoft Silverlight uygulamaları ile sağlanabilmektedir.

Rich Internet Applications yani Zengin İnternet Uygulamaları deyiminin kısa yazımı olan RIA son zamanlarda geliştiriciler tarafından oldukça sık duyurulan bir terimdir. RIA, genel olarak internet uygulamalarının daha zengin bir arayüz ile sunulmasını sağlayan uygulamalar olarak adlandırılır. RIA kavramı Adobe firması tarafından 1990 yılında ortaya atılmıştır. Günümüzde bir çok RIA uygulama geliştirme ortamı mevcuttur. Bunlardan en bilineni eskiden Macromedia'nın şimdi ise Adobe firmasının Flash merkezli uygulamalarıdır.

Günümüze kadar RIA daha çok interaktif web uygulamalarında sıkça ihtiyaç duyulan bir araç olmuştur. Çok kısa bir süre sonra RIA tabanlı web uygulamalarının hemen hemen bütün web uygulamalarının bir ihtiyacı olacağı uzmanlarca belirtilmektedir. Web arayüzünde basit HTML ile sınırlı kalmadan masaüstündeki zengin görsel öğeleri programatik olarak kullanabileceğiniz her tür araç RIA konseptini uygulamaktadır. Bu tez çalışmasında kullanılan Microsoft Silverlight teknolojiside en bilinen RIA uygulamalarındandır [3].

Örnek sayfaya http://www.beykentbank.com/ adresinden ulaşılabilir. Sayfanın ve Web Servislerin programlanmasında C# dili kullanılmıştır. Veritabanı(Database) olarak SQL Server 2008 kullanılmıştır. Uygulama geliştirme ortamı ise Visual Studio 2010 dur.

2. SİLVERLİGHT YAZILIM PLATFORMU

Silverlight Online yada Ofline Web, masaüstü ve mobil uygulamalar için çekici, interaktif kullanıcı deneyimleri oluşturmak için güçlü bir geliştirme platformudur [4].

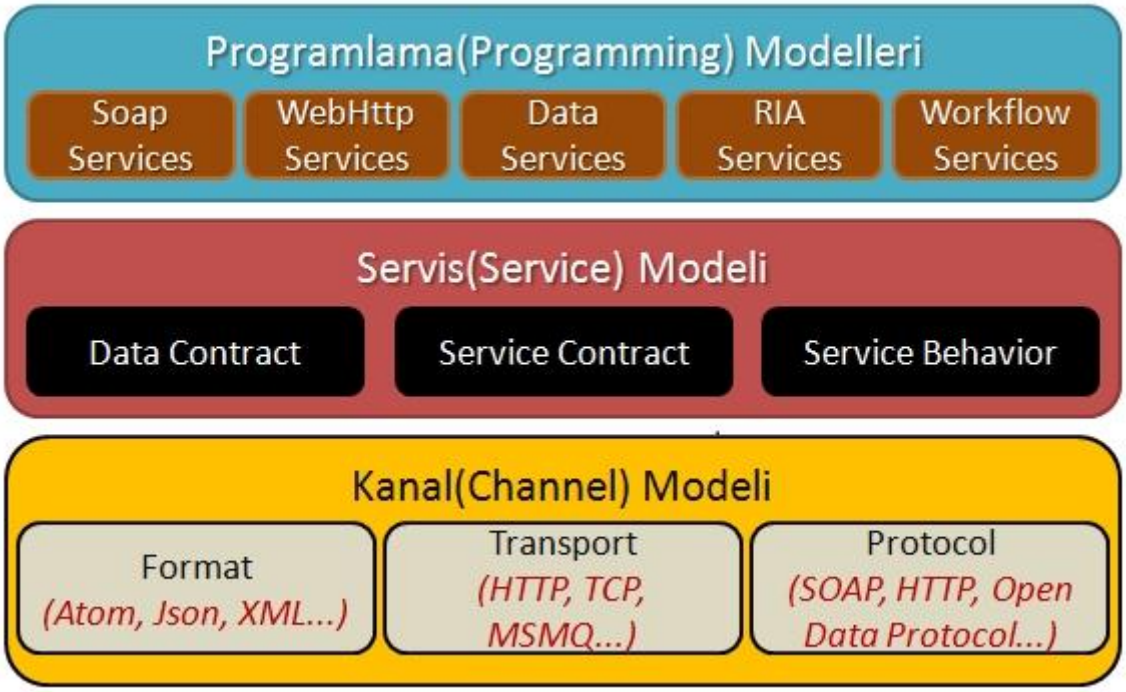
Microsoft Silverlight, ağ uygulamaları için animasyon, vektör, 3D grafik ve görüntü oynatma imkânları sağlayan zengin internet uygulamaları geliştirme düzlemidir. Silverlight, WPF türevi görsel yazılımlama tekniğiyle çokluortam, çizge (grafik), animasyon ve interaktif uygulamaların tek eklenti üzerinden yürütülmesini sağlar [5]. Esas amacı ağ tabanlı zengin interaktif uygulamaların hazırlanabilmesi için bir teknoloji düzlemi oluşturmak olan Silverlight özellikle Microsoft'un .NET tabanlı dillerine desteği ile dikkat çekmektedir. .NET Framework ile beraber gelen CLR'ın bir kısmını taşıyan Silverlight böylece istemci tarafında düzlem bağımsız olarak MultiThread ve MultiCore desteğine sahiptir [6].

Silverlight'ın alt yapısı Microsoft'un .Net teknolojisidir [7]. İnternet tarayıcısı üzerinden çalışan, yüksek görsellikli, hızlı ve gelişmiş web uygulamaları gerçekleştirmeyi sağlayan bir teknolojidir. Silverlight, Adobe Flash ve Ajax bileşenleri ile rekabeti amaçlamaktadır. Aynı zamanda, Silverlight yakın zamanda Sun Microsystems' ın JavaFX'i ile yarışması beklenmektdir [8].

Silverlight bu ismiyle lanse edilmeden önce WPF/E (Windows Presentation Foundation/Everywhere) ismiyle anılmaktaydı. WPF çalışmak için bir Windows işletim sistemi ve .NET Framework isterken, Silverlight bir browser plug-in'i sayesinde Windows, Linux ve hatta MacOs ayırt etmeden heryerde aynı şekilde çalışabilmektedir. Buna "varolan standartlara ayak uydurmaktansa, kendi standartını getirmeyi istemek" diye bilinen klasik Microsoft anlayışı diyebiliriz.

Silverlight bütün tarayıcılarda kullanılabilir. Bunun yapılabilmesi için Silverlight plug-in' in yüklenmiş olması yeterlidir. Gelişmiş grafikler oluşturabilir ve bunlara her türlü olay (event) atanabilir. Çok esnek bir animasyon hazırlama yapısına sahiptir. Desteklenen video ve sesleri projelere dahil etmek son derece kolaydır. İstemci (client) bazlı çalıştığı için sunucu tabanlı dillere göre hızlıdır. Silverlight

istemci tarafında çalışır ama sunucu tarafındaki gibi C# yada VB dillerinde üzerine kod yazılabilir. Silverlight, java applet ve javascript gibi istemci makinasında çalışır. Sunucudan veriye ulaşmak istendiğinde, direk olarak sunucuya erişilememektedir. Bunun için web servislerinden yararlanılmaktadır. Sunucuda hizmet veren web servis ile sunucuya istek (request) gönderilebilir. Bu tez çalışmasındaki İnternet Şube uygulamasında veritabanından bilgi almak ve işlem yapmak için web servisler yazılmıştır. Kullanılan servis modeli Silverlight Enabled WCF Service dir. Silverlight Enabled WCF Service özel bir binary binding kullanır. Böylece sunucu ile istemci arasındaki veri trafiği serileştirme işlemlerinde ara ürün olarak binary kullanıldığı için toplam giden gelen veri miktarı azalmakta ve harcanan işlemci yükü düşük olmaktadır [9].



Şekil. 2.1. WCF RIA Services [10]

Veritabanı olarak SQL Server 2008 kullanılmıştır. Veritabanında Table oluşturma, View oluşturma, Stored Procedure(SP) yazma gibi işlemler SQL Server 2008 Management Studio(SSMS) aracılığı ile yapılmıştır.

SQL Server 2008, Microsoft .NET ve Visual Studio kullanılarak geliştirilmiş özel uygulamalardan ve Microsoft BizTalk Server üzerinden hizmet odaklı mimariler (SOA) ve iş süreçlerinden gelen verilerin kullanılmasını sağlar. SQL Server 2008 verilerle sorgu, arama, senkronizasyon, raporlama ve analiz gibi daha fazla işlem gerçekleştirmeyi sağlayan zengin bir entegre hizmetler seti sunar [11]. Büyük veriler ile çalışılacağı öngörülüyor olsada güvenlik ve standardizasyon sebepleriyle lisans ile dağıtılan Microsoft SQL Server ve Oracle versiyonları tercih edilmektedir.

3. İNTERNET BANKACILIĞI VE SANAL İNTERNET ŞUBE UYGULAMASI

İnternet, kuruluşlara ve müşterilere getirdiği avantajlar nedeniyle bankacılık sektöründe giderek artan bir öneme sahiptir [12]. Bu çalışmada geliştirilen uygulama arayüzüyle bankaların İnternet Şube uygulamalarının eksik görünen ve eleştirilen tarafları göz önünde bulundurularak, bir banka İnternet Şube uygulamasının Silverlight teknolojisi ile nasıl daha iyi olabileceği gösterilmeye çalışılmıştır..

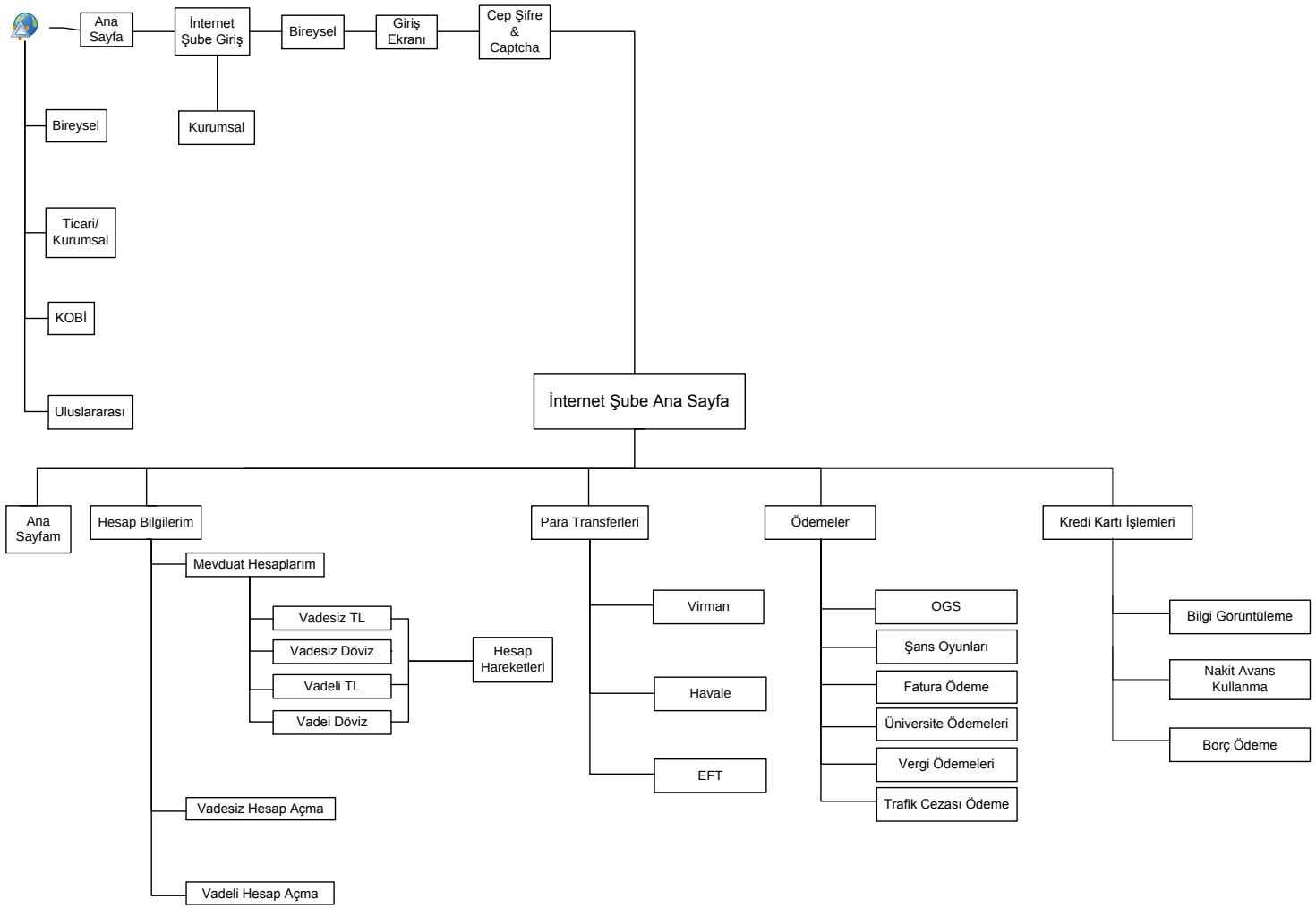
Uygulamada örnek olması amacıyla bazı bankacılık işlemleri yapılabilmektedir. Bunlar;

- Hesap Bilgileri Görüntüleme
- Hesp Hareketleri Görüntüleme
- Hesap Açma(Vadeli, Vadesiz)
- Para Transferi(Virman, Havale, EFT)
- Ödemeler(OGS, Şans Oyunları, Fatura Borç Ödeme, Üniversite Harç Ödeme, Vergi Ödeme, Trafik Cezası Ödeme)
- Kredi Kartı İşlemleri(Bilgi Görüntüleme, Nakit Avans Kullanma, Borç Ödeme)

İnternet Şube uygulamalarında önemi büyük olan Veritabanları, yaygın ve kolaylıkla bir çok güçlü veritabanı sistemine geçirilebilen Structured Query Language (SQL) dili kullanılarak tasarlanmıştır, Microsoft SQL Server gibi bilenen güçlü sunucuda çalışılmıştır.

Aşağıda bu tez çalışmasında geliştirilen banka İnternet Şube uygulamasının menü şemasını görebilirsiniz.

3.1. İnternet Şube Menü Şeması



Şekil. 3.1. İnternet Şube Menü Şeması

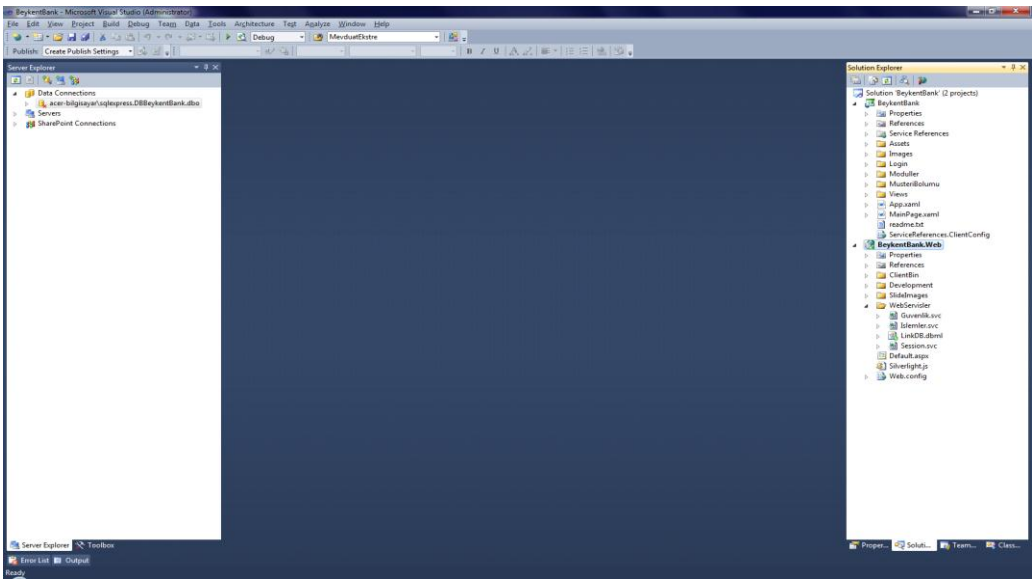
3.2. İnternet Şube Uygulamasının Yazılımsal Bileşenleri

Bu bileşenler geliştirme, sunucu ve istemci olarak bilinmektedir. Geliştirme biriminde, uygulamanın tasarlandığı program araçları, sunucu biriminde uygulamanın yayınlandığı sunucu sistemi, istemci biriminde ise uygulamayı kullanacak istemci bilgisayarların içerikleri hakkında detaylar verilmiştir.

3.2.1. Geliştirme Birimi

Geliştirme, Visual Studio 2010 (Integrated Development Environment – IDE) kullanılarak yazılmıştır. Uygulamanın dili C# dır. Program dili olarak C#'ın tercih edilmesinin nedeni hem makine diline hem de insan algısına eşit seviyede olmasıdır. C# uygulamaları hafıza ve işlemci gereksinimleri ile tutumlu olmak uzere tasarlanmıştır [13].

Visual Studio 2010, Microsoft'un IDE ürünüdür, oldukçagüçlüdür. Ayrıca uygulamaları entegre bir biçimde geliştirilebilmektedir. Bu uygulama'nın geliştirilmesinde olduğu gibi, Silverlight uygulaması, web servis'lerin yazımı, veritabanına bağlanılması ve tablo yapılarının alınması tek bir uygulama geliştirme ortamı kullanılarak yapılabilmiştir.



Şekil. 3.2. Visual Studio 2010 IDE

3.2.2. Sunucu Birimi

Web uygulamaları sunucu ve istemci bileşenlerinden oluşur. Uygulamanın çalışabilmesi için Silverlight uyumlu bir web sunucusuna ihtiyaç duyulur. Sunucuda .NET Framework 4 yüklü olmalıdır. Veritabanı için SQL Server 2008'in yüklü olduğu bir veritabanı sunucusu olmalıdır.

3.2.3. İstemci Birimi

Kullanıcının Browser'ında Silverligt Plug-in yüklü olmalıdır. http://www.microsoft.com/getsilverlight/ adresinden son versiyonu kurulabilir.

4. VERİTABANI

Tez için geliştirilen uygulamada veritabanı olarak SQL Server 2008 kullanılmıştır. Veritabanındaki tablolara(Table), View'lere, Stored Procedure'lere erişim için Microsoft'un ürünü olan SSMS - SQL Server Management Studio kullanılmıştır. Proje kapsamında geliştirilen Veritabanına "InternetSubev2" ismi verilmiş olup, Veritabanı 27 Table, 14 View, 18 Stored Procedure'den oluşmuştur.

Microsoft'un veritabanı teknolojisi üretmi, SQL Server'ın alt yapısının Sybase tarafından Microsoft'a satılmasıyla başlamıştır. SQL Server 1.0 için Microsoft, Sybase ve Ashton-Tate ile çalışmıştır. 1992'de Microsoft SQL Server 4.2'yi çıkarmıştır. Microsoft SQL Server 6.0 Windows NT için dizayn edilmiş ilk versiyodur. SQL Server 2008, 6 Ağustos 2008 tarihinde piyasaya sürülmüştür.

Microsoft, SQL Server'ı farklı versiyonlar ile sunmaktadır. Bu versiyonlar, farklı özellik setleri içerdiği gibi, farklı kullanıcıları da hedef alır [14].

i) Temel (Mainstream) Versiyonlar

Datacenter

SQL Server 2008 R2 Datacenter, yüksek seviye uygulama desteği ve ölçeklenebilirlik ihtiyaçlarını gideren, verimerkezleri (datacenters) için tasarlanmıştır. 256 mantıksal işlemciyi ve sınırsız hafıza desteklemektedir. StreamInsight Premium versiyonuyla birlikte gelmektedir.

Enterprise

SQL Server Enterprise versiyonu, SQL Server kümelerini yaratmak ve düzenlemek için gerekli araçları içeren versiyon olup, çekirdek veritabanı motoru ve add-on servislerini desteklemektedir. 524 petabayta kadar veritabanı yönetebilmekte, 2 terabayt hafıza içermekte ve 8 mantıksal işlemci desteklemektedir.

Standard

SQL Server Standart versiyonu, çekirdek veritabanı motorunu, bağımsız servislerle (stand-alone services) birlikte içermektedir. Daha az aktif instance (kümedeki ağ sayısı) desteklemesi, hot-add memory (sunucu çalışırken hafıza eklenebilmesi) gibi yüksek süreklilik fonksyonları ve paralel indeksleri içermemesi, SQL Server Enterprise versiyonundan farklı olduğu noktalar olmaktadır.

Web

SQL Server Web versiyonu, Web hosting için "low-TCO" (toplamda sahip olma maliyeti) bir seçenektir.

Workgroup

SQL Server Workgroup versiyonu, çekirdek veritabanı fonksyonlarını içermekte fakat ek servisleri içermemektedir.

Express

SQL Server Express versiyonu, çekirdek veritabanı motorunu içeren, ücretsiz bir versiyondur. Veritabanı ve kullanıcı sayısında bir sınırlama yoktur, ancak 1 işlemci, 1 GB hafıza ve 4 GB veritabanı dosya limiti bulunmaktadır.

ii) Özelleştirilmiş (Specialized) Versiyonlar

Azure

Microsoft SQL Azure Veritabanı, Microsoft SQL Server'ın bulut bazlı (cloud-based) versiyonudur. Azure Servisleri Platform'unda "software as a service" (servis olarak yazılım) olarak sunulmuştur.

Compact (SQL CE)

SQL Server Compact versiyonu, gömülü bir veritabanı motorudur. Diğer SQL Server versiyonlarının aksine, SQL CE motoru, SQL Mobile üzerine kuruludur ve aynı ikili değerleri paylaşmamaktadır. Boyutunun

küçüklüğüne (1 MB DLL footprint) ek olarak, özellik setleri belirgin olarak azaltılmıştır. Windows Service olarak çalıştırılamayacağı gibi, 4 GB veritabanı limiti bulunmaktadır.

Developer

SQL Server Developer versiyonu, SQL Server Datacenter versiyonuyla aynı özellikleri içermektedir, geliştirme (development) ve test sistemi olarak kullanılabilmektedir.

Embedded (SSEE)

SQL Server Embedded versiyonu, SQL Server Express veritabanı motorunun, sadece belli Windows Servisleri tarafından ulaşılabilen, özel olarak düzenlenmiş bir ismidir.

Evaluation

SQL Server Evaluation versiyonu, Deneme Sürümü (Trial Edition) olarakta bilinen, Enterprise versiyonunun tüm özelliklerini içeren, sadece 180 günle sınırlı olan versiyonudur.

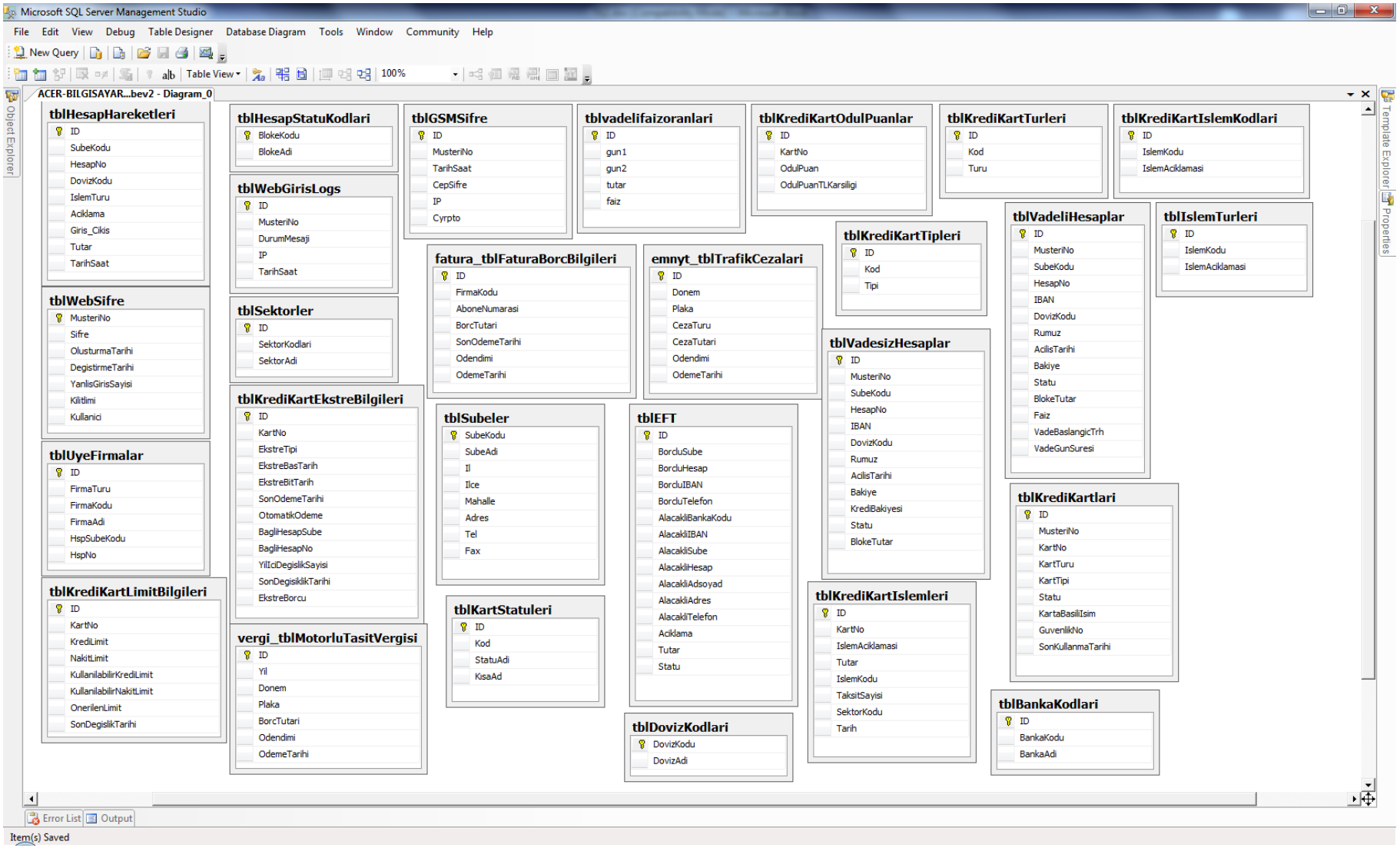
Fast Track

SQL Server Fast Track, özel olarak, ticari kurum bazında, veri depolama ve zeka işleme (intelligence processing) işlemlerinde kullanılan versiyondur.

