İNTERNET ÜZERİNDE DAĞITIK VE KATMANLI MİMARİ İLE UYGULAMA GELİŞTİRME

498691 ERKAN DURMAZ
498694 OLCAY ÜNVER

BİTİRME ÖDEVİ
MAYIS 2003
Yönetici: Yrd. Doç. Dr. Feza Buzluca

İÇİNDEKİLER

ÖNSÖZ

Katmanlı yapı, kurumsal uygulamlarda her geçen gün önemini giderek artırmakta, ve bu yapıyı sağlamak için çok çeşitli yazılım araçları sunulmaktadır. Bu araçlar kullanılarak, çok sağlam ve güvenilir yapıda uygulamalar geliştirilmekte ve şirketlerin güvenilirliği artmaktadır. Artık modern bir yazılım mimarisi olan katmanlı yapı, tüm yazılım şirketleri tarafından kullanılmaya başlanmıştır. Ayrıca dağıtık sistemlerde, şirketlerin kaçınılmaz gereksinmilerinden birisi haline gelmiştir. Bu çalışma bünyesinde; dağıtık ve katmanlı yapıda bir uygulama geliştirilmiştir.

Bilgisayar Mühendisliği lisans eğitimim süresince değerli bilgilerinden yararlandığımız, ilgi ve desteklerini gördüğümüz tüm hocalarımıza, bitirme çalışmamızın yürütülmesinde emeği geçen, her konuda destek gördüğümüz değerli hocamız Sayın Yrd. Doç. Dr. Feza Buzluca`ya teşekkür ederiz.

İstanbul 2003

Olcay ÜNVER
Erkan DURMAZ

1. GİRİŞ

Çağımızın en önemli gelişmelerinden biri olan İnternetin, her geçen gün yaygınlığı ve kulllanılabilirliği artmakta, günlük hayatımızın değişmez unsurlarından biri haline gelmektedir. İnternetin yaygınlaşmaya başlaması ile birlikte İnternet üzerinde çalışan uygulamalarının yaygınlığıda aynı paralellikte artmaktadır.

Günümüzdeki İnternet uygulamalarını iki alt başlıkta toplayabiliriz: İnternet gözatıcısı (browser) üzerinde koşan internet uygulamaları (örn: Yahoo, Amazon) ve TCP/IP protolkolü ile çalışan, işletim sistemine bağımlı internet uygulamaları(örn: ICQ, GetRight). Bu iki grup internet uygulamalarının artı ve eksi yönlerini irdeleyecek olursak; internet gözatıcısı üzerinde çalışan uygulamaların en önemli avantajı hemen her platformda çalışmalarıdır. Bu platform bağımsızlık HTTP(Hyper Text Transfer Protocol) ile sağlanmaktadır. Fakat, internet gözatıcılarının kısıtlı yetenekleri ve güvenlik sorunları büyük kurumsal uygulamaların geliştirilebilmesine engel teşkil etmektedir. Diğer grupta toplanan internet uygulamaları bu açıdan bakıldığında internet gözatıcısı üzerinde çalışan uygulamalardan daha avantajlıdır. Çünkü bu uygulamalar işletim sistemi desteği ile sistem kaynaklarını kullanabilmektedir. Fakat, bu grup internet uygulamaların en önemli dezavantajı üzerinde çalıştığı platforma bağımlı olmasıdır. Karşılaşılan bu sıkıntıların çözümlemesi için W3C, ISO gibi standardizasyon kurumlarının yanısıra, yazılım dünyasının Microsoft, Sun Microsystems gibi önde gelen kurumları çeşitli çözümler önermişlerdir. Bu çözüm önerilerinin ortak hedefi internet uygulamalarının platform bağımsız hale getirmek ve istemcinin sistem kaynaklarının en etkin şekilde kullanabilmesini sağlamaktır. Bir başka ifade ile Linux işletim sistemi altında çalışan bir uygulamanın, Windows işletim sistemi üzerinde çalışabilmeli ve hatta PDA gibi akıllı cihazlarda da (smart device) çalışabilmelidir. Bu hedeflerin gerçekleşmesi için atılan en önemli adımlardan birisi Web Servisleridir. Web Servisleri kavramı üzerinde ilerideki bölümlerde ayrıntılı olarak durulacaktır.

Bizde projemizde Web Servislerinin sağlamış olduğu imkanlardan istifade ederek, hemen her platformda çalışabilecek, istemci tarafındaki bilgisayarın kaynaklarını en efektif şekilde kullanabilecek, geliştirilebilir(esnek) ve ölçeklenebilir bir internet uygulaması modeli

geliştirilmiştir. Geliştirilen bu model ışığında, farklı platformlardaki oyuncuların birbirleriyle farklı oyunları karşılıklı oynayabilmesini sağlayan bir internet uygulaması gerçeklenmiştir.

1.1 PROJENİN AMACI VE KAPSAMI

Projenin amacı, Visual Studio .NET ortamında, .NET teknolojisinin sağlamış olduğu yenilikleri kullanarak, her tür platformda hizmet verebilen kurumsal ve katmanlı bir yapıda uygulama geliştirmektir. .NET teknolojileri, kurumsal uygulamalarda çok önemli rol oynamakla birlikte, gelecekte yazılım sektöründe çok önemli pazar payına sahip olması beklenmektedir.

Projenin kapsamı bu nedenle çok geniştir ve bir çok yeni teknolojinin kullanılmasını gerektirmiştir. Aşağıda, kullanılan teknolojilerin listesi bulunmaktadır.

- .NET Framework
- .NET Compact Framework
- XML Web Servisleri
- ADO.NET
- OLE.DB(Veri tabanı bağlantısı için)
- ASP.NET
- ASP.NET Mobile Controls
- Smart Device Extensions
- .NET Reflection
- Cryptography Services.

Bitirme projesinin basit olarak, N farklı platformdan, N farklı kullanıcıya , N farklı oyunu karşılıklı olarak oynama imkanı verebilen bir platform sağlamaktır. Şu anda hedeflemiş olduğu platformlar PC'ler, PDA'ler, Internet'e bağlanabilir telefonlar ve Web arayüzleridir. Oyun platformunda(GameNETwork), farklı platformlardan karşılılı olarak oynanan oyunların tek bir ortak katmanda tutulması için Web Servisleri kullanılmıştır. Web servisleri ile ilgili kapsamlı bilgi ileriki konularda ele alınacaktır.

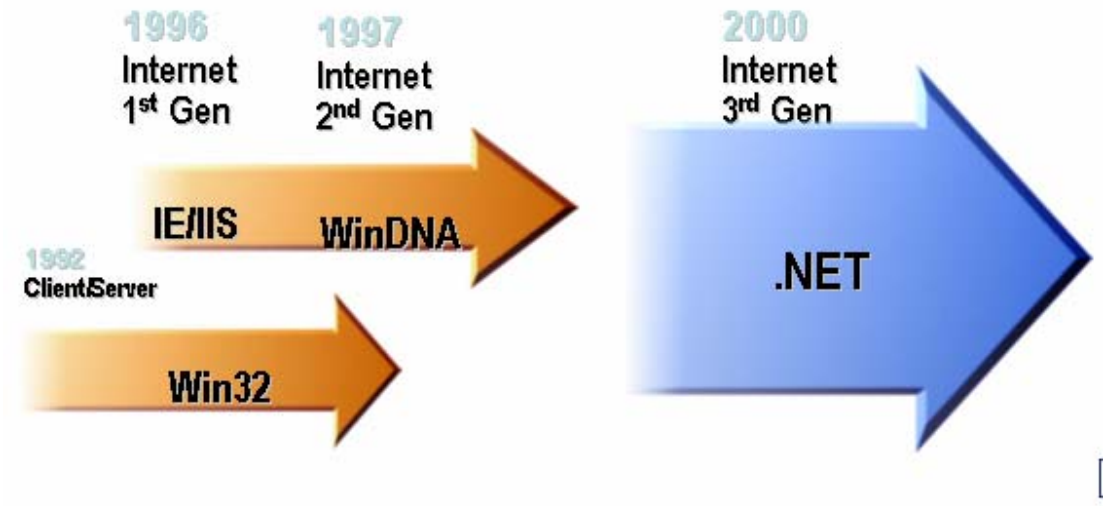
1.2 PROJENİN SOSYAL VE TİCARİ AÇIDAN ÖNEMİ

GameNETwork projesinin sosyal açıdan önemi, farklı kesimlerden insanları biraraya getirerek yeni arkadaşlıkların oluşmasına ve sağladığı mantık oyunları ile insanların zekalarının gelişmesine yardımcı olmasıdır.