Parallel Data Warehouse (PDW)

Parallel Data Warehouse versiyonu, yüzlerce terabaytlık veri depolama işlemleri için optimize edilmiş versiyondur.

Tez için kullanılan veritabanı, ücretsiz olduğu ve testler tek işlemci ve 4GB dan az veri kapasitesi ile yapıldığı için SQL Server 2008 Express versiyonunda oluşturulmuştur. İhtiyaca göre diğer versiyonlarda kullanılabilir. Veritabanı diagramı ve veritabanı nesneleri aşağıda listelenmiştir.



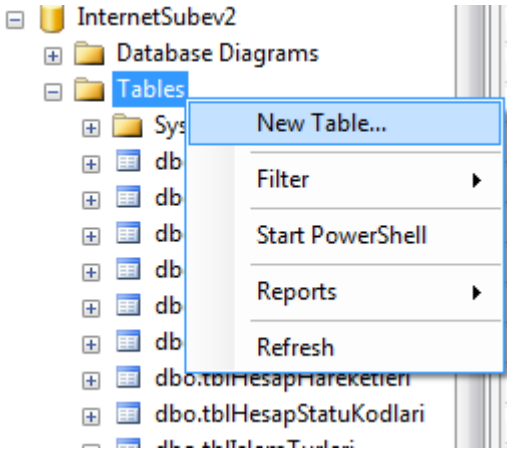
Şekil. 4.1. InternetSubev2 Veritabanı Diagram'ı

4.1. Tables

Tez kapsamında geliştirilen veritabanında 27 Table oluşturulmuştur.

Tablolar(Tables), satırlar(rows) ve sütunlar(colomns) dan oluşur. Tablo'daki alanlar(fields) sütunlardır. Satırlar veriler dir. Tablo'lar SSMS'de veritabanın'ın altında Table sekmesinden görüntülenebilmektedir. Aşağıda SSMS'da Table'ın nasıl oluşturulduğu Tez kasamında geliştirilen veritabanındaki "tblEFT" Table'ı oluşturularak gösterilmiştir.

1. Veritabanı içindeki Tables klasörüne sağ tuşla tıklayıp "New Table…" seçilir.



Şekil. 4.2. New Table

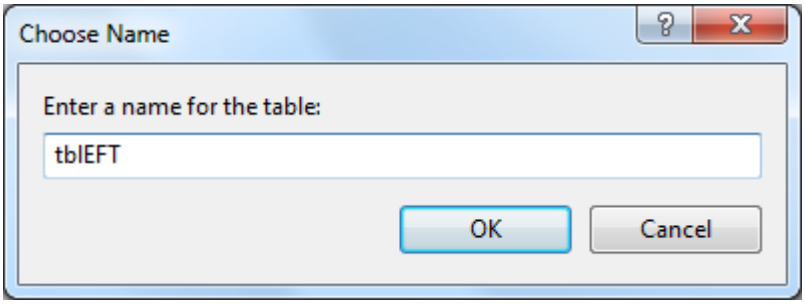
2. Table 15 field'den oluşmaktadır. Bunlar ID, BorcluSube, BorcluHesap, BorcluIBAN, BorcluTelefon, AlacakliBankaKodu, AlacakliIBAN, AlacakliSube, AlacakliHesap, AlacakliAdsoyad, AlacakliAdres, AlacakliTelefon, Aciklama, Tutar, Statu alanlarıdır. ID alanını birincil anahtar(Primary Key) ve otomatik artan(Identity Specification sekmesi altında bulunan Is Identity alanı Yes değerini almalı) olacaktır. "Column Name" bölümünde tabloda kullanılacak field'lar alt alta yazılır. Field'ın yan tarafında bulunan "Data Type" alanında veri tipi seçilir.(nvarchar, char, int vs…) Field'a

veri girilmesi zorunlu olacak ise yanındaki "Allow Nulls" kutucuğu temizlenir. Aşağıda tblEFT tablosunun SSMS'da Design modda görünümü gösterilmiştir.

Column Name	Data Type	Allow Nulls
ID	int	☐
BorcluSube	int	☑
BorcluHesap	int	☑
BorcluIBAN	nvarchar(50)	☑
BorcluTelefon	nvarchar(50)	☑
AlacakliBankaKodu	int	☑
AlacakliIBAN	nvarchar(50)	☑
AlacakliSube	nvarchar(50)	☑
AlacakliHesap	nvarchar(50)	☑
AlacakliAdsoyad	nvarchar(50)	☑
AlacakliAdres	nvarchar(50)	☑
AlacakliTelefon	nvarchar(50)	☑
Aciklama	nvarchar(MAX)	☑
Tutar	int	☑
Statu	nvarchar(50)	☑
		☐

Şekil. 4.3. tblEFT tablosu Design görünümü

3. Save butonuna tıklanarak "tblEFT" tablosu kaydedilir.



Şekil. 4.4. tblEFT Save

Aşağıda Tez kapsamında oluşturulan Table'lar listelenerek açıklanmıştır.

- **tblBankaKodlari**: Diğer bankaların kod bilgileri tutulmaktadır. Müşterilere EFT işleminde transfer işleminin yapılacağı banka seçeneklerini sunmak için veri depolanır.

- **tblDovizKodlari**: Döviz kodları tutulmaktadır. Mevduat hesapları listelendiğinde veya hesap açma işleminde döviz türünün seçilmesi için veri depolanır.

- **tblEFT**: Uygulamadan yapılan EFT işlemleri burada tutulmaktadır. Tablo'da yapılan EFT işlemi ile ilgili alacaklı ve borçlu bilgileri depolanmaktadır.

- **tblGSMSifre**: Müşterinin GSM telefonuna gönderilen şifre bilgilerinin depolandığı tablodur. Uygulama ile güvenli bilgi alış-verişi sağlamak için bu tabloda Crypto adı verilen alan tanımlanmıştır. Crypto, müşterinin IP'si ile Time bilgisini yanyana birleştirilerek oluşturuluyor ve uygulama bunu hem data tipi string olan crypto adında değişkene atıyor hemde veritabanına kaydediyor. GSM Sifre kontrolü yapılırken uygulama cyrpto'yu veritabanına iletiyor ve veritabanındaki Stored Procedure tablodaki deger ile uygulamadan gelen değeri karşılaştırıyor.

- **tblHesapHareketleri**: Mevduat hesabından yapılan işlem bilgileri, hesaba giriş-çıkış'lar bu tabloda tutulur.

- **tblHesapStatuKodlari:** Mevduat hesabının alabileceği statü bilgileri burada depolanmaktadır.(Açık, Kapalı, Donuk, Blokeli)

- **tblIslemTurleri:** İşlem kodları ve açıklamaları burada depolanmaktadır. (Havale, Virman, EFT, OGS Ödemesi, Kredi Kartı Nakit Avans, Kredi Kartı Borç Ödeme)

- **tblKartStatuleri:** Debit ve Kredi kartlarının alabileceği statü kodları ve açıklamaları burada depolanmaktadır.(Açık, Kayıp, Çalıntı, Müşteri İptal, Geçici İptal, Fraud)

- **tblKrediKartEkstreBilgileri:** Ekstre tipi, ekstre kesim tarihi, son ödeme tarihi, otomatik ödeme talimatı varmı yokmu, bağlı hesap bilgileri, ekstre borcu gibi müşteri kartlarının ekstre bilgileri burada depolanmatadır.

- **tblKrediKartIslemKodlari:** Kredi kartı ile yapılabilen işlemlerin kodları ve açıklamaları depolanmaktadır.(Peşin işlem, taksitli işlem)

- **tblKrediKartIslemleri:** Kredi kartından yapılan işlem bilgileri burada depolanmaktadır. İşlem kodu, tutar, taksit sayısı, sektör kodu, tarih bilgileri içermektedir.

- **tblKrediKartlari:** Müşterilerin kredi kart bilgileri depolanmaktadır. Kart türü, kart statüsü, kartın üstünde basılı olan ad soyad, kartın güvenlik numarası, son kullanma tarihi bilgilerini içermektedir.

- **tblKrediKartLimitBilgileri:** Müşteri kartlarının limit bilgilerinin depolandığı tablodur. Kartın toplam kredi limiti,toplam nakit avans limiti, kullanılabilir kredi limiti, kullanılabilir nakit avans limiti, banka tarafından önerilen limit, son limit değişiklik tarihi bilgilerini içermektedir.

- **tblKrediKartOdulPuanlar:** Banka tarafından kartını kullanan müşterilere belirlenen oranda işlem tutarına göre hediye puan verilmektdir. Müşteriler bu puanların TL karşılığı kadar bedava alış-veriş yapabilmektedirler.

- **tblKrediKartTipleri:** Banka tarafından belirlenen kart türlerinin kod ve açıklama bilgileri bu tabloda depolanmaktadır.

- **tblKrediKartTurleri:** Kartların alabileceği tür kodları burada depolanmaktadır.(Asıl kart, ekkart)

- **tblSektorler:** Sektör kodları ve açıklamaları bu tabloda depolanmaktadır. Kredi kartı ile yapılan harcama işlemlerinde hangi sektörden işlem yapıldıysa ilgili sektör kodu bu tablodan alınarak kredi kartı işlemleri tablosuna kaydedilir.(Tekstil, Elektronik, Kuyumcu, Sağlık, Gıda)

- **tblSubeler:** Bankanın şubelerinin bilgileri bu tabloda depolanır. Şube kodu, şube adı, bulunduğu il, ilçe, mahalle, adres, telefon, fax alanlarını içermektedir.

- **tblUyeFirmalar**: Bankanın üye işyerlerinin bilgileri bu tabloda depolanır. Ödeme işlemlerinde firma bilgileri tablodan alınır.(Fatura ödeme, şans oyunları ödemesi)

- **tblvadelifaizoranlari:** Bankanın vadeli hesaplar için uyguladığı güncel faiz oranları bu tabloda depolanmaktadır. Vade aralığına göre faiz uygulandığı için gun1 ve gun2 alanları tanıumlanmıştır. Bu alanlar vade aralığını ifade etmektedirler. Tutar, faiz alanlarını içermektedir.

- **tblVadeliHesaplar:** Müşteri vadeli mevduat hesap bilgileri bu tabloda depolanır. Hesap numarası, bulunduğu şube kodu, IBAN bilgisi, hesabın para cinsi, hesabın rumuzu, açılış tarihi, bakiyesi, hesap statüsü, blokede olan tutar, uygulanan faiz oranı, vade başlangıç tarihi alanlarını içermektedir.

- **tblVadesizHesaplar:** Müşteri vadesiz mevduat hesap bilgileri bu tabloda depolanır. Hesap numarası, bulunduğu şube kodu, IBAN bilgisi, hesabın para cinsi, hesabın rumuzu, açılış tarihi, bakiyesi, ek hesap bakiyesi, hesap statüsü, blokede olan tutar alanlarını içermektedir.

- **tblWebGirisLogs:** Burada müşterinin uygulamaya giriş bilgileri tutulur. Müşteri numarası, Durum mesajı, IP, Tarih ve saat alanlarını içermektedir.

- **tblWebSifre:** Müşterinin uygulamaya girerken kullandığı şifre bilgisinin depolandığı tablodur. Şifre SHA1 hash algoritmasıyla şifrelenmiş olarak depolanıyor.

- **emnyt_tblTrafikCezalari:** Uygulamada geliştirilen Ödemeler sekmesi altındaki Tarfik Cezası bölümünde kullanılmak için tasarlanmıştır. Gerçekte olması gereken bu bilgilerin ilgili kurum ile yapılan bağlantıdan alınması dır.

- **fatura_tblFaturaBorcBilgileri:** Uygulamada geliştirilen Ödemeler sekmesi altındaki Online Fatura Ödeme bölümünde kullanılmak için tasarlanmıştır. Gerçekte olması gereken bu bilgilerin ilgili kurum ile yapılan bağlantıdan alınması dır.

- **vergi_tblMotorluTasitVergisi:** Uygulamada geliştirilen Ödemeler sekmesi altındaki Motorlu Taşıtlar Vergisi bölümünde kullanılmak için tasarlanmıştır. Gerçekte olması gereken bu bilgilerin ilgili kurum ile yapılan bağlantıdan alınması dır.

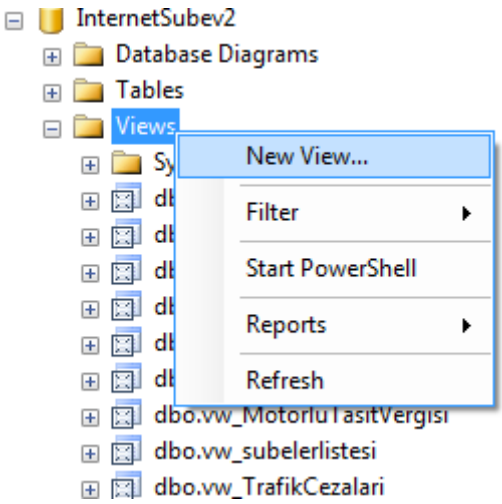
4.2. Views

View'ler SSMS'de veritabanın'ın altında View sekmesinden görüntülenebilmektedir. Tablolardan istenilen verileri bir arada getirip tek tablo olarak gösteren sorgulardır. Veri saklamazlar, verileri tablolardan anlık alırlar. Tablolarda değişiklik olsada bu değişiklik eş zamanlı olara View'lerede yansımaktadır. Birden fazla tablodan bilgi alınmak istendiğinde bu tablolardan View oluşturarak gerekli veri hem hızlı hemde düzgün olarak alınabilir. Sql Server'da View oluşturabilmek için gerekli rol'e sahip olunmalı veya Create View yetkisi kullanıcıya verilmiş olmalıdır.

Sql Server'da View oluşturabilmek için gerekli roller; **db_owner, db_ddladmin, sysadmin** dir.

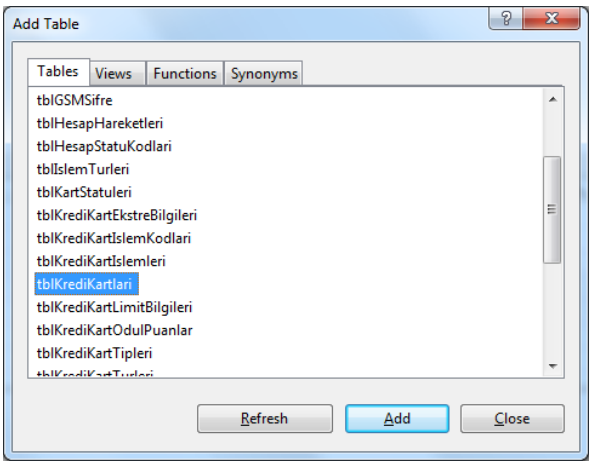
Aşağıda SSMS'de View'in nasıl oluşturulduğu Tez kasamında geliştirilen veritabanındaki "vw_KrediKartlari" View'i oluşturularak gösterilmiştir.

1. Veritabanı içindeki Views klasörüne sağ tuşla tıklayıp "New View…" seçilir.



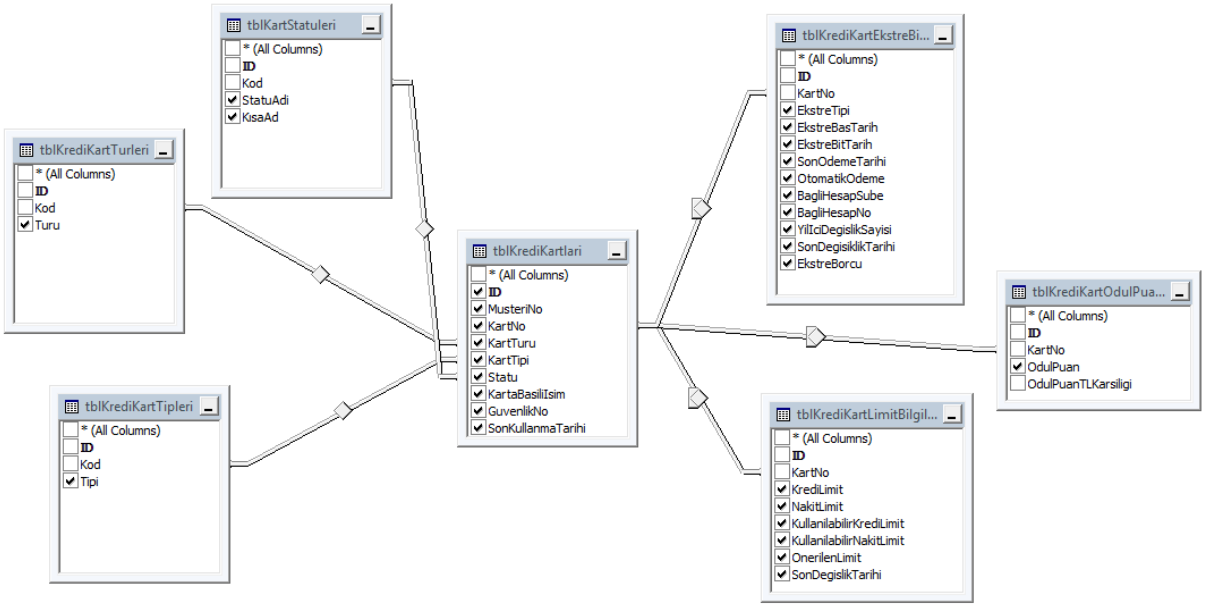
Şekil. 4.5. New View

2. Gelen ekrandan tblKrediKartlari, tblKrediKartTurleri, tblKrediKartTipleri, tblKartStatuleri, tblKrediKartLimitBilgileri, tblKrediKartEkstreBilgileri, tbl-KrediKartOdulPuanlar tabloları eklenir.



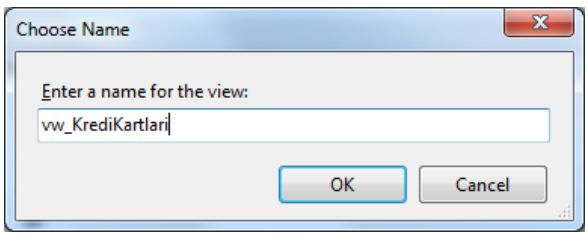
Şekil. 4.6. Add Table

3. Tablolardan hangi alanlar View'de gösterilmek isteniyorsa alan adının yanındaki kutucuk(Checkbox) işaretlenir.



Şekil. 4.7. View

4. Oluşturulan View "vw_KrediKartlari" adıyla kaydilmiştir.



Şekil. 4.8. vw_KrediKartlari Save

Aşağıda Tez kapsamında oluşturulan View'ler listelenerek açıklanmıştır.

- **vw_bankakodlari**: tblBankaKodlari tablosundaki alanların hepsi seçilerek oluşturulmuştur.

- **vw_dovizkodlari** : tblDovizKodlari tablosundaki alanların hepsi seçilerek oluşturulmuştur.

- **vw_FaturaBorcBilgileri** : fatura_tblFaturaBorcBilgileri tablosundaki alanların hepsi seçilerek oluşturulmuştur.
- **vw_hesaphareketleri** : tblHesapHareketleri ile tblSubeler tabloları SubeKodu alanı ile join yapılmıştır. tblHesapHareketleri'ndeki IslemTuru alanı ile tblIslemTurleri tablosundaki IslemKodu alanı ile iki tablo join yapılmıştır. tblHesapHareketleri ile tblDovizKodlari tabloları DovizKodu alanı ile join yapılmıştır. tblHesapHareketleri tablosundaki bütün alanlar, tblSubeler tablosundaki SubeAdi alanı, tblIslemTurleri tablosundaki IslemAciklamasi alanı, tblDovizKodlari tablosundaki DovizAdi alanı seçilmiştir.

- **vw_KrediKartIslemleri**: tblKrediKartIslemleri ile tblKrediKartIslemKodlari tabloları IslemKodu alanı ile join yapılmıştır. tblKrediKartIslemleri tablosundaki SektorKodu alanı ile tblSektorler tablosundaki SektorKodlari alanı ile iki tablo join yapılmıştır. tblKrediKartIslemleri tablosundaki bütün alanlar, tblKrediKartIslemKodlari tablosundaki IslemAciklamasi, tblSektorler tablosundaki SektorAdi alanları seçilmiştir.

- **vw_KrediKartlari**: tblKrediKartlari tablosu ile tblKrediKartTurleri, tblKrediKartTipleri, tblKartStatuleri, tblKrediKartLimitBilgileri, tblKrediKartEkstreBilgileri, tblKrediKartOdulPuanlar tabloları KartNo alanı ile joinlenmiştir. tblKrediKartlari tablosundaki bütün alanlar, tblKrediKartEkstreBilgileri tablosundaki KartNo hariç bütün alanlar, tblKrediKartOdulPuanlar tablosundaki OdulPuan alanı, tblKrediKartLimitBilgileri tablosundaki KartNo hariç bütün alanlar, tblKartStatuleri tablosundaki StatuAdi ve KısaAd alanları, tblKrediKartTurleri tablosundaki Turu, tblKrediKartTipleri tablosundaki Tipi alanı seçilmiştir.

- **vw_MotorluTasitVergisi**: vergi_tblMotorluTasitVergisi tablosundaki alanların hepsi seçilerek oluşturulmuştur.

- **vw_subelerlistesi**: tblSubeler tablosundaki tablosundaki alanların hepsi seçilerek oluşturulmuştur.

- **vw_TrafikCezalari:** emnyt_tblTrafikCezalari tablosundaki alanların hepsi seçilerek oluşturulmuştur.

- **vw_UyeFirmalar**: tblUyeFirmalar tablosundaki alanların hepsi seçilerek oluşturulmuştur.

- **vw_vadelifaizoranlari**: tblvadelifaizoranlari tablosundaki alanların hepsi seçilerek oluşturulmuştur.

- **vw_vadelihesaplar**: tblVadeliHesaplar ile tblDovizKodlari tabloları DovizKodu alanı ile join yapılmıştır. tblVadeliHesaplar ile tblSubeler tablosu SubeKodu alanı ile join yapılmıştır. tblVadeliHesaplar tablosundaki Statu alanı tblHesapStatuKodlari tablosundaki BlokeKodu alanı ile iki tablo join yapılmıştır. tblVadeliHesaplar tablosundaki bütün alanlar, tblSubeler tablosundan SubeAdi alanı, tblHesapStatuKodlari tablosundan BlokeAdi, tblDovizKodlari tablosundan DovizAdi alanı seçilmiştir.

- **vw_vadesizhesaplar**: tblVadesizHesaplar ile tblDovizKodlari tabloları DovizKodu alanı ile join yapılmıştır. tblVadesizHesaplarile tblSubeler tablosu SubeKodu alanı ile join yapılmıştır. tblVadesizHesaplartablosundaki Statu alanı tblHesapStatuKodlari tablosundaki BlokeKodu alanı ile iki tablo join yapılmıştır. tblVadesizHesaplar tablosundaki bütün alanlar, tblSubeler tablosundan SubeAdi alanı, tblHesapStatuKodlari tablosundan BlokeAdi, tblDovizKodlari tablosundan DovizAdi alanı seçilmiştir.

- **vw_webkullanici**: tblWebSifre tablosundaki alanların hepsi seçilerek oluşturulmuştur.

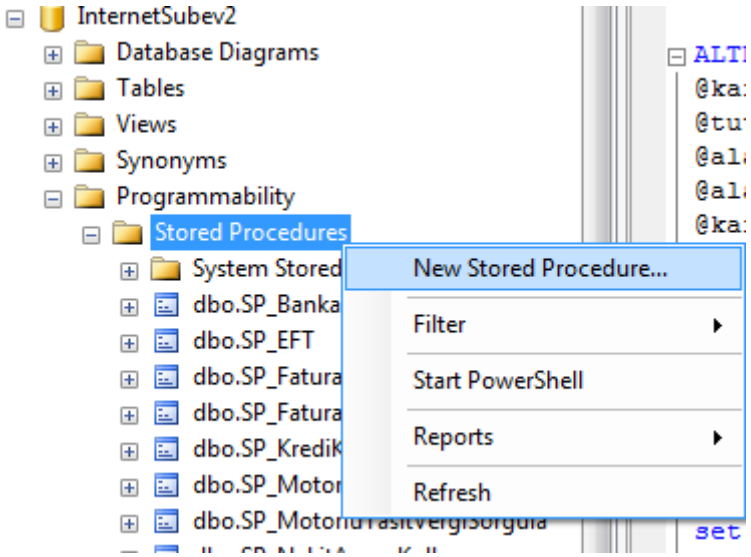
4.3. Stored Procedures

Stored Procedure(SP)'ler SSMS'de Database'in altında Programmability sekmesinin altında Stored Procedures sekmesinden görüntülenebilmektedir. Gerçekte Stored Procedure'ler veritabanında saklanan sql ifadeleridir. Dışarıdan parametre alabilirler ve sorgu içinde değişken tanımlaması izin verebilirler.