Ayrıca GameNETwork projesinin ticari açıdan önemi ise, platformun üyelik sistemi ile çalışmasıdır. Üyelerin belirli bir ücret karşılığı oyunları oynayabilmesi nedeni ile ticari açıdanda önemlidir.

2 .NET YAZILIM GELİŞTİRME PLATFORMU

Teknolojinin her geçen gün hızla gelişmesi ile yeni gereksinimlerde ortaya çıkmaktadır. Bugün, internetin her alanda hizmet verebilmesi nedeni ile kullanıcılara bu hizmeti sağlayabilmek için çeşitli yazılım geliştirme araçları ortaya çıkmıştır. .NET platformu bu araçların en önemlilerinden birisidir. .NET'e kadar olan tarihsel gelişim şu şekildedir:



2.1: Yazılımın Tarihsel Gelişimi

.NET platformunun ortaya çıkışının sebebi, bilgisayar dünyasının PC'lerin serverlara bağlanarak hizmet almalarından, akıllı cihazların(PDA, Smart Phone), bilgisayarların ve servislerin birarada çalışmasını sağlayacak yönde değişmesi şeklindeki gelişmesindendir.
.NET platformu Microsoft'un, bu değişim karşısında, yazılım geliştiricilere sunduğu bir cevap niteliğindedir. .NET Framework, .NET platformu denince akla gelen ilk konudur.
.NET Platformu, Visual Studio .NET(VS.NET), .NET Common Language Runtime(CLR), ve .NET Base Class Libraries(BCL)'den oluşmaktadır.

Genel mimarisine bakıldığında, .NET genel olarak üç ana kısımdamn oluşmaktadır:
.NET Framework: Tüm özelliklere sahip yeni bir geliştirme platformu.
Birçok .NET Ürünü: XML(Extensible Markup Language)i destekleyen, Exchange ve SQL Serverıda içeren ve .NET platformuna entegre edilmiş, .NET Framework ile ilgili birçok Microsoft uygulaması,

Çeşitli .NET servisleri: Microsoft tarafından .NET platformunda kullanılmak üzere sunulmuş çeşitli servisler. Bu servisler Microsoft'un HailStorm projesi kapsamında geliştirilmektedirler.

.NET Framework de kendi içinde üç bölümde incelenebilir.
Ortak Dil Çalıştırma Platformu(Common Language Runtime - CLR) Kontrol edilebilen ve bellek yönetimi, hata kontrolü, bellek tahsisatı, işletim sistemi servisleri ile etkileşimde bulunan bir çalıştırma ortamıdır. Uygulama geliştirmeyi kolaylaştırır, güçlü ve güvenli çalışma ortamı sağlar, birden çok dil desteği verir.

Taban Sınıf Kütüphanesi (Base Class Library -BCL): .NET Framework sınıf kütüphanesi, birtakım çalıştırma zamanı özellikleri ve her uygulama geliştiricinin ihtiyaç duyacağı diğer yüksek seviyeli servisleri sunar. Sınıflar, .NET uygulamaları geliştirmeyi kolaylaştırır. Uygulama geliştiriciler, sınıfları kendi kütüphanelerini oluşturarak genişletebilirler.