Stored Procedure'ler veritabanında saklandığından dolayı daha hızlı çalışırlar. Bir Stored Procedure ilk çalıştırıldığı zaman derlenir. Sonraki çalıştırmalarda derlenmelerine gerek yoktur. Bir uygulama içinden sql ifadeleri ile SQL Server'a bağlanıldığında her bağlantıda SQL derlenmesi zaman kaybına neden olmaktadır. Stored Procedure'un bir özelliğide programlama deyimleri içermesidir. If, next, set gibi programlama dillerindeki yapılara benzer özellikler sunar. Hatta bazı progamcılar programlarının hiç bir yerinde SQL ifadesi kullanmazlar. Her zaman stored procedure'ler ile çalışırlar [15].

Aşağıda SSMS'de Stored Procedure(SP)'in nasıl oluşturulduğu Tez kapsamında geliştirilen "SP_NakitAvansKullan" Stored Procedure'i üzerinden gösterilmiştir.

1. Veritabanı içindeki Stored Procedures klasörüne sağ tuşla tıklayıp "New Stored Procedure…" seçilir.



Şekil. 4.9. New Stored Procedure

2. SSMS Create Procedure yapısını otomatik olarak aşağıdaki ekranda göründüğü gibi oluşturmaktadır.

```
-- ================================================
-- Template generated from Template Explorer using:
-- Create Procedure (New Menu).SQL
--
-- Use the Specify Values for Template Parameters
-- command (Ctrl-Shift-M) to fill in the parameter
-- values below.
--
-- This block of comments will not be included in
-- the definition of the procedure.
-- ================================================
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:      <Author,,Name>
-- Create date: <Create Date,,>
-- Description: <Description,,>
-- =============================================
CREATE PROCEDURE <Procedure_Name, sysname, ProcedureName>
    -- Add the parameters for the stored procedure here
    <@Param1, sysname, @p1> <Datatype_For_Param1, , int> = <Default_Value_For_Param1, , 0>,
    <@Param2, sysname, @p2> <Datatype_For_Param2, , int> = <Default_Value_For_Param2, , 0>
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for procedure here
    SELECT <@Param1, sysname, @p1>, <@Param2, sysname, @p2>
END
GO
```

Şekil. 4.10. Create Procedure

3. Aşağıda SP'de yazılan kodların satır satır açıklamaları yanlarına yazılarak anlatılmıştır.

```
CREATE PROCEDURE SP_NakitAvansKullan
@kartno nvarchar(20),
@tutar int,
@alacakli_sube int,
@alacakli_hesap int,
@kartguvenlikno int,
@sonkullanmatarihi date
AS
BEGIN
Declare @mevcutlimit int;
Declare @mevcutguvenlino int;
Declare @mevcutsonkullanmatrh date;
set @mevcutlimit = (select KullanilabilirNakitLimit from tblKrediKartLimitBilgileri where KartNo = @kartno)
set @mevcutguvenlino = (select GuvenlikNo from tblKrediKartlari where KartNo=@kartno)
set @mevcutsonkullanmatrh = (select SonKullanmaTarihi from tblKrediKartlari where KartNo=@kartno)

if(@mevcutsonkullanmatrh = @sonkullanmatarihi)
begin
if(@mevcutguvenlino = @kartguvenlikno)
begin
if(@tutar<=@mevcutlimit)
Begin
update tblKrediKartLimitBilgileri set KullanilabilirKrediLimit = KullanilabilirKrediLimit - @tutar,
KullanilabilirNakitLimit = KullanilabilirNakitLimit - @tutar where KartNo = @kartno
INSERT INTO tblKrediKartIslemleri ([KartNo],[IslemAciklamasi],[Tutar],[IslemKodu],Tarih) VALUES (@kartno,'Nakit Avans Kullanımı',
@tutar,22,GETDATE())
UPDATE tblVadesizHesaplar SET Bakiye = Bakiye + @tutar WHERE SubeKodu = @alacakli_sube and HesapNo = @alacakli_hesap
INSERT INTO tblHesapHareketleri ([SubeKodu],[HesapNo],[DovizKodu],[IslemTuru],[Aciklama],[Giris_Cikis],[Tutar],[TarihSaat])
    VALUES (@alacakli_sube,@alacakli_hesap,100,200,@kartno+' nolu Karttan Nakit Avans Çekimi','+',@tutar,GETDATE())
End
Else
Begin
return 99 /*Limit yetersiz*/
End
end
Else
begin
return 75 /*Kartın Güvenlik Numarası Yanlış*/
end
end
else
begin
return 23 /*Kartın Son Kullanma Tarihi Hatalı*/
end

END
```

Şekil. 4.11. SP_NakitAvansKullan Kodları

- "CREATE PROCEDURE SP_NakitAvansKullan": SSMS'e oluşturulacak SP'nin adının "SP_ NakitAvansKullan" olacağı iletiliyor.

- "@kartno nvarchar(20),": @ işareti parametre tanımlarken parametre önüne eklenir. Parametre'nin adı "kartno" dur. Dışarıdan veri alacak parametre nvarchar(20) data tipinde tanımlanıyor. Nvarchar veri tipi sözel ifadeler

tutmaktadır. Parantez içindeki 20 karakter uzunluğunu maximum değeridir. 20 karakterden fazla veri almaz. Bir sınır belirtmek istenmiyorsa 20 yerine MAX yazılır.

- "@tutar int,": veri tipi sayısal olarak tanımlanıyor. Int'in açılımı integer' dır.
- "@alacakli_sube int,"
- "@alacakli_hesap int,"
- "@kartguvenlikno int,"
- "@sonkullanmatarihi date": veri tipi tarih olarak tanımlanıyor. Datetime veri tipi ile karıştırılmamalıdır. Datetime veri tipi veriyi tarih saat bitişik olarak tutmaktadır. Örnek olarak "01.07.2011 14:33:52". Date veri tipi sadece tarihi tutmaktadır. Örnek olarak "01.07.2011".
- AS
- "BEGIN": SP'nin kod bloğunun başladığı iletiliyor.
- "Declare @mevcutlimit int;": SP'nin içinde kullanılacak değişken tanımlaması yapılmaktadır. Değişkenin adı "mevcutlimit" tir. Değişken adının önüne @ işareti eklenir. Declare SP içinde kullanılacak değişken tanımlanacağını belirtiyor. Tanımlamanın bittiği ";" işareti ile belirtilmektedir.
- "Declare @mevcutguvenlino int;"
- "Declare @mevcutsonkullanmatrh date;"

- “set @mevcutlimit = (select KullanilabilirNakitLimit from tblKrediKartLimitBilgileri where KartNo = @kartno)”: Mevcutlimit değişkenine değer ataması yapılır. Select cümlesiyle tblKrediKart-LimitBilgileri tablosundaki KullanilabilirNakitLimit alanındaki değer dön-dürülür. Kriter olarak tablodaki KartNo alanın’ın kartno değişkeniyle aynı olması koşulu istenir.

- “set @mevcutguvenlino = (select GuvenlikNo from tblKrediKartlari where KartNo=@kartno)”

- “set @mevcutsonkullanmatrh = (select SonKullanmaTarihi from tblKrediKartlari where KartNo=@kartno)”

- “if(@mevcutsonkullanmatrh = @sonkullanmatarihi)”: Eğer mevcutsonkullanmatrh değişkeni ile sonkullanmatarihi değişkeni aynı ise aşağıdaki işlemlerin yapılması istenir.

- “Begin”: yapılması istenen işlemler bloğunun başladığını belirtir.

- “if(@mevcutguvenlino = @kartguvenlikno)”: Eğer mevcutguvenlino değişkeni ile kartguvenlikno değişkeni aynı ise aşağıdaki işlemlerin yapılması istenir.

- “Begin”: yapılması istenen işlemler bloğunun başladığını belirtir.

- “if(@tutar<=@mevcutlimit)”: Eğer tutar değişkeni mevcutlimit değişkeninden küçük veya eşit ise aşağıdaki işlemlerin yapılması istenir.

- “Begin” yapılması istenen işlemler bloğunun başladığını belirtir.

- “update tblKrediKartLimitBilgileri set KullanilabilirKrediLimit = Kullanil-abilirKrediLimit - @tutar, KullanilabilirNakitLimit = Kullanil-abilirNakitLimit

- @tutar where KartNo = @kartno": tblKrediKartLimitBilgileri tablosundaki KullanilabilirKrediLimit ve KullanilabilirNakitLimit alanları güncellenmektedir. Mevcut değerden tutar değişkeni çıkartılarak yeni değer atanmaktadır. Kriter olarak tablodaki KartNo alanın'ın kartno değişkeniyle aynı olması koşulu istenir.

- "INSERT INTO tblKrediKartIslemleri ([KartNo], [IslemAciklamasi], [Tutar], [IslemKodu], Tarih) VALUES (@kartno,'Nakit Avans Kullanımı', @tutar, 22, GETDATE())": tblKrediKartIslemleri tablosuna yeni veri eklenmektedir. KartNo alanına kartno değişkeni atanmaktadır. IslemAciklamasi alanına sabit veri olarak 'Nakit Avans Kullanımıatanıyor. Tutar alanına tutar değişkeni atanmaktadır. IslemKodu alanına sabit veri olarak 22(Peşin İşlem) atanmaktadır. Tarih alanına SSMS'in GETDATE() fonksiyonu çağrılarak Server'daki tarih bilgisi atanır.

- "UPDATE tblVadesizHesaplar SET Bakiye = Bakiye + @tutar WHERE SubeKodu = @alacakli_sube and HesapNo = @alacakli_hesap": tblVadesizHesaplar tablosu güncellenir. Mevcut Bakiye alanına tutar değişkeni eklenerek yeni değer atanır. Kriter olarak SubeKodu alanın'ın alacakli_sube değişkeni ile aynı olması ve HesapNo alanın'ın alacakli_hesap değişkeni ile aynı olması koşulu istenir.

- "INSERT INTO tblHesapHareketleri ([SubeKodu], [HesapNo], [DovizKodu], [IslemTuru], [Aciklama], [Giris_Cikis], [Tutar], [TarihSaat]) VALUES (@alacakli_sube, @alacakli_hesap, 100, 200, @kartno+' nolu Karttan Nakit Avans Çekimi', '+', @tutar, GETDATE())": tblHesapHareketleri tablosuna yeni veri eklenmektedir. SubeKodu alanına alacakli_sube değişkeni atanmaktadır. HesapNo alanına alacakli_hesap değişkeni atanır. DovizKodu alanına sabit veri olarak 100(TL) atanmaktadır. IslemKodu alanına sabit veri olarak 200(Kredi Kartı Nakit Avans) atanmaktadır. Aciklama alanına kartno+' nolu Karttan Nakit Avans Çekimi' atanmaktadır. Giris_Cikis alanına hesaba para girişini

ifade eden “+” işareti atanmaktadır. Tutar alanına tutar değişkeni atanmaktadır. Tarih alanına SSMS’in GETDATE() fonksiyonu çağrılarak Server’daki tarih bilgisi atanır.

- “End”: yapılması istenen işlemler bloğunun bittiğini belirtir.
- “Else”: Eğer tutar değişkeni mevcutlimit değişkeninden büyük ise aşağıdaki işlemlerin yapılması istenir.
- “Begin”: yapılması istenen işlemler bloğunun başladığını belirtir.
- “return 99 /*Limit yetersiz*/”: SP 99 değerini döner.
- “End”: yapılması istenen işlemler bloğunun bittiğini belirtir.
- “End”: yapılması istenen işlemler bloğunun bittiğini belirtir.
- “Else”: Eğer mevcutguvenlino değişkeni ile kartguvenlikno değişkeni farklı ise aşağıdaki işlemlerin yapılması istenir.
- “Begin”: yapılması istenen işlemler bloğunun başladığını belirtir.
- “return 75 /*Kartın Güvenlik Numarası Yanlış*/”: SP 75 değerini döner.
- “End”: yapılması istenen işlemler bloğunun bittiğini belirtir.
- “end”: yapılması istenen işlemler bloğunun bittiğini belirtir.
- “Else”: Eğer mevcutsonkullanmatrh değişkeni ile sonkullanmatarihi değişkeni farklı ise aşağıdaki işlemlerin yapılması istenir.

- “Begin”: yapılması istenen işlemler bloğunun başladığını belirtir.

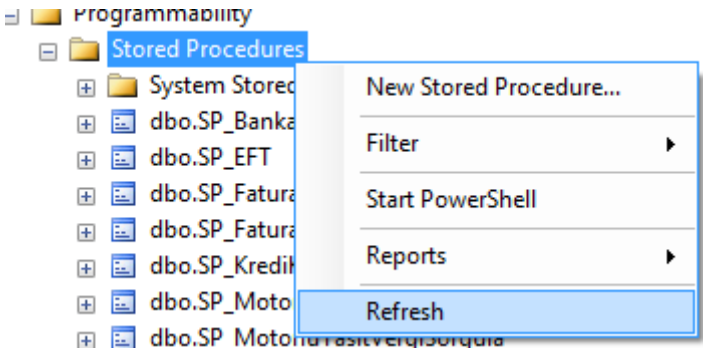
- “return 23 /*Kartın Son Kullanma Tarihi Hatalı*/”: SP 23 değerini döner. SP’de “/*” işareti ile başlayan ve “*/” işareti ile biten bloğun derlemeye dahil edilmeyeceği anlamına gelir.

- “End”: yapılması istenen işlemler bloğunun bittiğini belirtir.

- “END”: SP’nin bittiğini iletir.

4. SP yazımı tamamlandıktan sonra Execute butonu tıklanarak kod çalıştırılır ve SP_NakitAvansKullan Stored Procedure(SP)’ü oluşturulmuş olur. Bu SP’yi görebilmek için yan taraftaki Stored Procedure sekmesi sağ tıklanır ve Refresh yapılır.

Şekil. 4.12. Execute butonu



Şekil. 4.13. SP’nin yenilenmesi

Aşağıda Tez kapsamında oluşturulan Stored Procedure(SP)'ler listelenerek açıklanmıştır.

- **SP_BankaiciParaTransferi:** Aynı bankanın şubeleri arasındaki hesaplara para transferi için çağrılır. Dışarıdan 10 tane parametre almaktadır. Bunlar @borclu_musterino, @borclu_sube, @borclu_hesap, @borclu_dovizkodu, @alacakli_sube, @alacakli_hesap, @alacakli_dovizkodu, @tutar, @aciklama, @islemturu parametreleri dir. İlk olarak borçlu hesabın bakiyesi'nin aktarılacak tutar'a eşit veya fazla olması kontrol edilir. Eğer bakiye @tutar'a eşit veya daha fazla ise tblVadesizHesaplar tablosunda @borclu_hesap parametresi ile eşleşen kayıtın Bakiye alanındaki değerden @tutar parametresinin değeri çıkartılır. tblVadesizHesaplar tablosundaki @alacakli_hesap ile eşleşen kayıtın alanındaki değerden @tutar parametresinin değeri eklenir. Hesaplarda oluşan hareketler, tblHesapHareketleri tablosuna eklenir.(Borçlu ve alacaklı esap bilgileri, tutar, açıklama, işlem türü)

- **SP_EFT**: EFT işleminde çağrılır. Dışarıdan 13 parametre alır. Bunlar @borclusube, @borcluhesap, @borcluIBAN, @borclutelefon, @alacaklibankakod, @alacakliIBAN, @alacaklisube, @alacaklihesap, @alacakliadsoyad, @alacakliadres, @alacaklitelefon, @aciklama, @tutar parametreleri dir. İlk olarak borçlu hesabın bakiyesi'nin aktarılacak tutar'a eşit veya fazla olması kontrol edilir. Eğer bakiye @tutar'a eşit veya daha fazla ise tblVadesizHesaplar tablosunda @borclu_hesap parametresi ile eşleşen kayıtın Bakiye alanındaki değerden @tutar parametresinin değeri çıkartılır. tblEFT tablosuna borçlu ve alacaklı hesap bilgileri, telefon bilgileri, adres, ad soyad, açıklama, tutar bilgileri eklenir. Hesapta oluşan para çıkış hareketi, tblHesapHareketleri tablosuna eklenir.(Borçlu ve alacaklı hesap bilgileri, tutar, açıklama, işlem türü)

- **SP_FaturaBorcOdeme**: Fatura borcu ödeme işlemi için çağrılır. Dışarıdan 2 parametre alır. Bunlar @ID, @Tutar parametreleri dir. @ID parametresi

faturanın uniq ID numarasını ifade etmektedir. @Tutar fatura borcu değerini alır. SP içinde @borc, @sonuc adında 2 tane değişken tanımlanmıştır. İlk olarak fatura_tblFaturaBorcBilgileri tablosundan @ID ile eşleşen kayıtın BorcTutari bilgisi alınarak @borc değişkenine atanır. @Tutar parametresinin değeri ile @borc değişkenindeki değer karşılaştırılır. Eğer eşit ise fatura_tblFaturaBorcBilgileri tablosunda güncellemeler yapılır ve @sonuc değişkenine "0" değeri atanır.(İşlem başarılı) Eğer @Tutar ile @borc eşit değilse @sonuc değişkenine "1" değeri atanır.(Ödenecek tutar borcu dan farklı) SP çağrıldığında @sonuc değişkenini döner.

- **SP_FaturaBorcSorgulama**: Fatura borcu sorgulama işlemi için çağrılır. Dışarıdan @FirmaKodu, @AboneNumarasi adında 2 parametre alır. vw_FaturaBorcBilgileri View'inden @FirmaKodu ve @AboneNumarasi ile eşleşen kayıtın bilgilerini döner.

- **SP_KrediKartBorcOdeme:** Kredi kart borcu ödeme işlemi için çağrılır. Dışarıdan 4 parametre alır. Bunlar @kartno, @borclu_sube, @borclu_hesap, @tutar parametreleri dir. SP içinde @mevcutbakiye adında değişken tanımlanmıştır. Bu değişkene borçlu hesabın bakiye bilgisi atanır. Sonraki adımda @mevcutbakiye değişkeninin değeri ile @tutar parametresinin değeri karşılaştırılır. Eğer @mevcutbakiye, @tutar' a eşit veya fazla ise tblKrediKartEkstreBilgileri, tblVadesizHesaplar, tblKrediKartLimitBilgileri tabloları güncellenir. tblKrediKartIslemleri, tblHesapHareketleri tablolarına veri eklemesi yapılır.

- **SP_MotorluTasitBorcOdeme:** Motorlu Taşıt Vergi Borcu ödemesinde çağrılır. Dışarıdan @ID, @Tutar adında iki parametre alır. SP içinde @borc, @sonuc adında 2 değişken tanımlanır. vergi_tblMotorluTasitVergisi tablosundan @ID ile eşleşen kayıttan BorcTutari @borc değişkenine atanır. @borc değişkeni ile @Tutar parametresi karşılaştırılır. Eğer eşit ise vergi_tblMotorluTasitVergisi tablosu güncellenir ve @sonuc değişkenine "0"

değeri atanır. Eğer farklı ise @sonuc değişkenine “1” değeri atanır. SP çağrıldığında @sonuc değerini döner.

- **SP_MotorluTasitVergiSorgula:** Motorlu Taşıt Vergi Borcu sorgulamak için çağrılır. Dışarıdan @yil, @plaka adında iki parametre alır. vw_MotorluTasitVergisi View’inden bu parametreler ile eşleşen kayıtları döner.

- **SP_NakitAvansKullan:** Kredi kartından nakit avans çekme işleminde çağrılır. Dışarıdan 6 parametre alır. Bunlar @kartno, @tutar, @alacakli_sube, @alacakli_hesap, @kartguvenlikno, @sonkullanmatarihi parametreleri dir. SP içinde @mevcutlimit, @mevcutguvenlino, @mevcutsonkullanmatrh adında 3 değişken tanımlanmıştır. @mevcutlimit değişkenine tblKrediKartLimitBilgileri tablosundan @kartno ile eşleşen kayıttan KullanilabilirNakitLimit alanındaki değer atanmatadır. @mevcutguvenlino değişkenine tblKrediKartlari tablosundan @kartno ile eşleşen kayıttan GuvenlikNo alanındaki değer atanmatadır. @mevcutsonkullanmatrh değişkenine tblKrediKartlari tablosundan @kartno ile eşleşen kayıttan SonKullanmaTarihi alanındaki değer atanmatadır. @mevcutsonkullanmatrh ile @sonkullanmatarihi, @mevcutguvenlino ile @kartguvenlikno değişkenleri eşit, @mevcutlimit @tutar değişkeninden büyük veya eşit ise tblKrediKartLimitBilgileri ve tblVadesizHesaplar tabloları güncellenir. tblKrediKartIslemleri ve tblHesapHareketleri tablolarına veri eklenir.

- **SP_TrafikCezaOdeme:** Tarfik cezası ödeme işleminde çağrılır. Dışarıdan @ID ve @Tutar adında 2 parametre alır. SP içinde @borc ve @sonuc adında 2 değişken tanımlanmıştır. @borc değişkenine emnyt_tblTrafikCezalari tablosundan @ID ile eşleşen kayıtın CezaTutari değeri atanmaktadır. @borc ile @Tutar değerleri aynı ise emnyt_tblTrafikCezalari tablosu güncellenir ve @sonuc değişkenine “0” değeri atanır. Eğer farklı ise @sonuc değişkenine “1” değeri atanır. SP çağrıldığında @sonuc değerini döner.

- **SP_TrafikCezaSorgula :** Trafik Cezası sorgulamak için çağrılır. Dışarıdan @donem, @plaka adında iki parametre alır. vw_ vw_TrafikCezalari View'inden bu parametreler ile eşleşen kayıtları döner.

- **SP_VadeliHesapAc:** Vadeli hesap açma işlemi için çağrılır. Dışarıdan 6 parametre alır. Bunlar @MusteriNo, @SubeKodu, @DovizKodu, @Rumuz, @VadeBaslangicTrh, @VadeGunSuresi parametreleri dir. SP içinde 4 değişken tanımlanmıştır. Bunlar @Rastgelesayi, @Maximum, @Minumum, @faiz dir. İlk olarak @faiz değişkenine Tblvadelifaizoranlari tablosundan @VadeGunSuresi'nin aralığında olduğu kayıtın faiz alanının değeri atanır. @Minumum değişkenine banka tarafından belirlenen minumum üretilebilecek hesap numarası atanır. @Maximum değişkenine banka tarafından belirlenen maximum üretilebilecek hesap numarası atanır. @Rastgelesayi değişkenine @Minumum ve @Maximum değerleri arasında rastgele sayı üretilerek atanır. Diğer adımda while döngüsü ile tblVadeliHesaplar tablosundaki HesapNo alanındaki değer ile @Rastgelesayi değişkenindeki değer farklı olana kadar rastgele sayı üretilir. Böylelikle üretilen hesap numarasının tabloda varlığı kontrol edilir. Sonuç farklı ise tblVadeliHesaplar tablosuna kayıt eklenir. SP @Rastgelesayi değerini döner.

- **SP_VadesizHesapAc:** Vadesiz hesap açma işlemi için çağrılır. Dışarıdan 4 parametre alır. Bunlar @MusteriNo, @SubeKodu, @DovizKodu, @Rumuz, parametreleri dir. SP içinde 3 değişken tanımlanmıştır. Bunlar @Rastgelesayi, @Maximum, @Minumum dur. @Minumum değişkenine banka tarafından belirlenen minumum üretilebilecek hesap numarası atanır. @Maximum değişkenine banka tarafından belirlenen maximum üretilebilecek hesap numarası atanır. @Rastgelesayi değişkenine @Minumum ve @Maximum değerleri arasında rastgele sayı üretilerek atanır. Diğer adımda while döngüsü ile tblVadesizHesaplar tablosundaki HesapNo alanındaki değer ile @Rastgelesayi değişkenindeki değer farklı olana kadar rastgele sayı üretilir.

Böylelikle üretilen hesap numarasının tabloda varlığı kontrol edilir. Sonuç farklı ise tblVadesizHesaplar tablosuna kayıt eklenir. SP @Rastgelesayi değerini döner.

- **SP_Vadesiz_Vadeli_ParaTransferi:** Vadeli hesaptan vadesiz hesaba veya vadesiz hesaptan vadeli hesaba para transferi işlemlerinde çağrılır. Dışarıdan 10 parametre alır. Bunlar @vadelivadesiz_vadesizvadeli, @borclu_musterino, @borclu_sube, @borclu_hesap, @borclu_dovizkodu, @alacakli_sube, @alacakli_hesap, @alacakli_dovizkodu, @tutar, @aciklama parametreleridir. @vadelivadesiz_vadesizvadeli parametresi işlemin vadeli hesaptan vadesiz'e veya vadesiz hesaptan vadeli hesaba olduğunu iletir. Eğer "0" ise Vadeli'den vadesiz'e para aktarımıdır. tblVadeliHesaplar tablosundaki Bakiye alanının değeri @tutar kadar düşülür. tblVadesizHesaplar tablosundaki Bakiye alanının değeri @tutar kadar arttırılır. tblHesapHareketleri tablosuna kayıt eklenir.

- **SP_WebGirisLogs:** İnternet Şube'ye müşteri girişi yapıldığında çağrılır. Dışarıdan @musterino, @durummesaji ve @IP adında 3 parametre alır. tblWebGirisLogs tablosuna bu parametreleri ekler.

- **SP_WebGSMSifreKontrol**: Müşteriye gönderilen GSM şifre kontrolü için çağrılır. Dışarıdan 4 parametre alır. Bunlar @MusteriNo, @IP, @Cyrpto, @CepSifre dir. "CepSifre" parametresine atanan değer tbl_ tblGSMSifre tablosundaki veri ile eşleşiyorsa 1, eşleşmiyorsa 0 değerini döner.