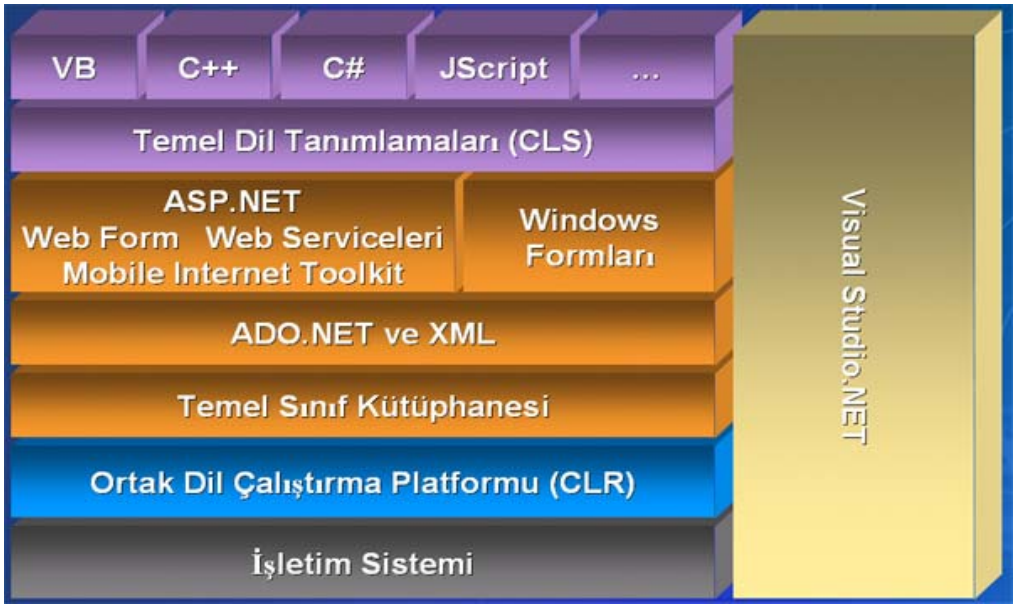
En önemli iki uygulama hedefi Web Uygulamaları(ASP.NET) ve Windows Uygulamaları(Windows Forms)

.NET Frameworkun sağlamış olduğu avantajlar: daha kısa yazılım geliştirme döngüleri(code reuse, daha az sürpriz, birden fazla programlama dilini destekleme), daha kolay yükleme, veri tipine bağlı hataların içsel tip güvenliği(integral type safety) ile azaltılması, garbage collector sayesinde daha az bellek sızıntısı, ve genel olarak daha genişleyebilir(scalable), daha güvenilir(reliable) uygulamalar.

2.1 MICROSOFT .NET MİMARİSİ

Aşağıdaki şekil .NET platformunun mimarisini göstermektedir. Temel olarak, .NET diller ailesinin her biri, Common Language Specification(CLS) e uygun olarak.Microsoft Intermediate Language (MSIL, yada sadece IL - ara dil)e derlenmektedir. Genel olarak uygulama geliştirme tipleri, Web Forms, Web Services, ve Windows Forms uygulamalarıdır. Bu uygulamalar, XML ve Simple Object Access Protocol (SOAP)'unu kullanarak Base Class Library'lerden fonksiyonalitelerini alarak, CLR(common Language Runtime)da çalışarak, birbirleriyle haberleşirler. Visual Studio.NET, .NET Framework uygulamaları geliştirebilmek

için gerekli değildir, fakat Visual Studio .NET in sunmuş olduğu kapsamlı mimari, Visual Studio .NET'in ideal bir seçenek olmasına sebeptir.



2.1.1: .NET Mimarisi

2.2 MICROSOFT .NET PLATFORMUNUN ÖZELLİKLERİ

.NET Platformunun çekirdeği, Ortak Dil Çalıştırma Platformu(Common Language Runtime-CLR) , Taban Sınıf Kütüphanesi (Base Class Library-BCL), ve Ortak Dil Tanımlama (Common Language Specification - CLS) dan oluşmaktadır.

.NET Taban Sınıf Kütüphanesi, Windows API'lerinin Windows işletim sistemini etkin hale getirmesinde kullanıldığı gibi, Ortak Dil Çaçalıştırma Platformuna özellikler sağlar ayrıca, Taban Sınıf Kütüphanesi kod tekrar kullanımı(code reuse)nı sağlamak için bir çok özellikte sunar. Aşağıda .NET platformunun bazı özelliklerine değinilmiştir.

2.2.1 ÇOK DİLLE(MULTILANGUAGE) UYGULAMA GELİŞTİRME

Birçok dilin .NET Ortak Dil Çalıştırma Platformunu desteklemesi ile, kendinize en uygun dili kullanarak yazılım geliştirebilmeniz artık daha kolay. Farklı dillerin birarada çalışmasını

sağlayan, COM ve CORBA gibi eski yöntemler Arayüz Tanımlama Dili (Interface Definition Language-IDL) kullanarak bunu gerçeklemişlerdir..NET platformu dillerin entegrasyonunu MSIL ile gerçeklemektedir. MSIL, assemblera benzer bir sentaksa sahip olmasına rağmen içierisinde nesnelerin yönetimi, hata durumlarını oluşturulması ve hata yakalanması ile ilgili çeşitli fonksiyonlarda mevcuttur.

Microsoft Ortak Dil Tanımlama (CLS), diğer derleyici üreticilerininde, diğer .NET dilleri ile entegrasyonunu sağlayabilmek için IL üretebilmesi için gerekli olan bilgileri içerir. .NET Ortak Dil Çalıştırma Platformu: C++, C#, J#, Jscript, ve Visual Basic dilleri için IL üretebilmektedir. Ayrıca birçok firmada .NET Ortak Dil Çalıştırma Platformunu hedefleyen derleyiciler üretmektedir. Şu anda, COBOL, Eiffel, Fortran, Perl, Python, Scheme, dilleri desteklemektedir.

2.2.2 PLATFORM VE İŞLEMCİ BAĞIMSIZLIĞI

Ara kod(IL), işlemci bağımsızdır ve birçok makine dilinden yüksek seviyelidir. .NET uygulaması bir kere yazılıp derlendiği zaman .NET Ortak Dil Çalıştırma Platformunu destekleyen herhangi bir platformda çalışabilir..NET Ortak Tip Sisteminin .NET uygulamaları için taban veri tiplerinin büyüklüklerini tanımlaması nedeniyle, uygulama geliştirici .NET'i destekleyen platformlarda donanım ve yazılım ayrıntılarından uzaklaştırılmıştır.

.NET uygulamaları şu an için sadece Windows platformlarını desteklemesine rağmen, Unix ve Linux içinde .NET Framework geliştirilmektedir.

2.2.3 OTOMATİK BELLEK YÖNETİMİ

Otomatik bellek yönetimi olmayan bir ortamda uygulama geliştiren bir yazılımcı için, bellek sızıntısı, saatlerce süren bir hata ayıklamaya sebep olabilir.

Visual Basic yada COM uygulama geliştiricleri bunu engellemek için referans sayma tekniğini kullanmışlardır. Bu tekniğe göre, bir nesne tarafından kullanılan bellek alanını, o nesneye işaret eden referans kalmadığı zaman bellekten silme yöntemi ile gerçeklenir. Bu teknik

doğruymuş gibi gözüksede, birkaç problemi vardır. En önemli problemi, biribirini referanse eden nesnelerin olması, ve bu nedenle bu nesnelerin sürekli bellekte kalmasıdır.

C yada C++ programcılığında ise oluşturulan nesnelerin, programcı tarafından bellekten silinmesi gerekmektedir. Fakat, .NET ortamında, Microsoft yazılım geliştirmeyi kolay hale getirmek amacı ile, ileride anlatılacak olan, garbage collection mekanizmasını kullanmıştır.

2.3 .NET MİMARİSİNİN BİLEŞENLERİ

.NET Framework yukarıdaki şekilden de görülebileceği gibi birçok bileşenden oluşmaktadır. Bu bileşenler şunlardır:

2.3.1 .NET ORTAK DİL ÇALIŞTIRMA PLATFORMU(CLR)

.NET Framework'ün kalbi Ortak Dil Çalıştırma Platformu(Common Language Runtime-CLR)' dur.