- **SP_WebGSMSifreUret**: Müşteri uygulamadan GSM şifre talep ettiğinde çağrılır. Dışarıdan 3 parametre alır. Bunlar @MusteriNo, @IP, @Cyrpto dur. SP içinde 4 değişken tanımlanmıştır. Bunlar @TarihSaat, @CepSifre, @Upper, @Lower dır. @ Lower ile @ Upper değişkenlerine atanan değerler arasında rastgele sayı oluşturulur ve bu sayı @CepSifre değişkenine atanır. Verileri tblGSMSifre tablosuna kaydeder.

- **SP_WebKullaniciOlustur :** Uygulamanın testi için yeni kullanıcı oluşturmada çağrılır.

- **SP_WebSifreKontrol**: Müşterinin İnternet Şubeye girerken çağrılır. Dışarıdan @MusteriNo ve @Sifre aında 2 parametre alır.SP @Sifre değişkenin'in değeri tbl_WebSifre tablosundaki veri ile eşleşiyorsa 1, eşleşmiyorsa 0 değerini döner.

5. WEB SERVİSLER

Veritabanından uygulamaya veri aktarımı ve uygulamadan veritabanına işlem yaptırımı için Web Service(WS)'ler oluşturulmuştur. Tez kapsamında geliştirilen WS'ler "Silverlight-enabled WCF Service" dir. WCF(Windows Communication Foundation) servis teknolojisidir ve Framework 3.0 ile birlikte gelmiştir.

WS ve Silverlight Enabled WCF Service arasında engine kapsamında ufak performans farkı dışında büyük bir veri trafiği performans farkı vardır. Silverlight Enabled WCF Service özel bir binary binding kullanır. Sunucu ile istemci arasındaki veri trafiğinde serileştirme ve deserileştirme işlemlerinde ara ürün olarak binary kullanıldığı için toplam veri trafiğindeki veri miktarı azalır ve binary serileştirme yapıldığı için serileştirme sürecinde harcanan işlemci yükü de düşük olur [16].

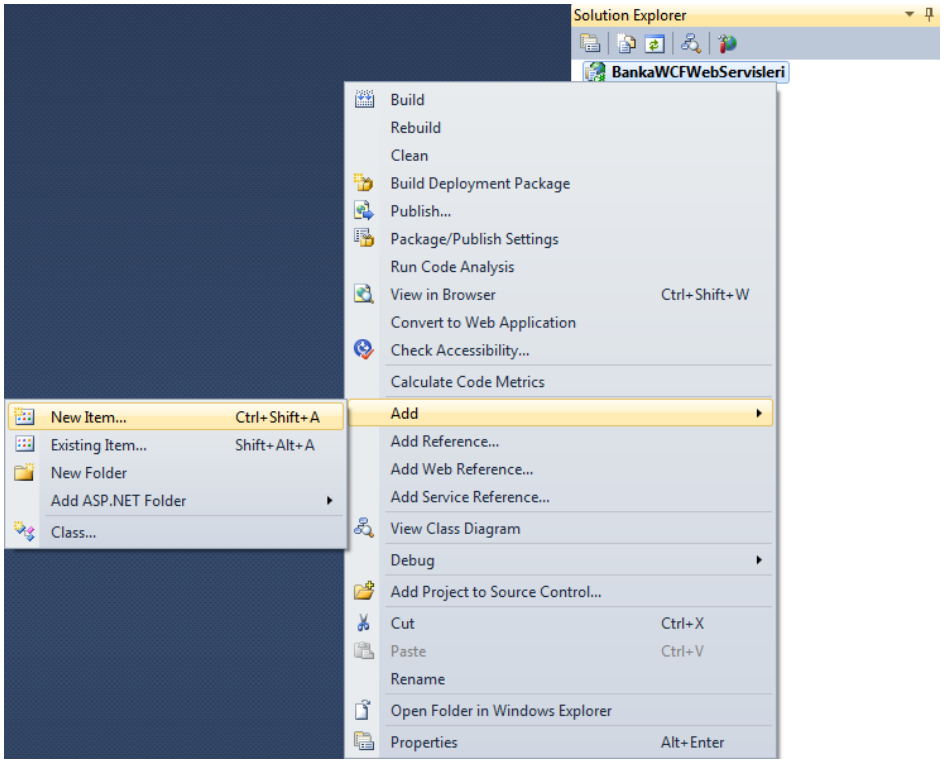
5.1. LINQ to SQL Classes

WS'lerin InternetSubev2 veritabanı ile bağlantı kurabilmesi için ara katman olarak VeriSiniflari.dbml adında LINQ to SQL Classes kullanılmıştır. Bu teknoloji sayesinde veritabanın'ın modeli otomatik olarak çıkartılmıştır. LINQ to SQL teknolojisinin temelini oluşturan DataContext sınıfı veritabanıındaki tabloları birer nesne, tabloya ait kolonları ise nesnenin özellikleri haline getirmektedir. WS içinde DataContext tanımlayarak veritabanı-ndaki Table, View, Stored Procedure nesnelerine kod yazarak ulaşılabilmektedir.

LINQ to SQL veritabanı ile iletişimi sağlayan ve çok basit cümlecikleri uygun sorgulara otomatik dönüştüren bir kütüphanedir [17].

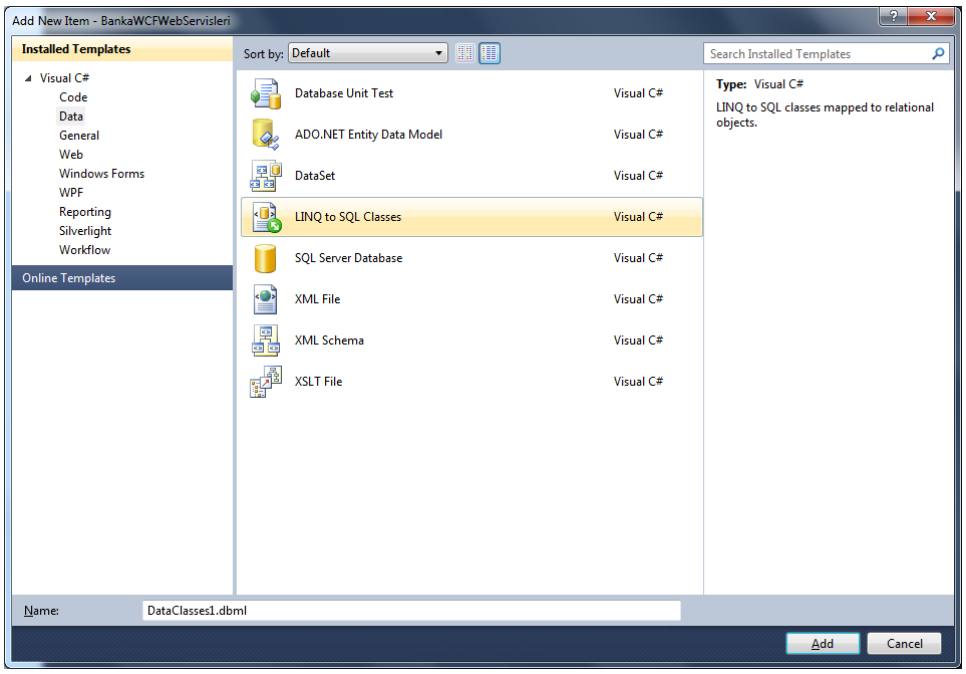
Aşağıda LINQ to SQL Classes'ın kullanımı Tez kapsamında oluşturulan vw_KrediKartlari View'i ve SP_NakitAvansKullan SP örnekleri ile anlatılmıştır.

1. Visual Studio(VS) Solution Explorer panelinden proje mouse ile sağ tıklanır ve "Add/New Item…" tıklanır.



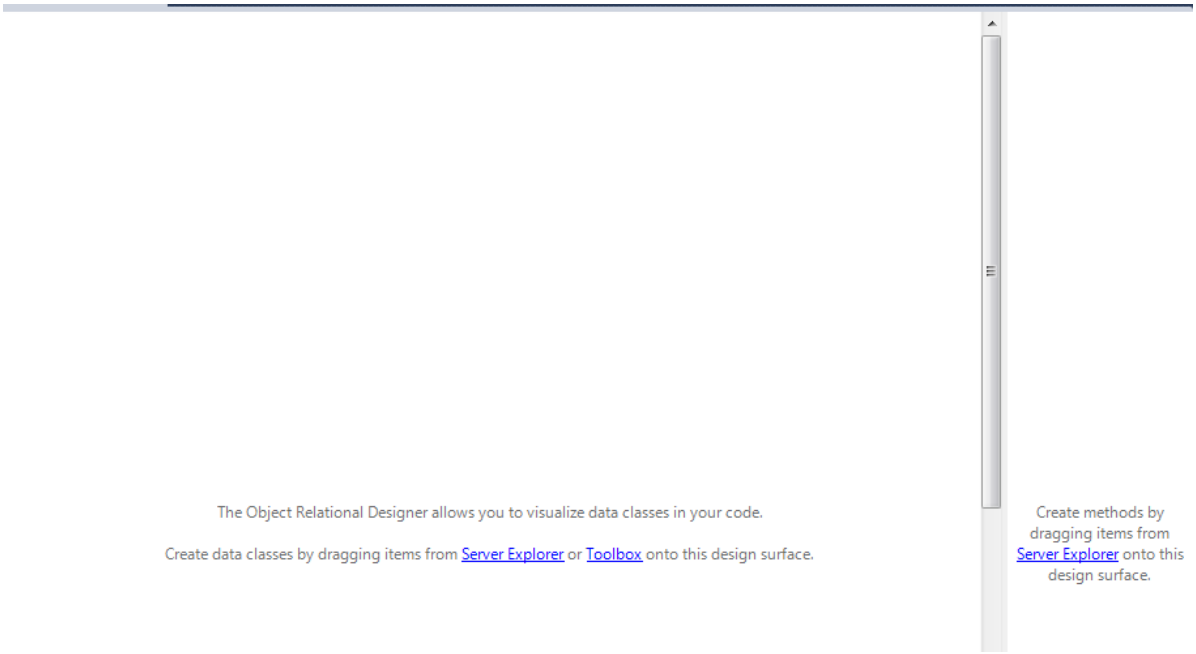
Şekil. 5.1. VS New Item

2. Gelen ekrandan LINQ to SQL Classes seçilir. Name alanına VeriSiniflari.dbml yazılır ve Add butonuna tıklanır.



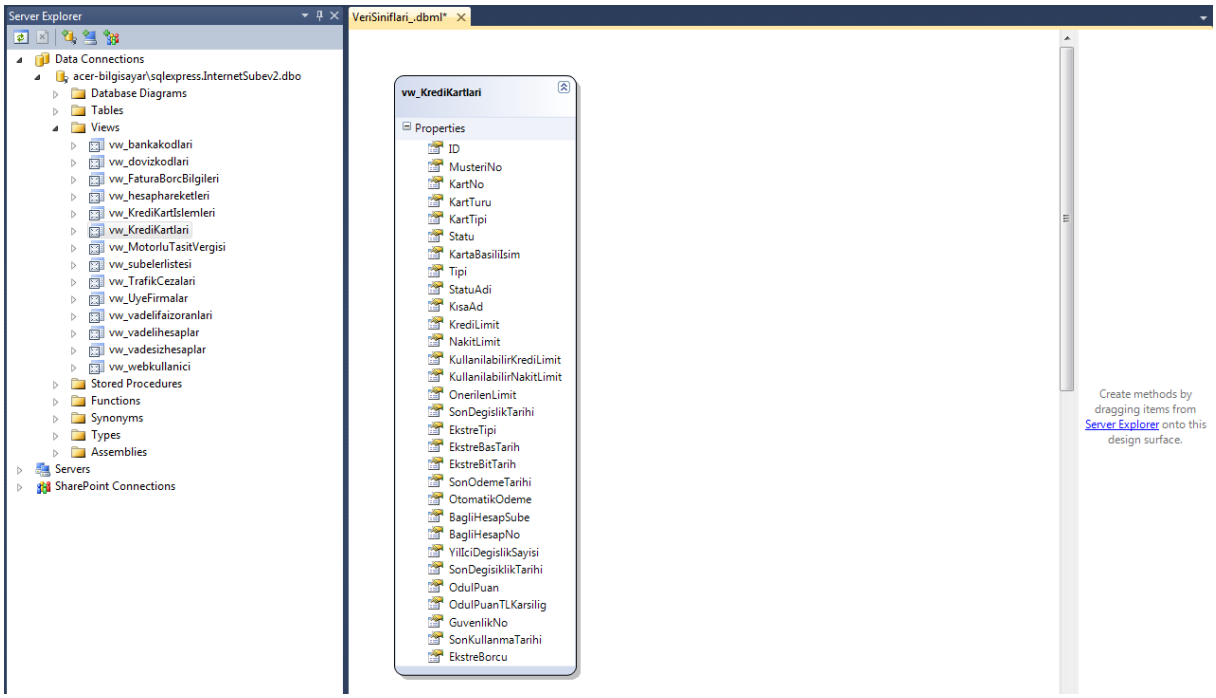
Şekil. 5.2. VS LINQ to SQL Classes Seçimi

3. Ekrana 2 bölüme ayrılmış pencere gelir. Bu pencerenin sol panelinde eklenen Table, View yapıları görünür. Sağ panelinde eklenen Stored Procedure yapıları görüntülenir.



Şekil. 5.3. VS LINQ to SQL Classes Panel

4. Table, View, Stored Procedure eklemek için VS'nun Server Explorer panelinden Data Connections ile veritabanına bağlanılır. Views sekmesi altından vw_KrediKartlari View'i sürüklenerek sol panele bırakılır. LINQ to SQL, vw_KrediKartlari View'ini nesneye dönüştürmektedir. Alanlarıda(ID, MusteriNo, KartNo, KartTuru vs…) bu nesnenin özellikleri olarak atamaktadır.



Şekil. 5.4. VS LINQ to SQL Classes vw_KrediKartlari

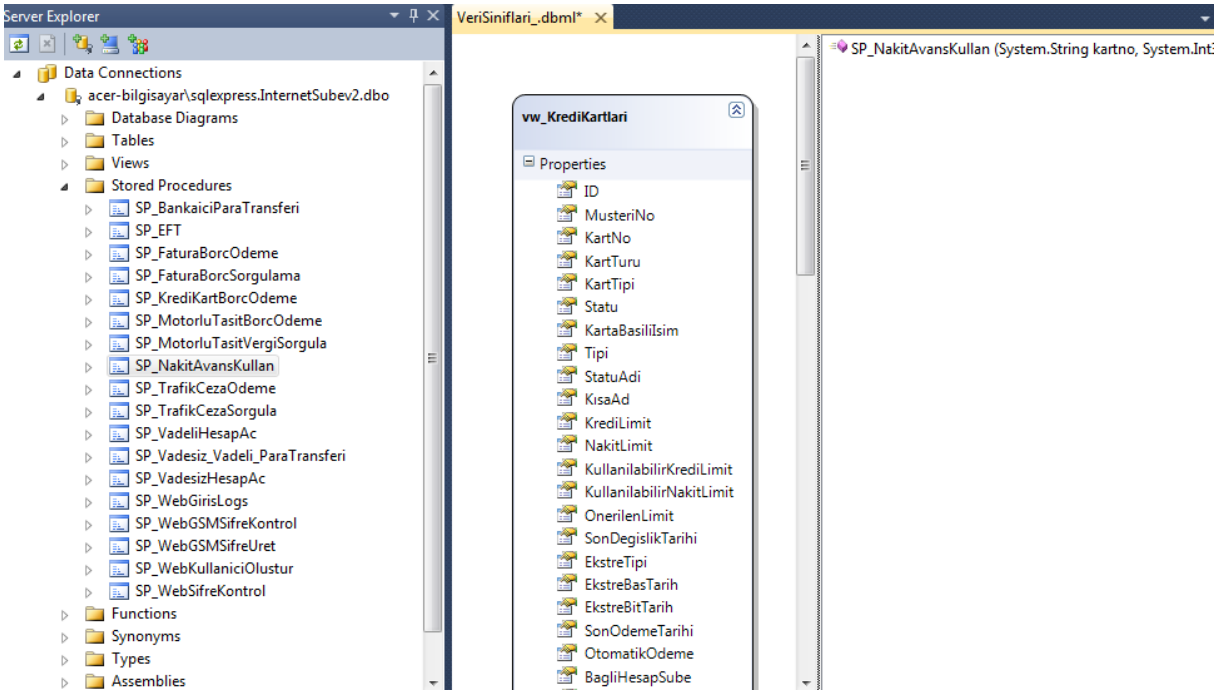
5. Solution Explorer panelinden VeriSiniflari.designer.cs tıklanırsa oluşturulan nesne ve özelliklerine erişilebilir. Burada VeriSiniflari adında DataContext oluşturulmuştur. Sonraki adımda vw_KrediKartlari adında class oluşturulmuştur.

```
#region Extensibility Method Definitions
partial void OnCreated();
#endregion

    public VeriSiniflari_DataContext() :
            base(global::System.Configuration.ConfigurationManager.ConnectionStrings["InternetSubev2ConnectionString'
    {
        OnCreated();
    }

    public VeriSiniflari_DataContext(string connection) :
            base(connection, mappingSource)
    {
        OnCreated();
    }

    public VeriSiniflari_DataContext(System.Data.IDbConnection connection) :
            base(connection, mappingSource)
    {
        OnCreated();
    }

    public VeriSiniflari_DataContext(string connection, System.Data.Linq.Mapping.MappingSource mappingSource) :
            base(connection, mappingSource)
    {
        OnCreated();
    }

    public VeriSiniflari_DataContext(System.Data.IDbConnection connection, System.Data.Linq.Mapping.MappingSource ma
            base(connection, mappingSource)
    {
        OnCreated();
    }

    public System.Data.Linq.Table<vw_KrediKartlari> vw_KrediKartlaris
    {
        get
        {
            return this.GetTable<vw_KrediKartlari>();
        }
    }
}

        [global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.vw_KrediKartlari")]
        public partial class vw_KrediKartlari
        {

            private int _ID;

            private System.Nullable<int> _MusteriNo;

            private string _KartNo;

            private System.Nullable<int> _KartTuru;

            private System.Nullable<int> _KartTipi;

            private System.Nullable<int> _Statu;

            private string _KartaBasiliIsim;

            private string _Tipi;

            private string _StatuAdi;

            private string _KısaAd;

            private string _KrediLimit;

            private System.Nullable<int> _NakitLimit;

            private System.Nullable<int> _KullanilabilirKrediLimit;

            private System.Nullable<int> _KullanilabilirNakitLimit;

            private System.Nullable<int> _OnerilenLimit;

            private System.Nullable<int> _SonDegislikTarihi;

            private System.Nullable<System.DateTime> _EkstreTipi;

            private string _EkstreBasTarih;

            private System.Nullable<System.DateTime> _EkstreBitTarih;

            private System.Nullable<System.DateTime> _SonOdemeTarihi;
```

```
private System.Nullable<int> _EkstreBorcu;

public vw_KrediKartlari()
{
}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="_ID", DbType="Int NOT NULL")]
public int ID
{
    get
    {
        return this._ID;
    }
    set
    {
        if ((this._ID != value))
        {
            this._ID = value;
        }
    }
}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="_MusteriNo", DbType="Int")]
public System.Nullable<int> MusteriNo
{
    get
    {
        return this._MusteriNo;
    }
    set
    {
        if ((this._MusteriNo != value))
        {
            this._MusteriNo = value;
        }
    }
}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="_KartNo", DbType="NVarChar(20)")]
public string KartNo
{
    get
    {
        return this._KartNo;
    }
    set
    {
```

Şekil. 5.5. VS LINQ to SQL Classes Kodlar

6. SP_NakitAvansKullan SP'ü eklemek için Server Explorer panelinden Stored Procedures sekmesi tıklanır ve SP_NakitAvansKullan mouse ile sürüklenerek sol veya sağ panele bırakılır. Gösterildiği yer sağ panel dir.



Şekil. 5.6. VS LINQ to SQL Classes SP_NakitAvansKullan

7. VeriSiniflari.designer.cs tıklandığında eklenen Stored Procedure'un parametre alan fonksiyon olarak tanımlandığı görülebilir.

```
[global::System.Data.Linq.Mapping.FunctionAttribute(Name="dbo.SP_NakitAvansKullan")]
public int SP_NakitAvansKullan([global::System.Data.Linq.Mapping.ParameterAttribute(DbType="NVarChar(20)")] string kartno
{
    IExecuteResult result = this.ExecuteMethodCall(this, ((MethodInfo)(MethodInfo.GetCurrentMethod())), kartno, tutar, al
    return ((int)(result.ReturnValue));
}
```

Şekil. 5.7. VS LINQ to SQL Classes SP_NakitAvansKullan Kodları

8. Son olarak VeriSiniflari.dbml LINQ to SQL Classes'ı kaydedilir.

5.2. Silverlight-enabled WCF Service

WCF servisinde kullanılacak methodlar OperationContract özelliğine sahip olurlar. OperationContract sınıfların metodlarının servis olarak kullanılmasını sağlar.

Aşağıda Tez kapsamında geliştirilen Web Servisler listelenmiştir.

- **Guvenlik.svc**: Güvenlik işlemlerini uygulayan fonksiyonlar tarafından çağrılmaktadır. 7 OperationContract'tan oluşur.
- **Hesaplar.svc**: Mevduat hesapları ile ilgili işlemlerde kullanılan fonksiyonlar tarafından çağrılmaktadır. 17 OperationContract'tan oluşur.
- **KrediKartlari.svc**: Kredi kartı işlemlerinde kullanılan fonksiyonlar tarafından çağrılmaktadır. 5 OperationContract'tan oluşur.
- **Odemeler.svc**: Firmalara veya kurumlara ödemelerde kullanılan fonksiyonlar tarafından çağrılmaktadır. 8 OperationContract'tan oluşur.
- **Sessions.svc**: Session işlemlerini uygulayan fonksiyonlar tarafından çağrılmaktadır. 4 OperationContract'tan oluşur.
- **Transferler.svc**: Transfer işlemlerinde kullanılan fonksiyonlar tarafından çağrılmaktadır. 4 OperationContract'tan oluşur.

5.3. Güvenlik İşlemleri Servisi ("Guvenlik.svc")

Aşağıda Web Servis'lerin kodları yanlarında açıklanarak listelenmiştir.

- "VeriSiniflariDataContext context = new VeriSiniflariDataContext();": Adı context olan yeni bir VeriSiniflariDataContext'i nesnesi oluşturulmaktadır.

- “[OperationContract]”: Fonksiyonun çağrılabilmesi OperationContract olduğu için belirtilmeli.

  “public bool SifreKontrol(int musterino, string sifre)”: Şifre kontrolü için çağrılır. Boolean(true, false) türünde veri dönmektedir. Parametre olarak musterino ve sifre parametrelerini almaktadır.

  {

  “string cyrptsifre = Cyrptola(sifre);”: sifre parametresi gönderilerek Cyrptola fonksiyonu çağrılmakta ve dönen değer cyrptsifre değişkenine atanmaktadır. Burada uygulamadan gelen müşterinin şifresi HASH1 algoritması ile HASH uygulanarak cyrptsifre değişkenine atama yapılmaktadır.

  “return (context.SP_WebSifreKontrol(musterino, cyrptsifre) == 1) ? true : false;”: if cümlesinin kısa halidir. Context DataContext’i aracılığıyla SP_WebSifreKontrol Stored Procedure’ü çağrılmaktadır. Eğer SP’nin döndüğü değer “1” ise fonksiyon “true” değerini döner, “1” den farklı ise “false” değerini döner.

  }

- [OperationContract]

  “public List<string> KullaniciTuru(int musterino)”: Kullanıcı türünü(Müşteri, Sistem Yönetim) döner. Uygulamada gösterilecek sayfaları belirler. String türünde değer döner. Parametre olarak musterino yu alır.

  {

  “List<string> temp = (from inc in context.vw_webkullanicis where inc.MusteriNo == musterino select inc.Kullanici).ToList();”: LINQ to SQL cümlesi ile vw_webkullanici tablosundan MusteriNo alanı musterino parametresi ile eşleşen kayıt’ın Kullanici alanının değerini temp değişkenine atar. “from inc in context.vw_webkullanicis” cümlesinde dikkat edilirse “vw_webkullanici” View’in sonuna “s” eklenmiştir. Burada inc değişkenine vw_webkullanici View’i geçici olarak atanmaktadır. “where” sql dilinde kullanılan koşul ifadesi dir. “where inc.MusteriNo == musterino” Burada MusteriNo alanı ile musterino parametresinin aynı olması koşulu var. Dikkat edilirse “=” ikitane dir.(==) “select inc.Kullanici” ile Kullanici alanın’ın değerleri dönmesi için seçilmektedir.

  “return temp;”: Fonksiyon temp değişkenindeki veriyi döner.

}

- [OperationContract]

  “public string Cyrptola(string plaintext)”: plaintext parametresine atanan değere HASH uygulayarak string türünde döner.

  {

  “return FormsAuthentication.HashPasswordForStoringInConfigFile(plaintext, "SHA1");” plaintext’e atanan değere SHA1 algoritması uygulanıyor ve fonksiyon bunu dönüyor.

  }

- [OperationContract]

  “public void GirisiLogla(int musterino, string IP, string durummesaji)”: 3 parametre alır. Bir değer dönmez. Giriş bilgilerini veritabanına kaydeder.

  {

  “context.SP_WebGirisLogs(musterino, durummesaji, IP);”: SP_WebGirisiLogs Stored Procedures’i çağrılmaktadır.

  }

- [OperationContract]

  “public string KullaniciOlustur(int musterino, string sifre, string kullanici)”: Uygulamanın test’i için kullanıcı oluşturur. String türünde değer döner.

  {

  “string temp = null;”: string türünde temp değişkeni tanımlanıyor ve null değeri atanıyor.

  try

  {

  “string cyrptsifre = Cyrptola(sifre);”: sifre parametresine HASH uygulanıyor.

  “context.SP_WebKullaniciOlustur(musterino, cyrptsifre, kullanici);”: SP çağrılıyor ve veri veritabanına kaydediliyor.

“temp = "Kullanıcı başarıyla tanımlandı.";”: işlem başarılı olursa temp değişkenine veri atanıyor.

}

catch

{

temp = "Hata oluştu."; işlem başarısız olursa temp değişkenine veri atanıyor.

}

return temp;

}

- [OperationContract]

  “public bool GSMSifreTalep(int musterino, string IP, string cyrpto)”: Boolean türünde veri dönmektedir. Dışarıdan 3 parametre alır.

  {

  “return (context.SP_WebGSMSifreUret(musterino, IP, cyrpto) == 0) ? true : false;”: SP çağrılıyor eğer dönen değer “0” ise fonksiyon “true” değerini dönecek, “0” dan farklı ise fonksiyon “false” değerini döner.

  }

- [OperationContract]

  “public bool GSMSifreKontrol(int musterino, string IP, string cyrpto, string GSMsifre)”: Boolean türünde veri dönmektedir. Dışarıdan 4 parametre alır.

  {

  “return (context.SP_WebGSMSifreKontrol(musterino, IP, cyrpto, GSMsifre) == 1) ? true : false;” SP çağrılıyor eğer dönen değer “1” ise fonksiyon “true” değerini dönecek, “1” dan farklı ise fonksiyon “false” değerini döner.

  }

5.4. Hesap İşlemleri Servisi (“Hesaplar.svc”)

Aşağıda Web Servis’in bazı kodları yanlarında açıklanarak listelenmiştir. Güvenlik İşlemleri Servisi ile aynı yapıda olan OperationContract’lar genel bilgileri verildiği için aynı yapıların kod açıklamaları bu bölümde ve sonraki bölümlerde tekrar yapılmamaktadır. OperationContract’ların gerçekleştirdikleri işlemler belirtilmektedir.

- VeriSiniflariDataContext context = new VeriSiniflariDataContext();

- [OperationContract]
  “public List<string> SubeSehirleri()”: Şube’lerin bulunduğu şehirleri listelemektedir.
  {
  “List<string> sonuc = (from inc in context.vw_subelerlistesis select inc.Il).Distinct().ToList();” vw_subelerlistesi View’inden il alanının değerlerini getirmektedir. “Distinct()” kullanılmasının nedeni dönen verilerdeki mükerrerliği engellemektir. Mükerrer verileri teke indirir. “ToList()” ile dönen değerler listelenmektedir.
  return sonuc;
  }

- [OperationContract]
  public List<string> SubeIlceleri(string il)
  {
  List<string> sonuc = (from inc in context.vw_subelerlistesis where inc.Il == il select inc.Ilce).Distinct().ToList();
  return sonuc;
  }
  Alınan il parametresi ile eşleşen şubelerin bulunduğu ilçeleri listeler.

- [OperationContract]
  public List<string> SubeMahalle(string il, string ilce)

{

List<string> sonuc = (from inc in context.vw_subelerlistesis where inc.Il == il && inc.Ilce == ilce select inc.Mahalle).Distinct().ToList();

return sonuc;

}

Alınan il ve ilçe parametresi ile eşleşen şubelerin bulunduğu mahalleleri listeler.

- [OperationContract]

  "public List<vw_subelerlistesi> Subeler(string il, string ilce, string mahalle)": vw_subelerlistesi türünde değer döner. Dönen değer vw_subelerlistesi View'i ile aynı yapıdadır. Burada LINQ to SQL'in sağladığı avantaj görülmektedir. Kendi oluşturduğu yapıya dönen verileri yerleştirerek değer döndürmektedir.

  {

  "List<vw_subelerlistesi> sonuc = (from inc in context.vw_subelerlistesis where inc.Il == il && inc.Ilce == ilce && inc.Mahalle == mahalle select inc).Distinct().ToList();": Burada "&&" işareti "ve" anlamına gelmektedir. vw_subelerlistesi tablosunda Il alanı ile il parametre değeri ve Ilce alanı ile ilce parametre değeri ve Mahalle alanı ile mahalle parametre değerleri aynı olan kayıtın dönmesi isteniyor.

  return sonuc;

  }

- [OperationContract]

  public List<vw_subelerlistesi> ButunSubeler()

  {

  List<vw_subelerlistesi> sonuc = (from inc in context.vw_subelerlistesis select inc).Distinct().ToList();

  return sonuc;

  }

  vw_subelerlistesi View'inden bütün şubelerin bilgilerini getirir. Dönen yapı vw_subelerlistesi yapısındadır.

- [OperationContract]
  public List<vw_dovizkodlari> DovizKoduListesi()
  {
  List<vw_dovizkodlari> sonuc = (from inc in context.vw_dovizkodlaris select inc).ToList();
  return sonuc;
  }
  vw_dovizkodlari View'inden döviz kodlarının listesini vw_dovizkodlari yapısında döner.

- [OperationContract]
  public int VadesizHesapAc(int MusteriNo, int SubeKodu, int DovizKodu, string Rumuz)
  {
  int temp = 0;
  temp = context.SP_VadesizHesapAc(MusteriNo, SubeKodu, DovizKodu, Rumuz);
  return temp;
  }
  Vadesiz hesap açma işleminde çağrılır. 4 parametre alır. Veritabanından SP_VadesizHesapAc Stored Procedure(SP)'ünü çağırır. Veritabanı SP işlemini bitirit döndüğü değer temp değişkenine aktarılır ve OperationContract çağrıldığında bu değeri döner. Eğer hata yoksa dönen değer "0" olur.

- [OperationContract]
  "public string VadeliHesapAc(int MusteriNo, int SubeKodu, int DovizKodu, string Rumuz, int tutar,int vadesuresi, vw_vadesizhesaplar borclu)": Vadeli hesap açma işleminde çağrılır. 7 parametre alır. Dikkate edilecek olursa vw_vadesizhesaplar yapısında parametre almaktadır. Bu da LINQ to SQL'in sağladığı avantajdır. Veritabanındaki View'i sınıfa dönüştürerek veri atanmasına olanak sağlamaktadır.
  {
  string hesapno = null;

string sonuc = null;

string hata = null;

“DateTime vadebas = DateTime.Today;”: vadebas adında değişken tanımlanıp, bugünün tarih ve saat’i atanıyor.

“string yarin = DateTime.Today.AddDays(1).DayOfWeek.ToString();”: yarin adında değişken tanımlanıyor. Bugünün tarihine 1 gün eklenerek değişkene atanıyor.

“if (yarin == "Saturday")”: yarin değişkenin’in değeri “Saturday” ise aşağıdaki işlemi yapması isteniyor. Burada ertesi günün Cumartesi olup olmadığının kontrolü yapılmaktadır.

“vadebas = DateTime.Today.AddDays(-1);”: vadebas değişkenine, bugünün tarihinden 1 eksiltilerek oluşturulan yeni tarih atanıyor.

“else if (yarin == "Sunday")”: yarin değişkenin’in değeri “Sunday” ise aşağıdaki işlemi yapması isteniyor. Burada ertesi günün Pazar olup olmadığının kontrolü yapılmaktadır.

“vadebas = DateTime.Today.AddDays(1);”: vadebas değişkenine, bugünün tarihinden 1 arttırılarak oluşturulan yeni tarih atanıyor.

“if (vadebas.AddDays(vadesuresi).DayOfWeek.ToString() == "Saturday")

hata = "Girilen vade günü Cumartesi gününe denk gelmektedir. Vade süresini 1 gün eksiltin.";”: Eğer vadebas değişkenine vadesuresi kadar gün eklendiğinde tarih “Saturday”(Cumartesi) oluyorsa hata değişkenine "Girilen vade günü Cumartesi gününe denk gelmektedir. Vade süresini 1 gün eksiltin." ataması yapılıyor.

“else if (vadebas.AddDays(vadesuresi).DayOfWeek.ToString() == "Sunday")

hata = "Girilen vade günü Pazar gününe denk gelmektedir. Vade süresini 1 gün arttırın.";”: Eğer vadebas değişkenine vadesuresi kadar gün eklendiğinde tarih “Sunday”(Pazar) oluyorsa hata değişkenine "Girilen vade günü Pazar gününe denk gelmektedir. Vade süresini 1 gün arttırın." ataması yapılıyor.

“if (hata == null)”: Eğer hata değişkeni boş ise aşağıdaki işlemi yapması isteniyor.

{

“try”: Aşağıdaki işlemlerden birinde hata alınmazsa devam edilmesini istiyor. Eğer hata alınırsa “catch” bloğunun yapılması isteniyor.

{

hesapno = context.SP_VadeliHesapAc(MusteriNo, SubeKodu, DovizKodu, Rumuz, vadebas, vadesuresi).ToString();

“context.SP_Vadesiz_Vadeli_ParaTransferi('1', borclu.MusteriNo, borclu.SubeKodu, borclu.HesapNo,borclu.DovizKodu,

SubeKodu,Convert.ToInt32(hesapno),DovizKodu, tutar, "Vadeli Hesap Açılışı");”: Veritabanından SP_Vadesiz_Vadeli_ParaTransferi Stored Procedure’ü çağrılıyor. İlk parametre “1” olarak iletiliyor. Anlamı vadesiz hesaptan vadeli hesaba para transferinin yapılacağıdır. İkinci parametre müşteri numarası diğer parametreler sırasıyla para çıkışı olacak şubenin kodu, para çıkışı olacak hesap numarası, para çıkışı olacak döviz kodu, para girişi olacak hesap numarası, para girişi olacak döviz kodu, tutar, açıklama dır.

“sonuc = hesapno + " nolu vadeli hesabınızın vade başlangıç tarihi: " + vadebas.ToShortDateString();”: sonuç değişkenine hesap numarası ve vade başlangıç tarihi atanıyor.

}

“Catch”: Eğer yukarıdaki işlem bloğunda bir hata olursa aşağıdaki bloğun çalıştırılması isteniyor.

{

“sonuc = "Bir hata oluştu. ";”: sonuc değişkenine "Bir hata oluştu. " mesajı atanıyor.

}

}

“else”: Eğer hata değişkeni boş değilse aşağıdaki işlemi yapması isteniyor.

“sonuc = hata;”: sonuc değişkenine hata değişkenindeki değerin ataması yapılıyor.

“return sonuc;”: Servis sonuc değişkenindeki değeri dönüyor.

}

- [OperationContract]

  public List<vw_vadesizhesaplar> VadesizHesaplariListele(int musterino, int subekodu, int dovizkodu)

  {

List<vw_vadesizhesaplar> sonuc = (from inc in context.vw_vadesizhesaplars where inc.MusteriNo == musterino && inc.SubeKodu == subekodu && inc.DovizKodu == dovizkodu select inc).ToList();
return sonuc;
}

3 parametre alır. vw_vadesizhesaplar View'inde musterino, subekodu, dovizkodu ile eşleşen kayıtları vw_vadesizhesaplar yapısında döner.

- [OperationContract]
  public List<vw_vadelihesaplar> VadeliHesaplariListele(int musterino, int subekodu, int dovizkodu)
  {
  List<vw_vadelihesaplar> sonuc = (from inc in context.vw_vadelihesaplars where inc.MusteriNo == musterino && inc.SubeKodu == subekodu && inc.DovizKodu == dovizkodu select inc).ToList();
  return sonuc;
  }

  3 parametre alır. vw_ vadelihesaplar View'inde musterino, subekodu, dovizkodu ile eşleşen kayıtları vw_ vadelihesaplars yapısında döner.

- [OperationContract]
  public List<vw_vadesizhesaplar> VadesizTLHesaplariListele(int musterino)
  {
  List<vw_vadesizhesaplar> sonuc = (from inc in context.vw_vadesizhesaplars where inc.MusteriNo == musterino && inc.DovizKodu == 100 select inc).ToList();
  return sonuc;
  }

  Tek parametre alır. vw_ vadesizhesaplar View'inde musterino ile eşleşen kayıtları vw_ vadelihesaplar yapısında döner.

- [OperationContract]

```
public List<vw_vadelihesaplar> VadeliTLHesaplariListele(int musterino)
{
List<vw_vadelihesaplar> sonuc = (from inc in context.vw_vadelihesaplars where inc.MusteriNo == musterino && inc.DovizKodu == 100 select inc).ToList();
return sonuc;
}
```

Tek parametre alır. vw_ vadelihesaplar View'inde musterino ile eşleşen ve döviz kodu 100(TL) olan kayıtları vw_ vadelihesaplar yapısında döner.

- ```
  [OperationContract]
  public List<vw_vadesizhesaplar> VadesizDovizHesaplariListele(int musterino)
  {
  List<vw_vadesizhesaplar> sonuc = (from inc in context.vw_vadesizhesaplars where inc.MusteriNo == musterino && inc.DovizKodu != 100 select inc).ToList();
  return sonuc;
  }
  ```

  Tek parametre alır. vw_ vadesizhesaplar View'inde musterino ile eşleşen ve döviz kodu 100(TL)'den farklı olan kayıtları vw_ vadesizhesaplar yapısında döner.

- ```
  [OperationContract]
  public List<vw_vadelihesaplar> VadeliDovizHesaplariListele(int musterino)
  {
  List<vw_vadelihesaplar> sonuc = (from inc in context.vw_vadelihesaplars where inc.MusteriNo == musterino && inc.DovizKodu != 100 select inc).ToList();
  return sonuc;
  }
  ```

  Tek parametre alır. vw_ vadelihesaplar View'inde musterino ile eşleşen ve döviz kodu 100(TL)'den farklı olan kayıtları vw_ vadelihesaplar yapısında döner.

- ```
  [OperationContract]
  ```
  ```

```
public List<vw_hesaphareketleri> HesapHareketleri(int subekodu, int hesapno,DateTime bastrh, DateTime bittrh)
{
List<vw_hesaphareketleri> sonuc = (from inc in context.vw_hesaphareketleris where inc.SubeKodu == subekodu && inc.HesapNo == hesapno && inc.TarihSaat>bastrh && inc.TarihSaat<bittrh select inc).ToList();
return sonuc;
}
```

4 parametre alır. Bunlar işlemlerin listeleneceği hesabın şube kodu, hesap numarası ve başlangıç tarihi ve bitiş tarihi dir. İşlemin alınan 2 tarih parametresi arasında gerçekleşmiş olması gerekir. OperationContract vw_hesaphareketleri yapısıyla döner.

- ```
  [OperationContract]
  public List<vw_vadesizhesaplar> ButunVadesizHesaplariListele(int musterino)
  {
  List<vw_vadesizhesaplar> sonuc = (from inc in context.vw_vadesizhesaplars where inc.MusteriNo == musterino select inc).ToList();
  return sonuc;
  }
  ```

  Tek parametre alır. vw_ vadesizhesaplar View'inde musterino ile eşleşen kayıtları vw_ vadesizhesaplar yapısında döner.

- ```
  [OperationContract]
  public List<vw_vadesizhesaplar> HariciVadesizHesaplariListele(int musterino,int hesapno,int dovizkodu)
  {
  List<vw_vadesizhesaplar> sonuc = (from inc in context.vw_vadesizhesaplars where inc.MusteriNo == musterino && inc.HesapNo!=hesapno && inc.DovizKodu==dovizkodu select inc).ToList();
  return sonuc;
  }
  ```

3 parametre alır. vw_ vadesizhesaplar View'inde musterino ve döviz kodu ile eşleşen , hesap numarası hesapno parametresinden farklı olan kayıtları vw_ vadesizhesaplar yapısında döner.

5.5. Kredi Kartları İşlemleri Servisi ("KrediKartlari.svc")

- VeriSiniflariDataContext context = new VeriSiniflariDataContext();

- [OperationContract]
  public List<vw_KrediKartlari> KrediKartlariniListele(int musterino)
  {
  List<vw_KrediKartlari> sonuc = (from inc in context.vw_KrediKartlaris where inc.MusteriNo == musterino select inc).ToList();
  return sonuc;
  }
  Alınan musterino parametresi ile eşleşen kredi kartlarını listeler.

- [OperationContract]
  public List<vw_KrediKartlari> KartBilgileriniAl(string kartno)
  {
  List<vw_KrediKartlari> sonuc = (from inc in context.vw_KrediKartlaris where inc.KartNo == kartno select inc).ToList();
  return sonuc;
  }
  Alınan kartno parametresi ile eşleşen kredi kartının bilgilerini vw_KrediKartlari yapısında döner.

- [OperationContract]
  public string NakitAvansKullan(string kartno, int tutar, int alacaklisube, int alacaklihesap, int kartguvenlikno, DateTime sonkullanmatarihi)

```
{
int sonuc;
string temp = null;
sonuc = context.SP_NakitAvansKullan(kartno, tutar, alacaklisube,
alacaklihesap,kartguvenlikno,sonkullanmatarihi);
if (sonuc == 0)
temp = "İşlem başarılı.";
else if (sonuc == 99)
temp = "Kart Limiti yetersiz.";
else if (sonuc == 75)
temp = "Kartın Güvenlik Numarası Yanlış";
else if (sonuc == 23)
temp = "Kartın Son Kullanma Tarihi Yanlış";
return temp;
}
```

Kredi kartından nakit avans kullanmak için çağrılır.

- ```
  [OperationContract]
  public string BorcOdeme(string kartno, int tutar, int borclusube, int  borcluhesap)
  {
  int sonuc;
  string temp = null;
  sonuc = context.SP_KrediKartBorcOdeme(kartno, borclusube, borcluhesap, tutar);
  if (sonuc == 0)
  temp = "İşlem başarılı.";
  else if (sonuc == 99)
  temp = "Hesap Bakiyesi Yetersiz.";
  return temp;
  }
  ```

  Kredi kartı borç ödeme işlemi için çağrılır.
  ```

- [OperationContract]
  public List<vw_KrediKartIslemleri> Islemler(string kartno, DateTime bastrh, DateTime bittrh)
  {
  List<vw_KrediKartIslemleri> sonuc = (from inc in context.vw_KrediKartIslemleris where inc.KartNo == kartno && bastrh <= inc.Tarih && bittrh >= inc.Tarih select inc).ToList();
  return sonuc;
  }
  Alınan kartno parametresi ile vw_KrediKartIslemleri View'inde eşleşen kayıtların Tarih değeri bastrh ile bittrh parametreleri arasında olan kayıtları vw_KrediKartIslemleri yapısıyla döner.

5.6. Ödeme İşlemleri Servisi ("Odemeler.svc")

- VeriSiniflariDataContext context = new VeriSiniflariDataContext();

- "Transferler transfer = new Transferler();": Yeni bir transfer nesnesi oluşturuluyor. Transferler Web Servis Odemeler Web Servisi içinde nesne olarak kullanılmaktadır.

- [OperationContract]
  public List<vw_UyeFirmalar> FirmaListesi(string firmaturu)
  {
  List<vw_UyeFirmalar> sonuc = (from inc in context.vw_UyeFirmalars where inc.FirmaTuru == firmaturu orderby inc.FirmaAdi select inc).ToList();
  return sonuc;
  }
  Alınan firmaturu parametresi ile eşleşen firmaların listesi vw_UyeFirmalar yapısıyla dönülür.

- [OperationContract]
  public List<vw_UyeFirmalar> FirmaBilgileri(int firmakodu)
  {
  List<vw_UyeFirmalar> sonuc = null;
  sonuc = (from inc in context.vw_UyeFirmalars where inc.FirmaKodu == firmakodu select inc).ToList();
  return sonuc;
  }
  Alınan firmakodu parametresi ile eşleşen firmanın bilgileri vw_UyeFirmalar yapısıyla dönülür.

- [OperationContract]
  public List<vw_FaturaBorcBilgileri> FaturaBorcSorgulama(int firmakodu, string aboneno)
  {
  List<vw_FaturaBorcBilgileri> sonuc = context.SP_FaturaBorcSorgulama(firmakodu, aboneno).ToList();
  return sonuc;
  }
  Alınan firmakodu ve aboneno parametreleri ile eşleşen abonenin fatura borç bilgileri vw_ FaturaBorcBilgileri yapısıyla dönülür.

- [OperationContract]
  public string FaturaBorcOdeme(int id, int tutar, int borclumustno, int alacaklisb, int alacaklihsp, int borclusb, int borcluhsp)
  {
  string sonuc = null;
  int donen = context.SP_FaturaBorcOdeme(id, tutar);
  if (donen == 0)
  {
  sonuc = "İşlem başarıyla gerçekleşti.";

“transfer.Havale(borclumustno, borclusb, borcluhsp,100, alacaklisb, alacaklihsp,100, tutar, "Fatura Ödeme");”: Transfer nesnesindeki Havale fonksiyonu çağrılıyor. İlk ödeme işlemini yapacak borçlu müşteri numarası parametre olarak iletiliyor sonra sırasıyla, borçlu şube, borçlu hesap, borçlu hesap döviz tipi 100(TL), alacaklı firmanın hesabının bulunduğu şube kodu, alacaklı hesap numarası, alacaklı hesap döviz tipi 100(TL), tutar, açıklama parametreleri atanıyor.

```
}
else
sonuc = "Ödenecek tutar borç dan farklı.";
return sonuc;
}
```

Fatura borç ödeme işlemi için çağrılır. Eğer veritabanından çağrılan SP_FaturaBorcOdeme Stored Procedure’ü işleminde hata olmayıp “0” değerini dönerse OperationContract’dan dönen değer "İşlem başarıyla gerçekleşti." dir. Eğer SP’de hata olup “0” dan farklı bir değer dönerse OperationContract’dan dönen değer "Ödenecek tutar borç dan farklı." dır.

-
```
[OperationContract]
public List<vw_TrafikCezalari> TrafikCezaSorgulama(int cezadonemi, string plaka)
{
List<vw_TrafikCezalari> sonuc = context.SP_TrafikCezaSorgula(cezadonemi, plaka).ToList();
return sonuc;
}
```

Alınan cezadonemi ve plaka parametreleri ile eşleşen vw_TrafikCezalari View’inden vw_TrafikCezalari yapısıyla kayıtlar dönülür.

-
```
[OperationContract]
public string TrafikCezaOdeme(int id, int tutar, int borclusube, int borcluhesap, string borcluIBAN)
{
string sonuc = null;
```

```
int donen = context.SP_TrafikCezaOdeme(id, tutar);
if (donen == 0)
{
sonuc = "İşlem başarıyla gerçekleşti.";
transfer.EFT(borclusube, borcluhesap, borcluIBAN, "", 555, "", "899", "88899", "", "",
"", id.ToString() + " Nolu Trafik Cezası Ödeme", tutar);
}
else
sonuc = "Ödenecek tutar borcu dan farklı.";
return sonuc;
}
```

Trafik cezası ödeme işlemi için çağrılır.

- ```
  [OperationContract]
  public List<vw_MotorluTasitVergisi> MotorluTasitBorcSorgulama(int yil, string plaka)
  {
  List<vw_MotorluTasitVergisi> sonuc = context.SP_MotorluTasitVergiSorgula(yil, plaka).ToList();
  return sonuc;
  }
  ```

  Alınan yil ve plaka parametresi ile vw_MotorluTasitVergisi View'inde eşleşen kayıtlar dönülür.

- ```
  [OperationContract]
  public string MotorluTasitVergiOdeme(int id, int tutar, int borclusube, int borcluhesap, string borcluIBAN)
  {
  string sonuc = null;
  int donen = context.SP_MotorluTasitBorcOdeme(id, tutar);
  if (donen == 0)
  {
  ```

sonuc = "İşlem başarıyla gerçekleşti.";
transfer.EFT(borclusube, borcluhesap, borcluIBAN, "", 555, "", "899", "777555", "", "", "", id.ToString() + " Nolu Motorlu Taşıt Vergisi Ödeme", tutar);
}
else
sonuc = "Ödenecek tutar borcu dan farklı.";
return sonuc;
}
Motorlu taşıt vergi ödemesi yapılmak için çağrılır.

5.7. Sessions Servisi (“Sessions.svc”)

- [OperationContract]
  public void SessionAta(string sessionicerik, string sessionID)
  {
  “System.Web.HttpContext.Current.Session[sessionID] = sessionicerik;”: Bu kodla serverda session oluşturulur. Session’ın adı sessionID parametresi, değeri sessionicerik parametresidir.
  }

- [OperationContract]
  public string SessionGetir(string sessionID)
  {
  “return System.Web.HttpContext.Current.Session[sessionID].ToString();”: Adı sessionID parametresinin değeri olan session’ın içeriği string olarak dönülür.
  }

- [OperationContract]
  public void SessionSil()
  {

“System.Web.HttpContext.Current.Session.Abandon();”: Oluşturulan session silinir.