Java'nın Sanal Makina(Virtual Machine)sına benzer yapıda olup, Ara kodu(IL) çalıştıran bir çalıştırma platformudur. Java'dan farklı olarak, Temel Dil Tanımlarına uyan dillere hizmet verebilmektedir. Görevleri şu şekildedir:

- Çalışan kodun yönetimini üstlenir.
  - Tip uyumluluğunu kontrol eder.
  - Otomatik memory yönetimi ve hata kontrolü sağlar.
- Ortak bir tip sistemi sağlar.
  - Value tipler (integer, float, structure, vs.)
  - Nesneler, Interface'ler.
- Sistem kaynaklarına erişim sağlar.
  - Native API, COM interop, vs.

2.3.2 DÜZENLİ/DÜZENSİZ KOD

Ortak Dil Çalıştırma platformunu hedefleyen her kod düzenli koddur. Yani, CLR tarafından çalışması ve davranışı kontrol edilebilen kod düzenli koddur. Düzenli kodda bulunan metadata, düzenli kodun CLR tarafından güvenli bir şekilde çalıştırılmasını sağlar. Güvenli

çalıştırma, bellek ve güvenlik yönetimi, tip güvenliği, ve dillerin ortak kullanımı anlamına gelir. Düzensiz kod ise, kendisine ait olmayan bellek bölgesine yazma gibi çeşitli uygun olmayan davranışlar sergileyen koddur. Bugün, Windowsta çalışan birçok uygulama düzensiz bir yapıya sahiptir.

2.3.3 ARA DİL(IL)

.NET'in ara dili(Intermediate Language-MSIL), Ortak Dil tanımlamasında tanımlanmıştır. Bu makine diline benzer , düşük seviyeli bir dildir. Uygulamalar bu dille doğrudan yazılabilir, fakat bu sıradışı durumlar için gerekmektedir.

2.3.4 ORTAK TİP SİSTEMİ

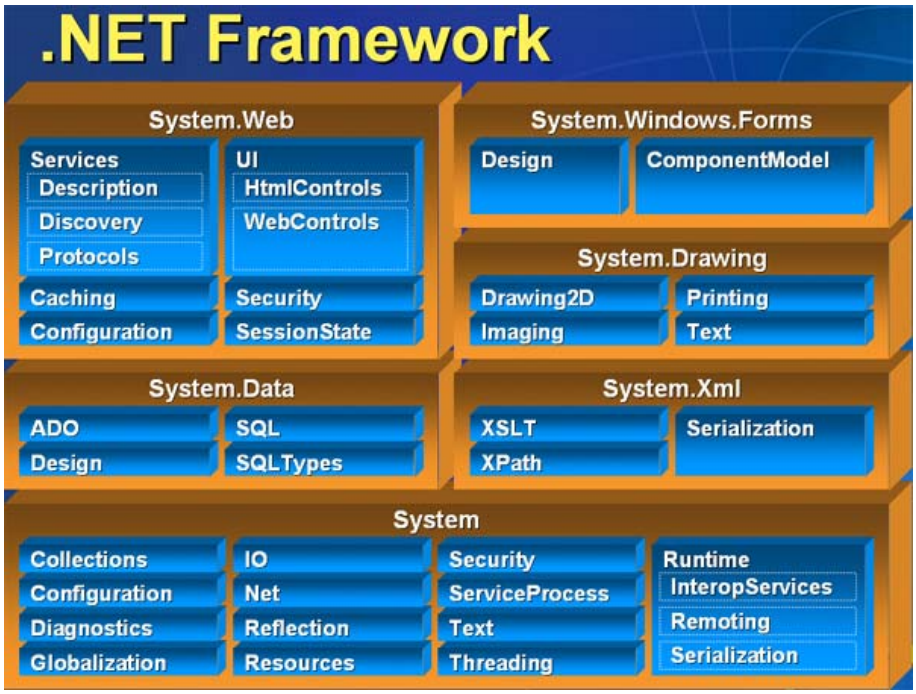
.NET uygulamaları, yazıldıkları dilden bağımsız olarak, ortak bir tip sistemini kullanmaktadırlar. Buda bir dilde kullanılan bir tipin diğerinde nasıl kulanıldığının bilinmesinin gerekliliğini ortadan kaldırmaktadır. Tüm .NET veri tipleri *System.Objec*t tipinden türetilmişlerdir.

Tüm tiplerin tekbir taban sınıftan türetilmeleri, ortak özellikleri paylaşmalarına, örneğin stringe çevrilebilmelerine, serialized hale dönüştürülebilmelerine, yada bir koleksiyon (collection) içinde tutulabilmelerine, imkan sağlamıştır.

2.3.5 .NET TABAN SINIF KÜTÜPHANESİ

.NET dillerinin hepsinin ortak kütüphaneleri paylaşmasından dolayı, C# da kullanılan bir fonksiyon ile başka bir dilde kullanılan fonksiyon aynıdır. Buda .NET'i hedefleyen tüm dillerin, farklı sentaksa sahip olmalarına rağmen, ortak özelliklere sahip olduğunun göstergesidir.

.NET Namespace mimarisi şu şekildedir.



2.3.5.1: .NET Namespace Mimarisi

2.3.6 YANSIMA(REFLECTION)

Yansıma, bir assembly'de(.dll) bulunan nesneler hakkında bilgiyi çalışma zamanında(runtime) elde etmek için kullanılan, ve ayrıca bu nesneleri ve metodlarını çalıştırmak için kullanılan .NET bileşenidir.

2.3.7 JUST-IN-TIME DERLEME

NET Ortak Dil Çalıştırma Platformu(CLR) Just In Time (JIT) derleme teknolojisi ile Ara Dili, cihaza bağımlı koda dönüştürür. Üç çeşit JIT derleyici bulunmaktadır:

Pre-JIT Bu JIT bir assemblynin tamamını bağımlı koda dönüştürür, ve kurulum zamanında kullanılır.

Econo-JIT Bu JIT, kısıtlı kaynağa sahip cihazlarda kullanılır. Ara Kodu bit bit derler, ve cacheleme işlemini gerçekler.

Normal JIT Bu JIT ise kodu sadece çağırma işlemi gerçeklendiği zaman derler, ve derlenmiş kodu caheine yerleştirir.

JIT derleyicisinin asıl amacı, tercüme edilmiş kodu, cachee yerleştirerek, performansı artırmaktır. Böylece, bir sonraki çağırma durumunda, cachede bulunan kod çağrılacağından performans artar.

2.3.8 GARBAGE COLLECTION

Bellek yönetimi, bellek sızıntılarının kontrol edilmesi dıurumunda, çok fazla yazılım geliştirme zamanı harcanmasına sebep olan konulardan birisidir.

.NET bu sıkıntıyı sağlamış olduğu Garbage Collection mekanizması ile önlemektedir. Bu mekanizma, uygulama bellek sıkıntısı çekmeye başladığı zaman çalışmaya başlar. Mekanizma şu şekilde çalışmaktadır:

Uygulama daha fazla bellek isteğinde bulunduğu ve bellek tahsisatçısı daha fazla bellek bulamadığını bildirdiği zaman, garbage collection mekanizması çalışır. Garbage Collector, bellekte her şeyin çöp olduğunu ve bellekten atılabileceğini varsayar. Daha sonra, uygulamada referanse edilmiş tüm bellek alanlarına ait grafı çıkartarak, referanse edilmeyen alanları, çıkartarak belleği sıkıştırır. Bu işlem gerçeklendiğinde, bütün referansların adreslerinin bu nedenle değiştirir.

Normalde, CLR garbage collection mekanizmasını çalışması ile ilgilenmektedir, fakat bazı durumlarda garbage collector mekanizması yazılımcı tarafından aşırı belleğe ihtiyaç duyan uygulamaları çalıştırmadan önce de kullanılmak durumunda olabilir. Bunu yapabilmek için, *GC.Collect()* fonksiyonu çalıştırılır.