}

- [OperationContract]
  public string IPGetir()
  {
  “return
  System.Web.HttpContext.Current.Request.ServerVariables["REMOTE_HOST"];”: Kullanıcının IP adresi dönülür.
  }

5.8. Transfer İşlemleri Servisi (“Transferler.svc”)

- VeriSiniflariDataContext context = new VeriSiniflariDataContext();

- [OperationContract]
  public string Virman(int musterino,int borclusube,int borcluhesap,int borcludoviz,int alacaklisube,int alacaklihesap,int alacaklidoviz,int tutar,string aciklama)
  {
  “int sonuc = context.SP_BankaiciParaTransferi(musterino, borclusube, borcluhesap, borcludoviz, alacaklisube, alacaklihesap, alacaklidoviz, tutar, aciklama, 150);”: Veritabanındaki SP_BankaiciParaTransferi Stored Procedure’ü çağrılır. En son iletilen parametre değeri 150’nin anlamı virman dır. İşlem kodları tblIslemTurleri tablosunda dır.
  “if (sonuc == 99)”: Eğer SP’den gelen değer 99 ise OperationContract “Bakiye yetersiz” değerini döner. 99 dan farklı ise “İşlem başarılı” değeri dönülür.
  return "Bakiye yetersiz.";
  else
  return "İşlem başarılı";
  }

- [OperationContract]
  public string Havale(int musterino, int borclusube, int borcluhesap, int borcludoviz, int alacaklisube, int alacaklihesap, int alacaklidoviz, int tutar, string aciklama)
  {
  int sonuc = context.SP_BankaiciParaTransferi(musterino, borclusube, borcluhesap, borcludoviz, alacaklisube, alacaklihesap, alacaklidoviz, tutar, aciklama, 150);
  if (sonuc == 99)
  return "Bakiye yetersiz.";
  else
  return "İşlem başarılı";
  }
  Havale işlemi için çağrılır.

- [OperationContract]
  public List<vw_bankakodlari> BankalariListele()
  {
  List<vw_bankakodlari> sonuc = (from inc in context.vw_bankakodlaris select inc).ToList();
  return sonuc;
  }
  vw_bankakodlari View'indeki bütün kayıtları vw_bankakodlari yapısıyla döner.

- [OperationContract]
  public string EFT(int borclusube, int borcluhesap, string borcluIBAN, string borclutelefon, int alacaklibankakodu, string alacakliIBAN, string alacaklisube, string alacaklihesap, string alacakliadsoyad, string alacakliadres, string alacaklitelefon, string aciklama, int tutar)
  {
  string donen = null;

```
int sonuc = context.SP_EFT(borclusube, borcluhesap, borcluIBAN, borclutelefon, alacaklibankakodu, alacakliIBAN, alacaklisube, alacaklihesap, alacakliadsoyad, alacakliadres, alacaklitelefon, aciklama, tutar);
if (sonuc == 0)
donen = "İşlem başarıyla gerçekleşmiştir.";
else if (sonuc == 99)
donen = "Bakiye Yetersiz.";
else
donen = "Bir hata oluştu.";
return donen;
}
```

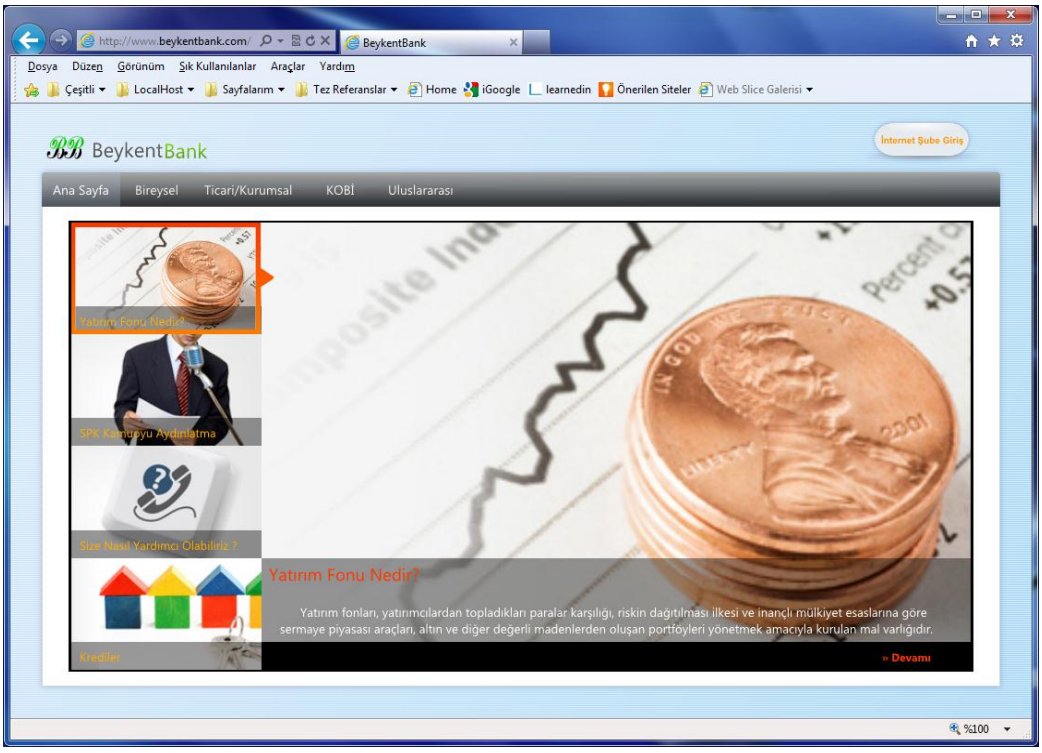
EFT işlemi için çağrılır.

6. PROGRAM ÇIKTILARI

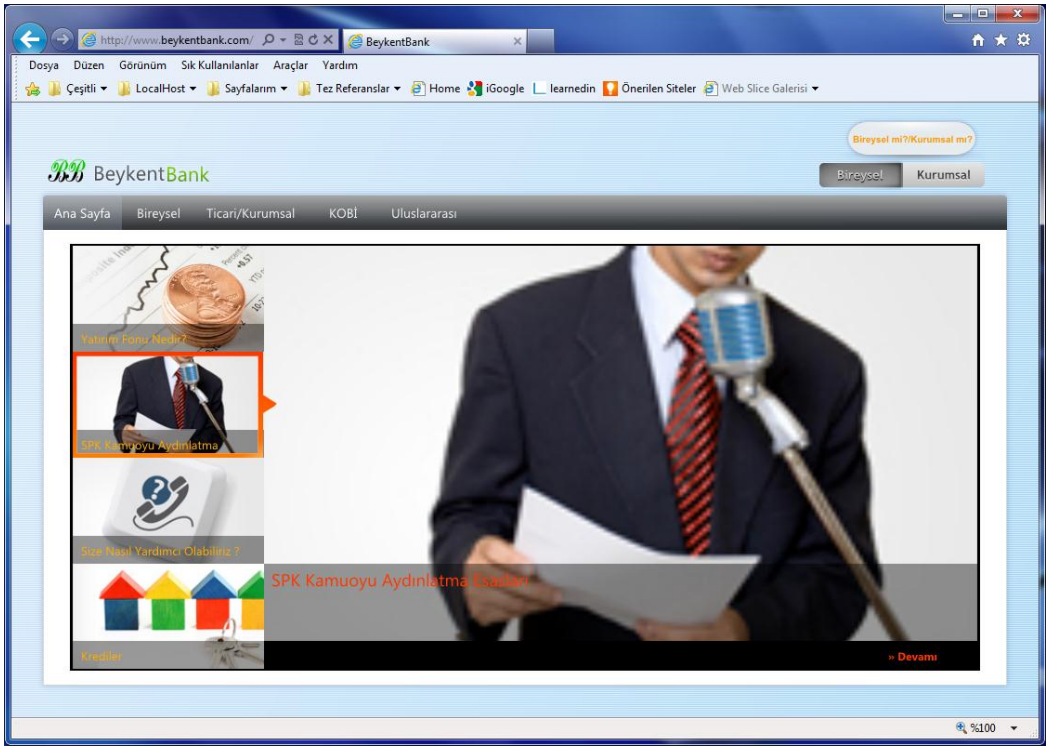
6.1. Ana Sayfa ve İnternet Şube Girişi

İnternet kullanıcılarının en çok istedikleri sayfaların sade ve kolay anlaşılabilir olmasıdır. Aranan bilgiye kısa yoldan nasıl erişilir sorusu kullanıcıların kafasında hep vardır. Silverlight teknolojisi ile yapılan sayfalarda hızlı erişim ve kullanım kolaylığı sağlayan birçok yapı yapılabilmektedir. Tez kapsamında geliştirilen uygulamada Silverlight teknolojisinin sağladığı Tool(araç)'lar sayesinde kullanım kolaylığı sağlayan ve hızlı sayfalar yapılmıştır.

İnternet Şube'ye girmek için sağ üst taraftaki "İnternet Şube Giriş" butonun'un üstüne gelindiğinde Bireysel kullanıcı ve Kurumsal kullanıcı girişi olarak 2 seçenek çıkmaktadır. "Bireysel" tıklanarak bireysel kullanıcı işlemleri bölümüne yönlenilir.



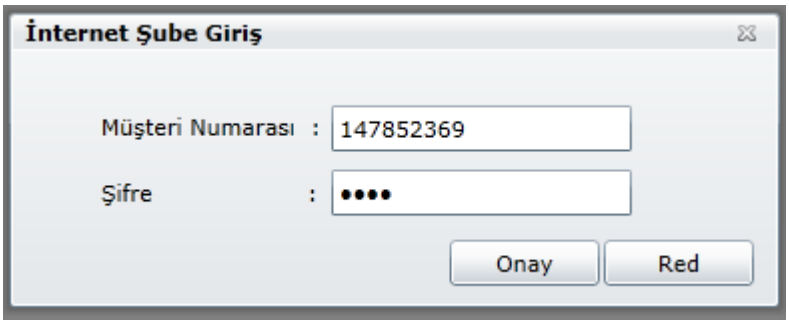
Şekil. 6.1. BeykentBank Ana Sayfa



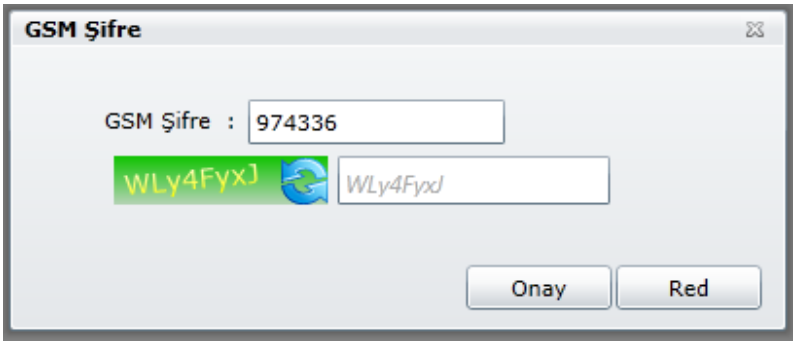
Şekil. 6.2. BeykentBank Bireysel Müşteri Seçim

6.2. İnternet Şube Giriş Ekranı

Silverlight sayesinde ”Bireysel” butonuna tıklanınca sayfa başka bir adrese yönlenmeden hemen giriş paneli ekrana gelir. Bu işlem Silverlight kullanılmadan javascript(JS) veya jquey gibi script teknolojileri ile HTML sayfasında da yapılabirdi. Fakat hem görsellik için çok fazla script kodu yazılmalı hemde bilgilerin Web Service(WS)’e iletilip gelen bilgilerin kontrolü bu dillerle hem çok zor olur hemde çok zaman alırdı. Silverlight sayesinde daha az kod yazarak bu işlem yapılır. Banka müşteri numarası ve İnternet Şube giriş şifresi girilerek “Onay” butonuna basılıp sonraki ekrana “GSM Şifre” ekranına geçilir. Kullanıcının bankada kayıtlı cep telefonuna 6 haneli ve her girişte rastgele oluşturulan rakamlardan oluşan bir şifre gönderili. Müşterinin gönderilen şifreyi girmesi istenir. Ek güvenlik için captcha(Completely Automated Public Turing test to tell Computers and Humans Apart)’e yönetimi kullanılarak ekranda rastgele bir yazı belirir ve kullanıcının bu yazıyı kutuya doğru girmesi istenir. Captcha’nin kullanılmasının sebebi otomatik sistemlerin sayfayı test etmelerini engellemektir.



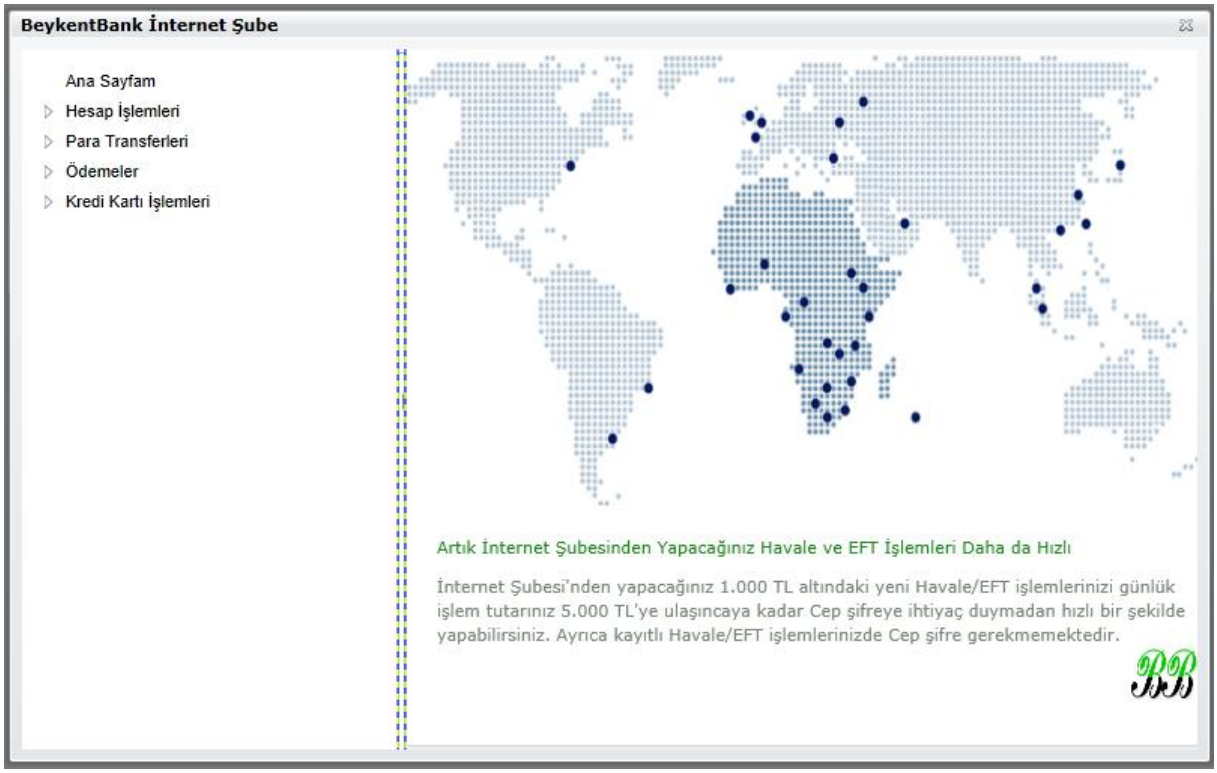
Şekil. 6.3. İnternet Şube İlk Giriş Ekranı



Şekil. 6.4. İnternet Şube Cep Şifre Ekranı

6.3. İnternet Şube Ana Sayfa

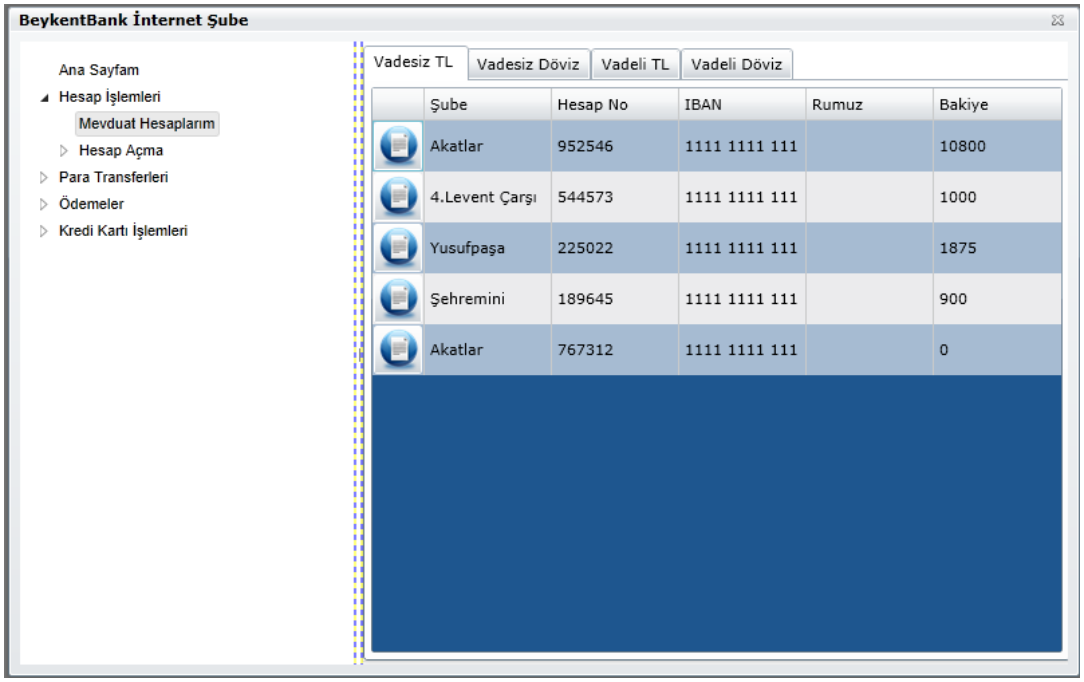
Sayfa, kullanıcıların fonksiyonel olarak kullanması için basit tasarlanmıştır. Kullanıcı hangi işlemi yapmak istiyorsa ilgili kategoriyi ekranın yan tarafında görmektedir. Karmaşık olmaması için Treeview(Ağaç) yapı kullanılmıştır. Kategoriler tıklandığında kapalı ise aşağı doğru açılmakta, kapalı ise yukarı doğru kapanmaktadır. Kategori paneli ile içerik paneli arasındaki ayraç mouse ile tutularak bölümlerin genişlikleri ayarlanabilir. Eğer mouse'un sağ tuşu ile tıklanırsa yan taraftaki kategori paneli kapanarak içerik bölümü ekranı kaplar. Tekrar ayraç mouse ile sağ tıklanırsa kategori eski genişlik boyutunda açılır.



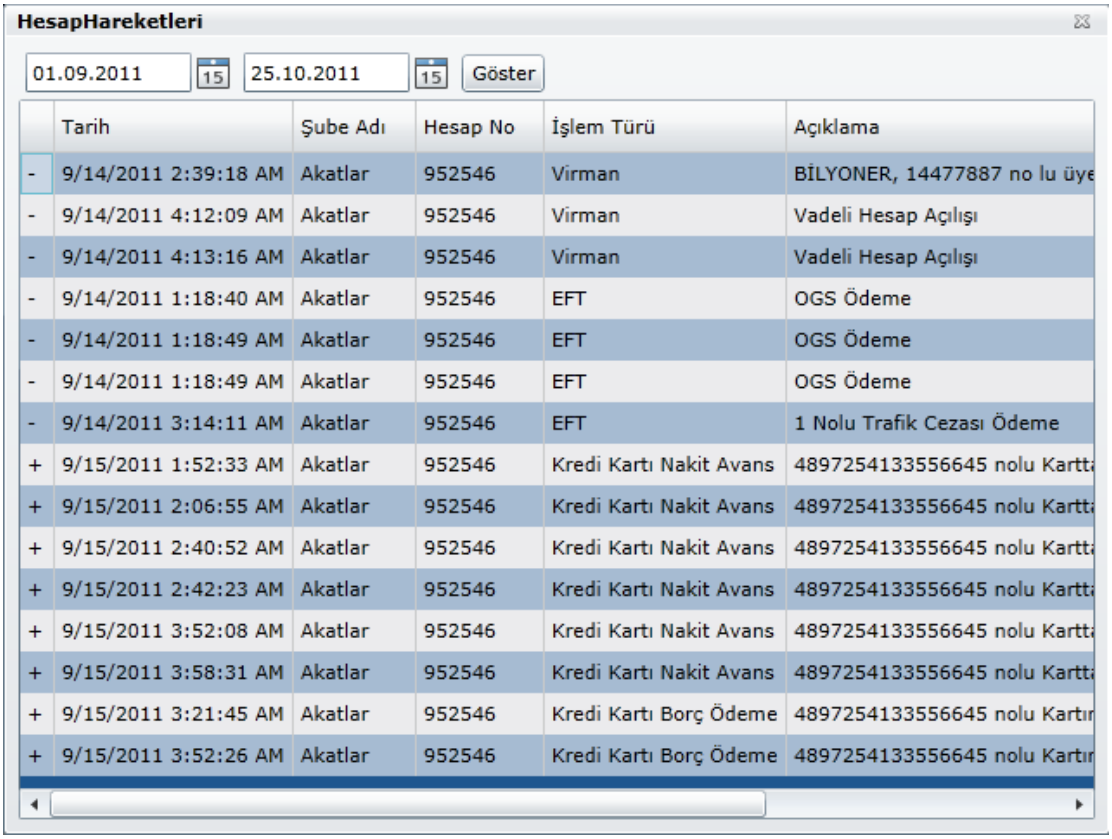
Şekil. 6.5. İnternet Şube Ana Sayfa

6.4. Hesap İşlemleri/Mevduat Hesaplarım Ekranı

Müşterilerin mevduat hesap(vadeli,vadesiz,döviz hesapları) bilgilerini hızlı ve kolay ulaşabilmesi için Silverlight teknolojisinin TabControl aracı herhangibir kod yazılmadan sürükle bırak ile sayfaya eklenerek kullanılmıştır. Diğer script dillerinde(JS, Jquery vs..) bu yapıyı hazırlamak için çok fazla kod yazılmalı ve çok fazla tanımlamalar yapılmalıdır. Ayrıca bu yapı Adobe firmasının ürünü olan Flash ile yapılmak istendiğinde, ekran için frame'ler hazırlanmalı, actionscript dilinde kod yazılmalı ve bunlar web sayfası ile entegre edilmelidir. Bu seçim çok zahmetli ve çok maliyetli olurdu. Vadesiz Döviz tıklanarak Vadesiz Döviz hesapları izlenebilir. Hesap haraketlerini görmek için ilk sütundaki buton tıklanarak ilgili hesaptan yapılan işlemler listelenir(para girişi, para çıkışı).



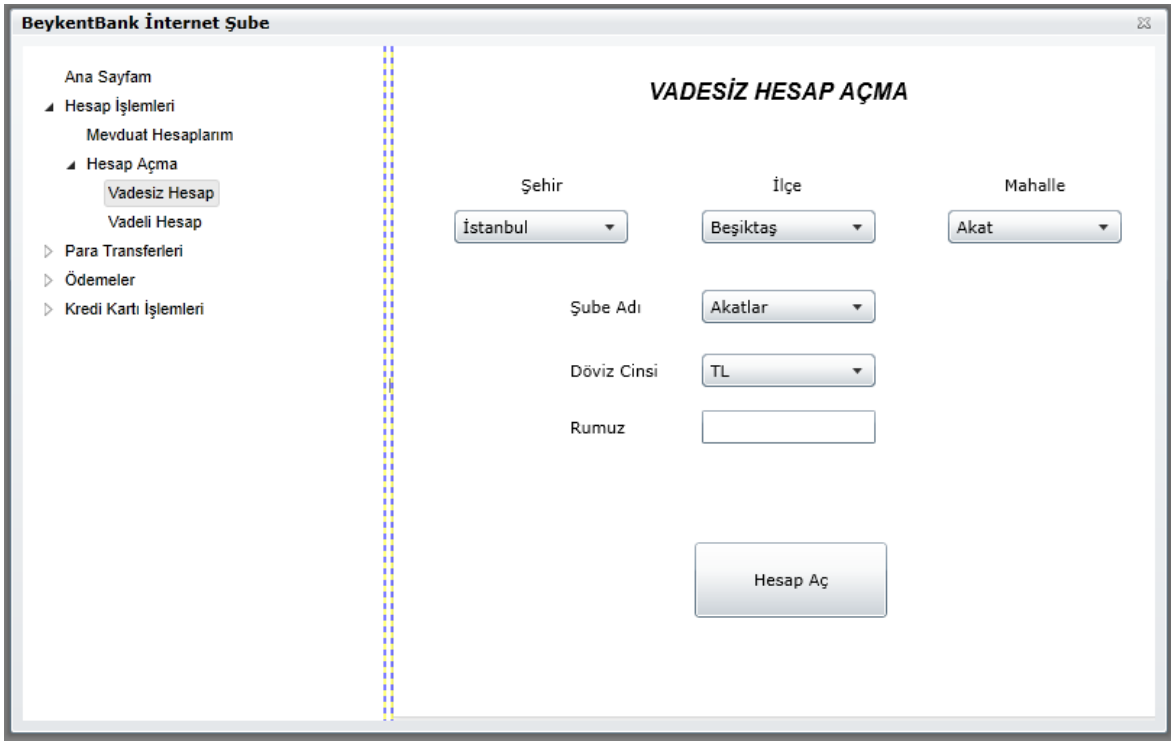
Şekil. 6.6. Mevduat Hesaplarım Ekranı



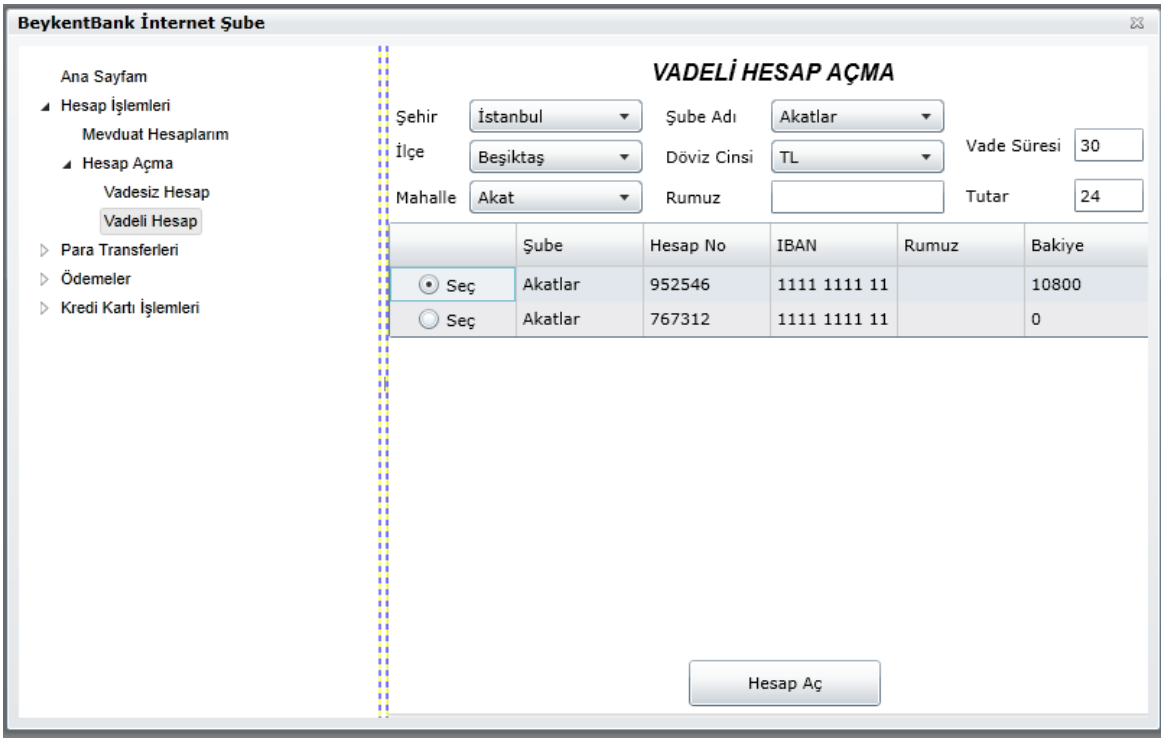
Şekil. 6.7. Mevduat Hesabı Ekstre Ekranı

6.5. Hesap İşlemleri /Vadeli - Vadesiz Hesap Açma Ekranları

Kullanılan araçlar Silverlight'ın sunduğu klasik araçlardır. Bu araçlar VS Toolbox panelinden sürükle bırak ile sayfaya eklenmektedir. Kod sayfasından ilgili kısa kodlar yazılarak nesneler kullanılır. Diğer script dillerinde bu yapıda combobox'lar yapılamaz. Flash ile yapılabilir. Fakat nesne tasarımı yapılmalı, gerekli kodlar actionscript ile yazılmalı ve nesne web sayfasına entegre edilmeli. Bu hem zahmetli bir iştir hemde maliyetli dir. Vadesiz hesap açma işleminde, hesabın açılacağı şube'nin bulunduğu şehir, ilçe, mahalle, şube adı, döviz cinsi seçilir. Rumuz tanımlanmak isteniyorsa textbox'a girişi yapılır. Hesap Aç butonuna tıklanarak işlem tamamlanır. Vadeli hesap açma ekranında vadesiz'dekine ek olarak döviz cinsi seçilince aşağıda seçilen döviz cinsinden bulunan vadesiz hesaplar listelenir. Vadeli hesaba paranın aktarılacağı vadesiz hesap seçilir, aktarılacak tutar ve vade süresi girilerek Hesap Aç butonuna tıklanarak işlem tamamlanır.



Şekil. 6.8. Vadesiz Hesap Açma Ekranı

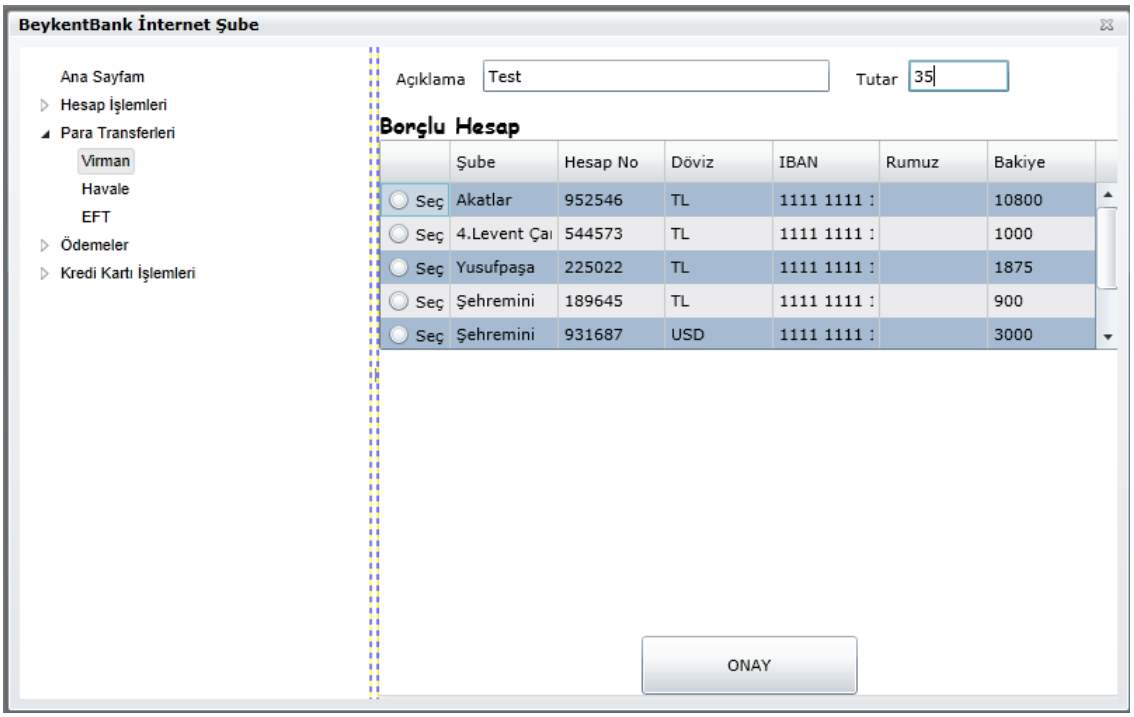


Şekil. 6.9. Vadeli Hesap Açma Ekranı

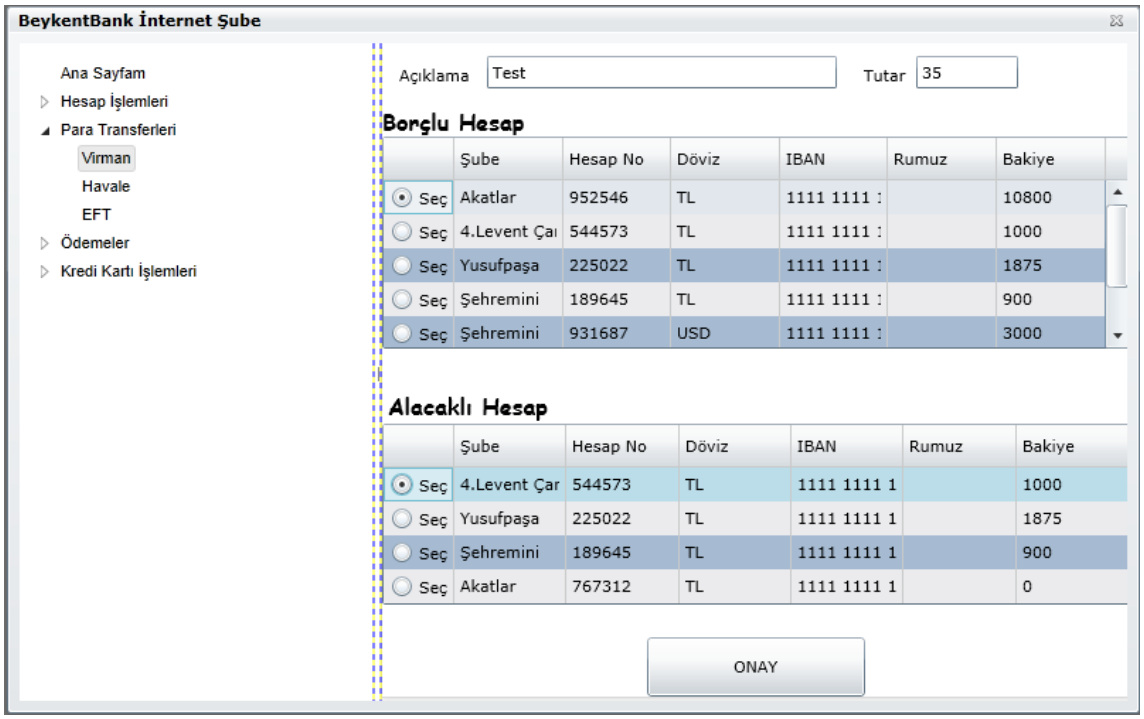
6.6. Para Transferi/Virman

Burada bankaların İnternet Şubelerinde yapmadıkları/yapamadıkları bir özellik kullanımıştır. Silverlight kullanıcılara bunu çok basit birkaç işlemle sağlamaktadır. Ekranın üst tarafında para çıkışının olacağı müşteri hesap seçimi yaptıktan sonra altında müşterinin parasını aktarabileceği seçilen borçlu hesap haricindeki diğer hesapları listelenmektedir. Borçlu hesabın seçilmesi işlemi tetikler ve bilgiler WS'e gönderilip gelen liste alacaklı hesap DataGrid'inde gösterilir. Avantaj olarak sayfanın tamamı sunucuya gönderilmez sadece ilgili bölüm WS'i çağırır. Bunu diğer banka uygulamaları başka sayfaya yönlendirerek yapmaktadırlar. Bu durumda sayfa tamamıyle sunucuya gönderilmekte ve sayfayı yorarak hem zaman kaybına neden olmaktadır hemde kaynakları tüketmektedir. Bu yapıyı diğer script dilleri ve Flash uygulaması destekleyemez. AJAX teknolojisi ile yapılmak istenseydi ajax yüklü olan bir VS ile yapılsaydı ilk önce sayfaya AJAX Extensions sekmesinden ScriptManager nesnesi eklenmeli ikinci olarak UpdatePanel nesnesi eklenmeli üçüncü olarak grid nesnesi eklenmeli ve daha sonra birbirleri ile bağlantıları yapılarak kod yazılmalı. Bu

durumda yine zaman kaybına neden olmaktadır. Silverlight'ta az satır kod ve sadece sayfaya grid eklemeyle yapılmaktadır.



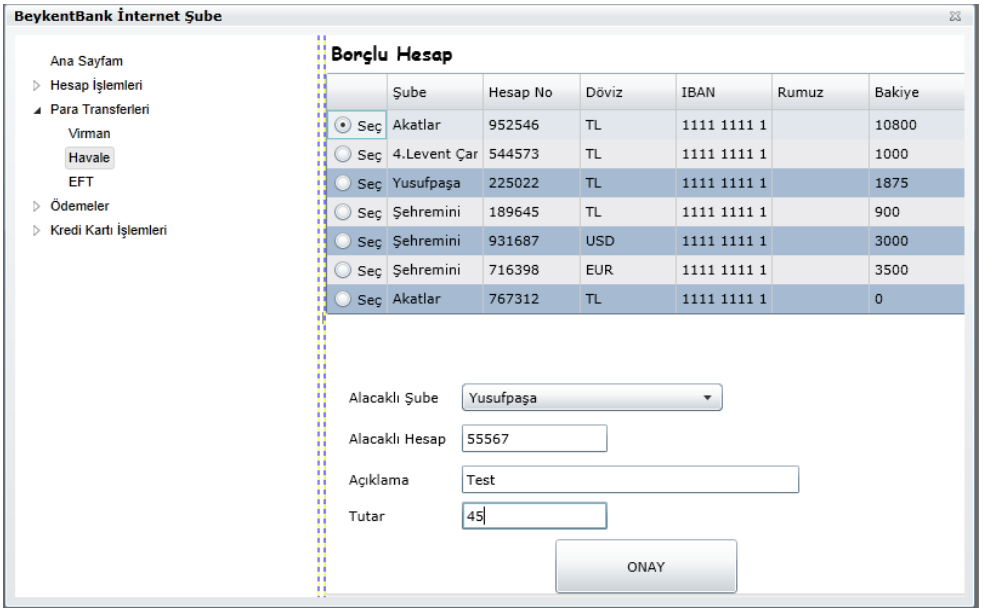
Şekil. 6.10. Virman Borçlu Hesap Ekranı



Şekil. 6.11. Virman Borçlu-Alacaklı Hesap Ekranı

6.7. Para Transferi/Havale

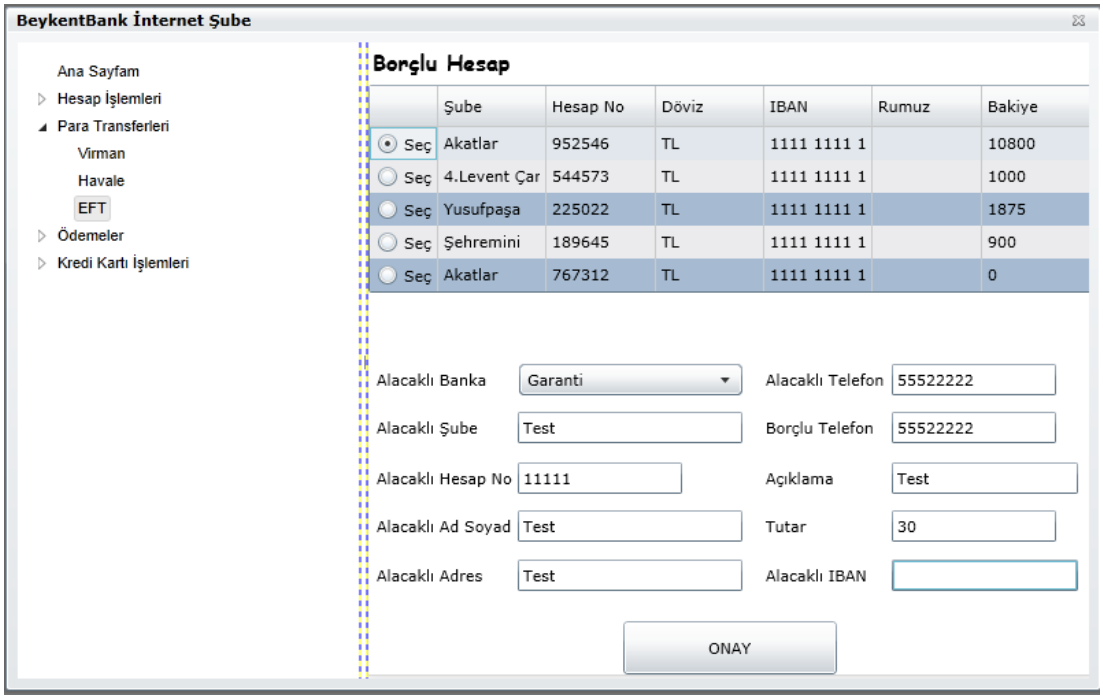
Virman işlemindeki gibi borçlu hesap seçilir. Alacaklı hesap bilgileri girilerek para transfer işlemi gerçekleştirilir.



Şekil. 6.12. Havale Ekranı

6.8. Para Transferi/EFT

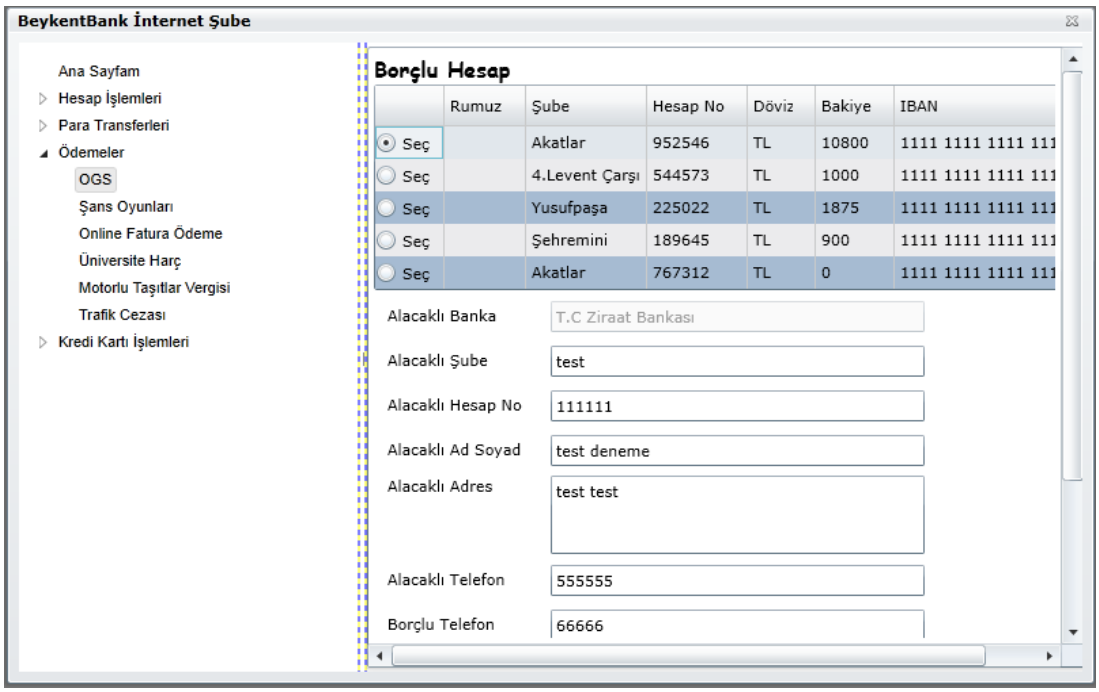
Başka bankalara para transfer işlemi için kullanılır. Borçlu hesap seçilir. Alacaklı banka hesap bilgileri girilerek işlem gerçekleştirilir.



Şekil. 6.13. EFT Ekranı

6.9. Ödemeler/OGS

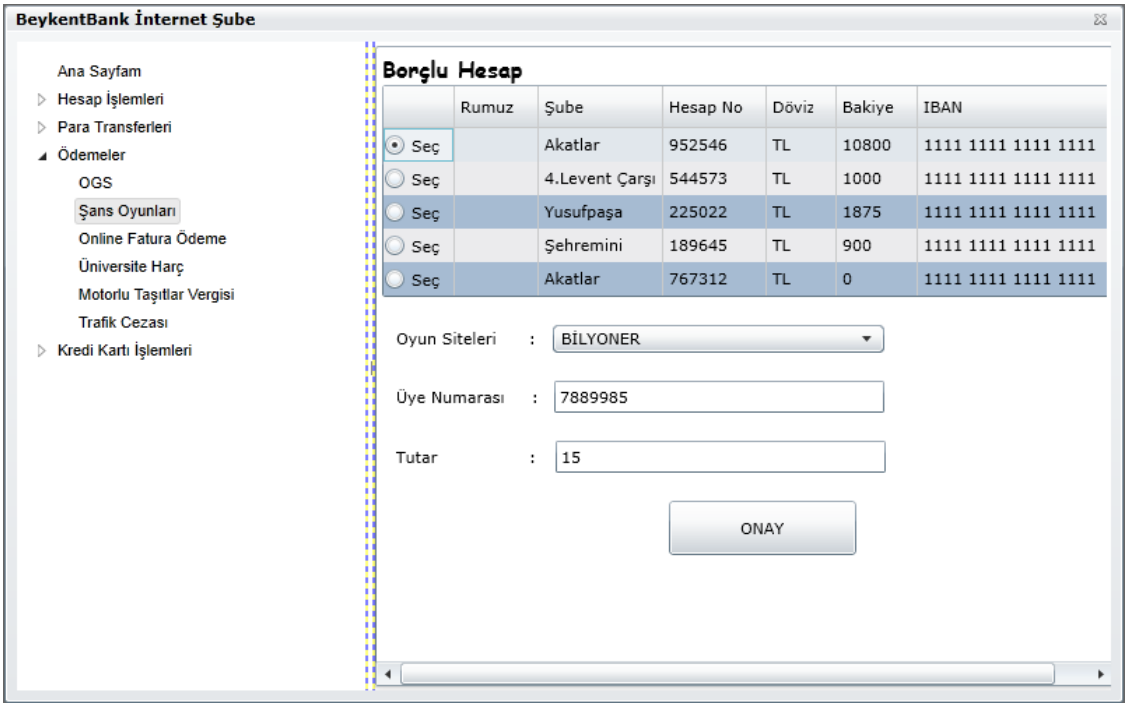
Borçlu hesap seçilerek OGS için para transferi yapılır.



Şekil. 6.14. OGS Ekranı

6.10. Ödemeler/Şans Oyunları

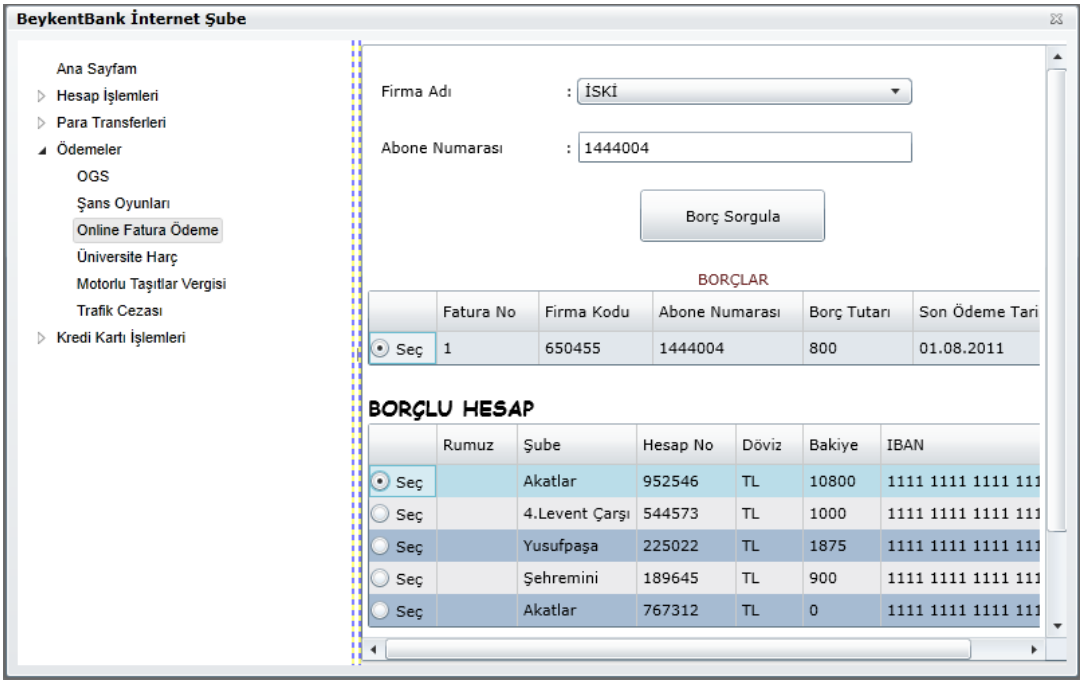
Banka ile anlaşması bulunan şans oyunu firmalarındaki üyelere para transferi yapılır.



Şekil. 6.15. Şans Oyunları Ekranı

6.11. Ödemeler/Online Fatura Ödeme

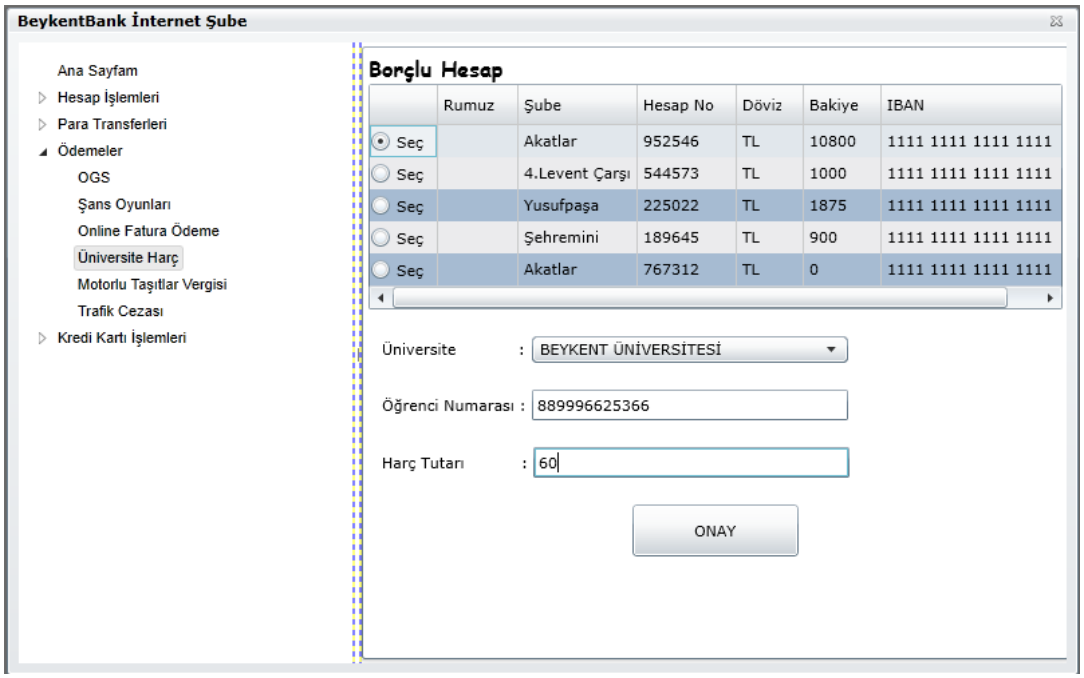
Ekranın üst tarafında firma seçimi yapılır. Alt tarafta abone numarasının girileceği alan bulunmaktadır. Bilgiler girildikten sonra "Sorgula" butonu ile fatura borç bilgisi sorgulanır buradada sayfanın tamamı sunucuya gönderilmeden ilgili alan bilgileri gönderilir ve alınan liste borçlar tablosuna eklenir. Eğer borç varsa aşağıda borç bilgileri ve ödeme yapılacak müşteri hesap bilgileri listelenir.



Şekil. 6.16. Fatura Ödeme-Borç Sorgulama ve Ödeme Ekranı

6.12. Ödemeler/Üniversite Harç Ödeme

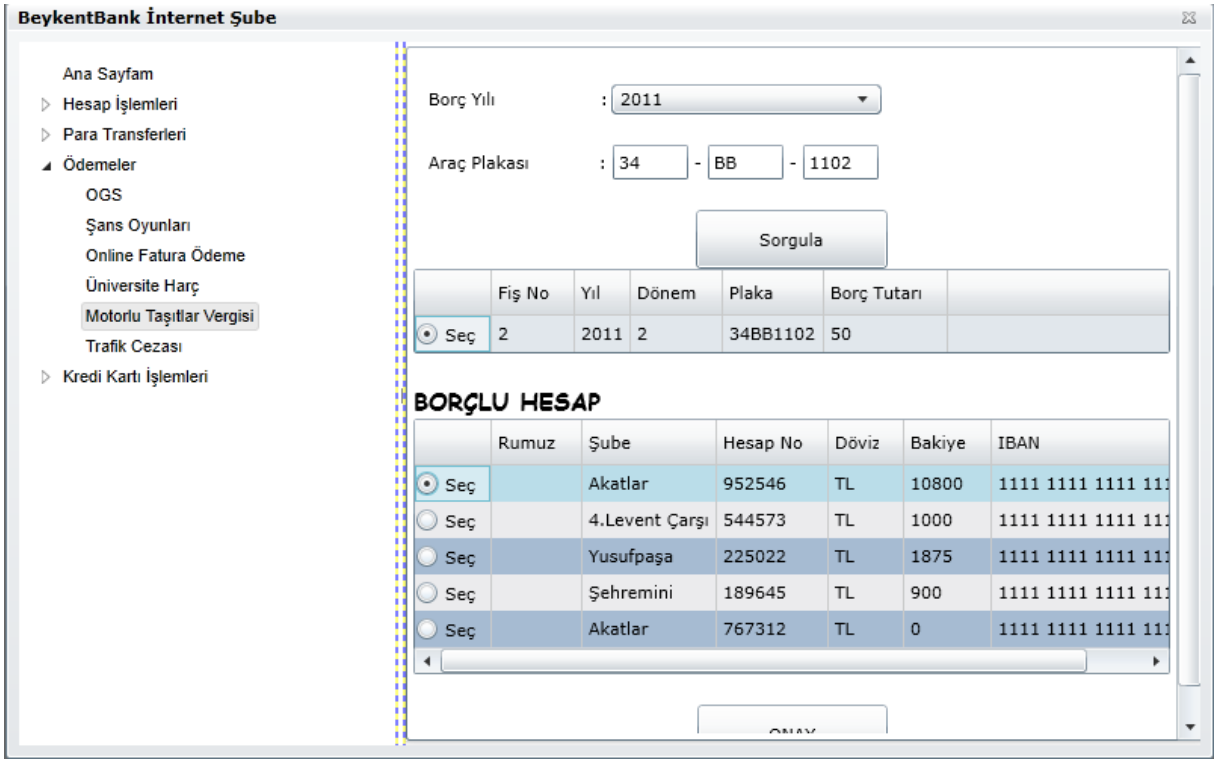
Banka ile anlaşması bulunan Üniversitelere para transferi yapılır.



Şekil. 6.17. Üniversite Harç Ödeme Ekranı

6.13. Ödemeler/Motorlu Taşıtlar Vergisi Ödeme

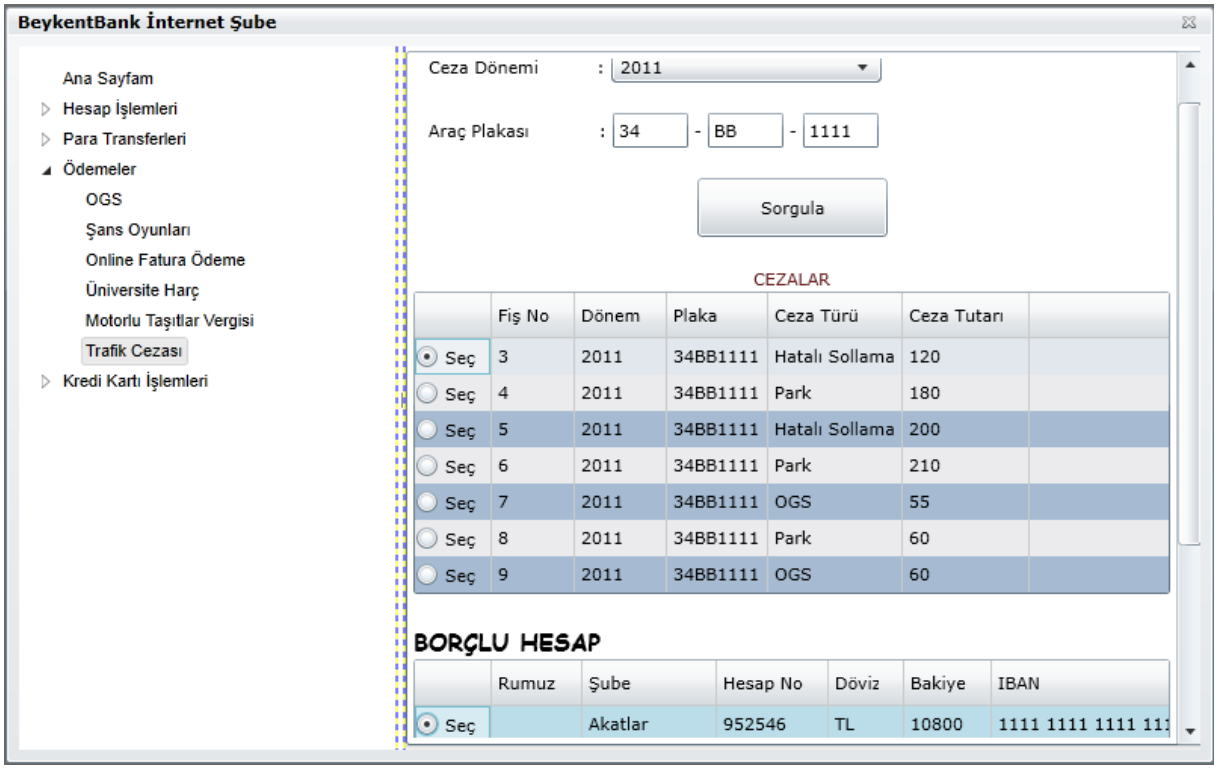
Borç yılı ve araç plakası girilerek borç sorgulaması yapılır. Eğer ödenmesi gereken borç varsa borçlu hesap seçilerek ödeme yapılır.



Şekil. 6.18. Motorlu Taşıtlar Vergisi Ödeme Ekranı

6.14. Ödemeler/Trafik Cezası Ödeme

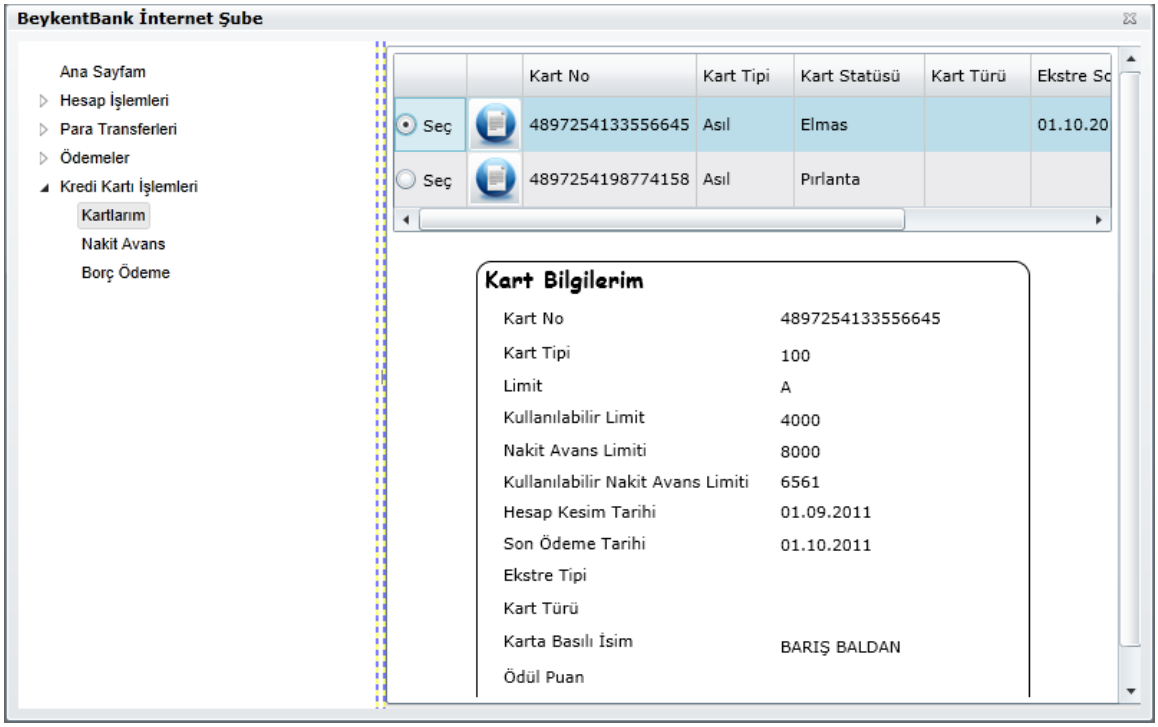
Ceza dönemi ve araç plakası girilerek ceza sorgulaması yapılır. Ödenmesi gereken ceza varsa borçlu hesap seçilerek ödeme yapılır.



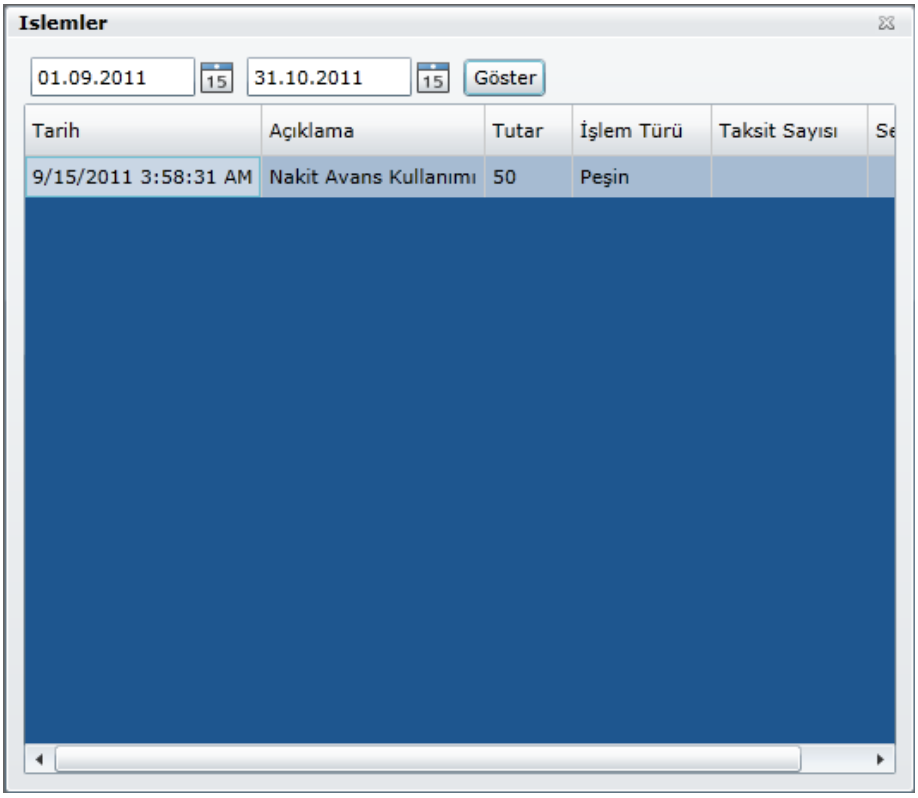
Şekil. 6.19. Trafik Cezası Ödeme Ekranı

6.15. Kredi Kartı İşlemleri/Kartlarım

Müşterinin bankada sahip olduğu kart bilgileri üst taraftaki datagrid'de listelenir. Seç radiobuton'u seçilerek kart bilgileri aşağıda listelenir. Kartların listelendiği tabloda kart numarasının yanındaki buton tıklanarak kart ile yapılan işlemlerin görüntüleneceği ekran açılabilir.



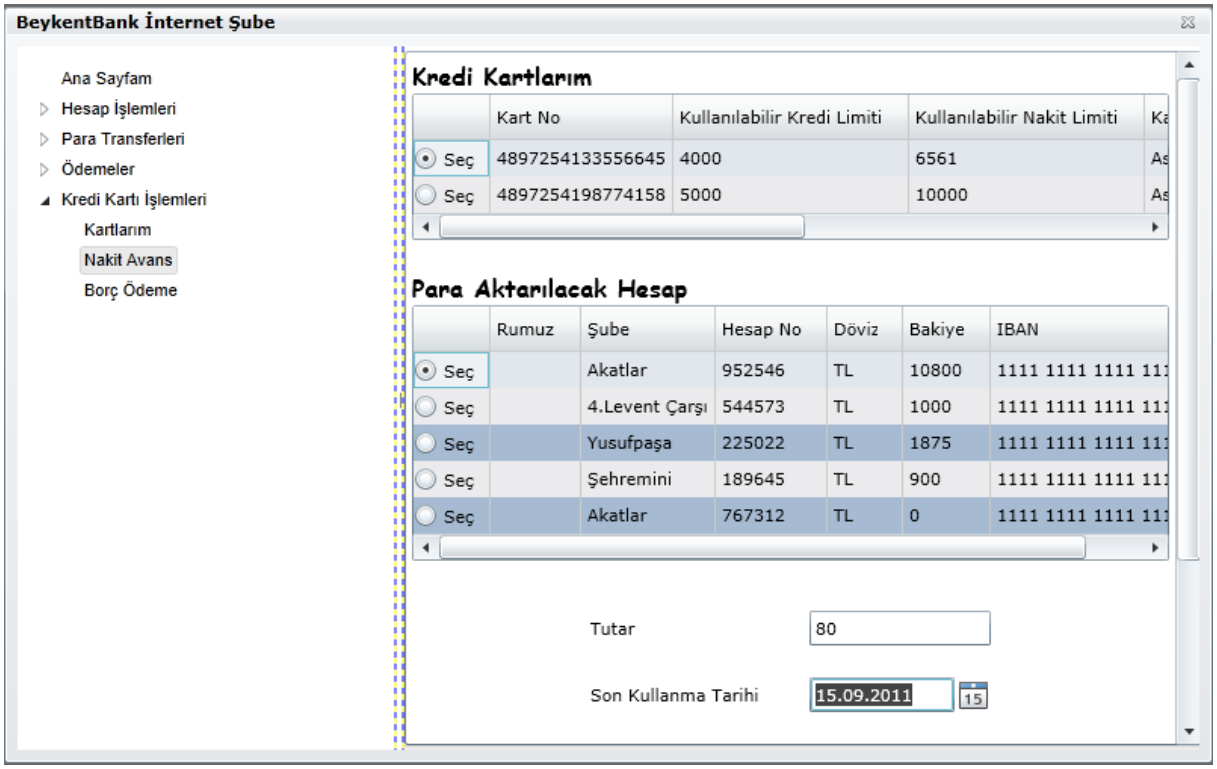
Şekil. 6.20. Kartlarım Ekranı



Şekil. 6.21. Kart İşlemleri Ekranı

6.16. Kredi Kartı İşlemleri/Nakit Avans

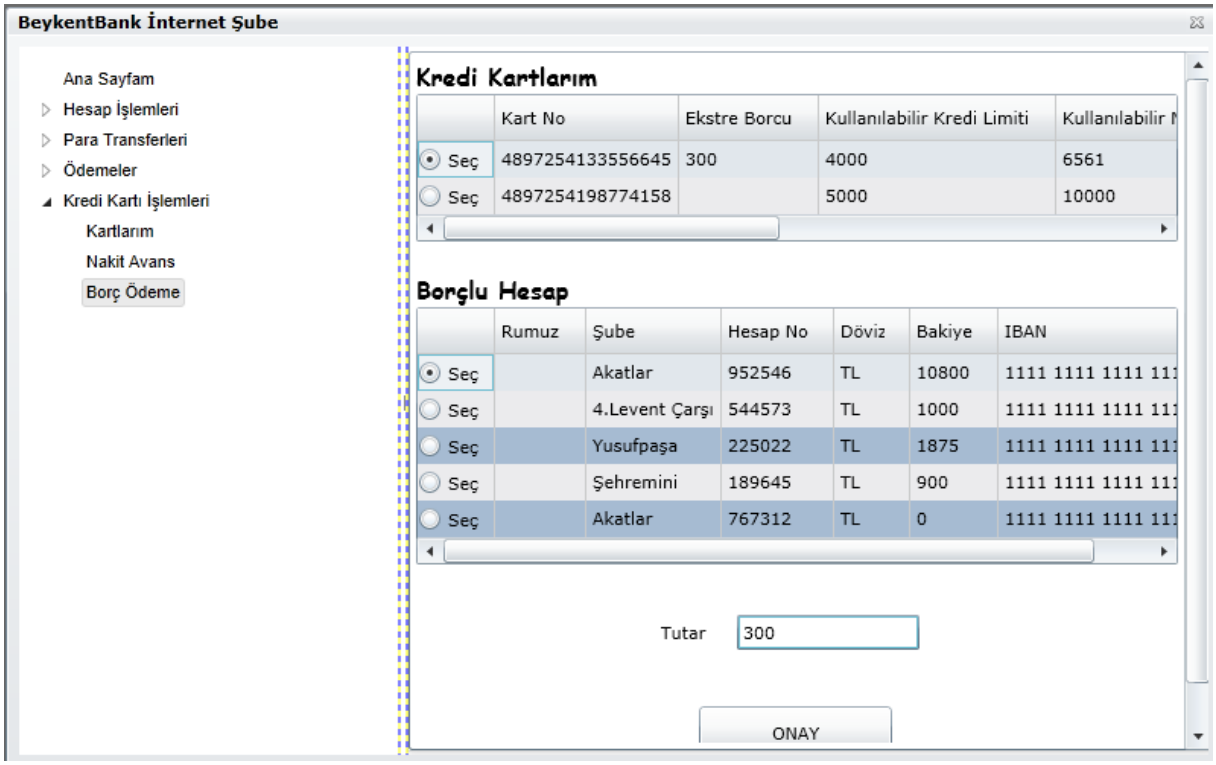
Kredi kartından nakit avans çekme işlemi için kullanılır. Listelenen kredi kartlarım'dan kart seçilir eğer uygun nakit avans limit varsa para aktarılacak hesaplar listelenir ve seçilir. Daha sonra tutari son kullanma tarihi, kart güvenlik numarası ve açıklama girilerek onay butonuna tıklanarak işlem tamamlanır.



Şekil. 6.22. Nakit Avans Ekranı

6.17. Kredi Kartı İşlemleri/Borç Ödeme

Kredi kartına borç ödeme işlemi için kullanılır. Listelenen kredi kartlarım'dan kart seçilir ve borç ödemecek hesaplar listelenir ve seçilir. Daha sonra tutar girilerek onay butonuna tıklanarak işlem tamamlanır.



Şekil. 6.23. Borç ÖdemeEkranı

7. SONUÇ VE ÖNERİLER

Bu tez çalışmasında Silverlight yazılım teknolojisinin günümüzde yaygınlaşan internet bankacılığı uygulamalarına sağladığı avatajlar tez kapsamında geliştirilen bir elektronik banka yazılımı üzerinden açıklanmaya çalışılmıştır. Bu avantajlar; sayfadaki görsellik, müşterilerin kullanım kolaylığı, tasarım ve kodlama maliyetlerinin azaltılması ve zaman tasarrufu olarak özetlenebilir.

Uygulamada Silverlight teknolojisinin en belirgin avantajı Visual Studio(VS) ortamı ile entegre oluşu ve programlama dili (C#, VB vb) ile yazılabilir olmasıdır. Programcılar için, tasarım ve kodlama'da tek program kullanılmasının sağlayacağı avantaj yazılım geliştirmede oldukça önemlidir. Alternatif teknoloji olan Adobe Flash'da, animasyonlar ve kodlamalar için farklı ortamların kullanılması, Silverlight teknolojisinin bir başka önemli avantajıdır. Ayrıca Visual Studio, proje oluşturulmaya başlandığında uygulamayı sayfaya entegre etmek gibi bazı önemli ayarları kullanıcı yerine doğrudan program araçları yardımıyla gerçekleştirir. Bu ise bir dezavantaj olarak geliştirici'nin bilgisayarında Visual Studio IDE ortamı'nın yüklü olmasını gerektirir. Flash sayfalarını tasarlamak için ise aynı şekilde Adobe firmasının Flash programının makinada yüklü olması gerekmektedir.

Web 2.0 akımı ve zengin internet uygulamaları(RIA) günümüz popüler konuları-arasındadır.. Bu nedenle tez çalışması bu amaca yönelik olarak son teknoloji olan Silverlight teknolojisi kullanılarak geliştirilmiştir. Silverlight teknolojisi kullanılarak kullanıcının istekleriyle sayfanın tamamen yenilenmeden (zaman kaybı olmaksızın) ilgili işlemin yürütülebilmesi sağlanmaktadır. Diğer script dillerinde(JS, Jquery vs..) bazı sayfa yapılarını hazırlamak için çok fazla satır kod yazılmalı ve nesne tanımlamaları yapılmalıdır. Adobe firmasının ürünü olan Flash ile yapılmak istendiğinde, ekran için frame'ler hazırlanmalı, actionscript dilinde kod yazılmalı ve bunlar web sayfası ile entegre edilmelidir. Bu seçim çok zahmetli ve çok maliyetli olmaktadır. Bazı sayfa yapılarını script dilleri ve Flash uygulaması desteklemeyebiliyor. AJAX teknolojisi kullanılmak istendiğinde ise AJAX yüklü olan bir VS ile yapılmalı ve önce sayfaya AJAX Extensions sekmesinden ScriptManager nesnesi eklenmelidir. İkinci olarak UpdatePanel nesnesi, üçüncü olarak data nesnesi eklenerek ve sonrasında birbirleri ile bağlantıları yapılarak kod yazılmalıdır. Bu durum da yine zaman

kaybına neden olmaktadır. Silverlight'ta az satır kod ve sadece sayfaya data nesnesi ekleyerek bu işlemler yapılabilmektedir. Geliştirilen uygulama çapraz tarayıcı(internet explorer haricinde diğer browser'lar) desteğine sahip, esnek ve kolay kullanılabilir bir yapıya sahiptir. Ayrıca sayfayı tasarlayacak olan tasarımcı/programcı açısından da görselliği zengin ve fazla süre almayan (adam x gün) uygulamalar geliştirilebilir.

Tezde, halen kullanılan bankaların internet şube uygulamalarındaki eksikliklere, görsel zengileştirme gereksinimlerine de ayrıca değinilmiştir.

KAYNAKLAR

[1] Journal of Internet Banking and Commerce, December 2008, vol. 13, no.3 : Internet Banking Websites Performance in Greece

[2] Journal of Internet Banking and Commerce, August 2010, vol. 15, no.2: An Evaluation of Internet Banking in Turkey

[3] Silverlight Team, http://team.silverlight.net/

[4] Silverlight, http://www.silverlight.net/

[5] Erman G. "Silverlight 3", 2003

[6] Wikipedia, http://tr.wikipedia.org/wiki/Microsoft_Silverlight

[7] Microsoft, http://www.microsoft.com/silverlight/

[8] ProgrammerWorld.NET**,** http://faq.programmerworld.net/programming/silverlight.html

[9] Daron Yöndem Blog**,** http://daron.yondem.com/tr/

[10] http://www.ilkayilknur.com/post/2010/07/01/WCF-RIA-Servicesa-Giris.aspx

[11] Microsoft, http://www.microsoft.com/sqlserver/2008/tr/tr/overview.aspx

[12] Journal of Internet Banking and Commerce, December 2007, vol. 12, no.3 : An Analysis of Web Navigability in Spanish Internet Banking

[13] Wikipedia, http://tr.wikipedia.org/wiki/C_Sharp_(programlama_dili)

[14] http://tr.wikipe-dia.org/wiki/Microsoft_SQL_Server

[15] http://www.godoro.com/Divisions/Ehil/Mecmua/Magazines/Articles/txt/html/article_StoredProcedures.html

[16] Daron Yöndem, http://social.msdn.microsoft.com/Forums/trTR/vstr/thread/5abecde8-4e8a-428f-9a4a-6f5e8350897a/

[17] Daron Yöndem, http://www.sanalkurs.net/linq-to-sql-kutuphanesi-ile-insert-update-ve-delete-islemleri-4331.html

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, Adı : BALDAN, Barış

Uyruğu : T.C.

Doğum tarihi ve yeri : 26.07.1980 İstanbul

Medeni hali : Bekar

Telefon : 0 (262) 686 00 00

e-posta : baldan_bb@hotmail.com

Eğitim

Derece	Eğitim Birimi	Mezuniyet tarihi
Lisans	Trakya Üniversitesi/ İ.İ.B Fak.. İşletme Bölümü	2005
Lise	Ahmet Rasim Lisesi	1998

İş Deneyimi

Yıl	Yer	Görev
2007-2011	Akbank T.A.Ş	Yönetim Destek YY

Yabancı Dil

İngilizce