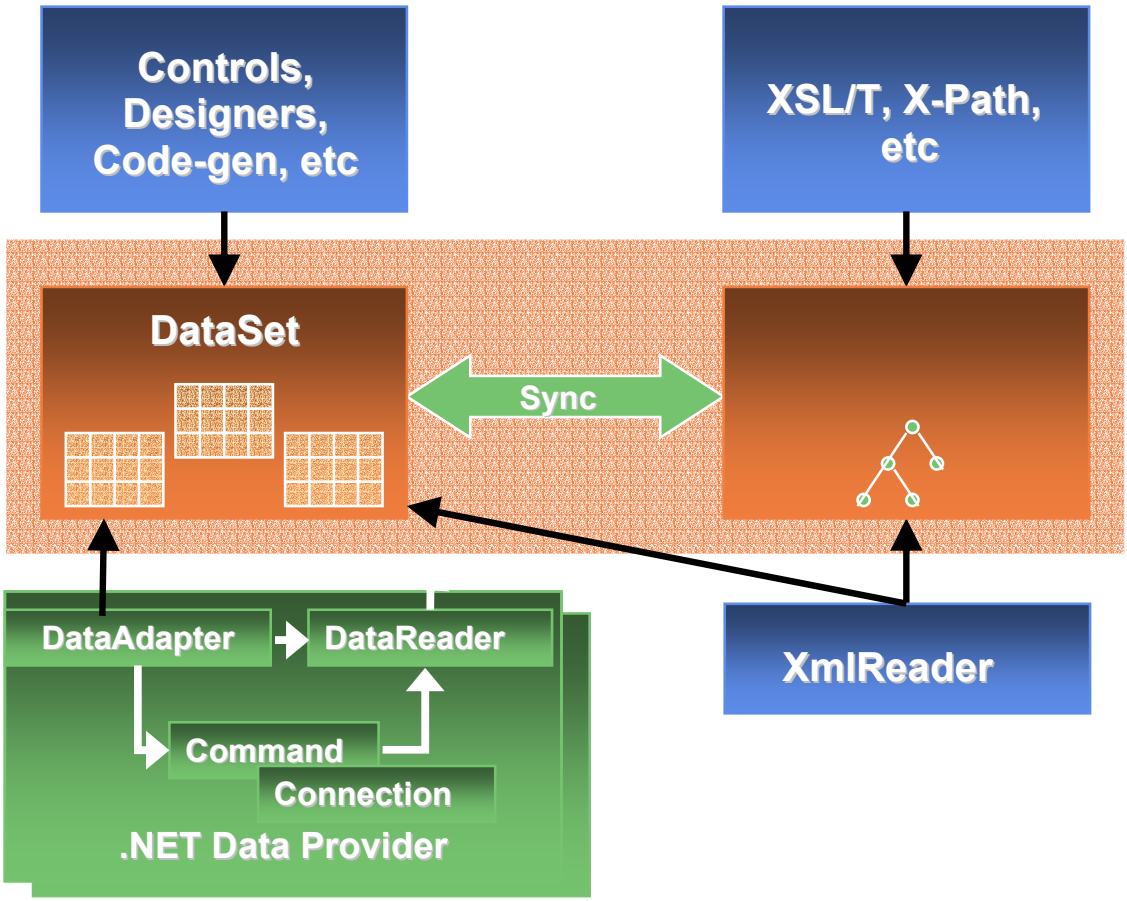
2.4 .NET TEKNOLOJİLERİ

.NET Platformu yazılım geliştiriciler için çeşitli teknolojiler sağlamıştır. Bu teknolojilerden bazıları aşağıda analatılmıştır.

2.4.1 ADO.NET

ADO.NET, Microsoft ActiveX Data Object (ADO) teknolojisinin bir sonraki neslidir. ADO.NET bağlantısız programlama modeli ve zengin bir XML desteği sunar. ADO.NET temelinde XML mimarisini kullandığı için gelecek teknolojilerine uygun bir altyapıda yer alıyor.

.NET Frameworkte ADO ve XML arasındaki ilişki aşağıdaki şekildeki gibidir:



2.4.1.1 ADO.NET VE XML

2.4.1.1 SQL VERİTABANINDAN VERİ ALMA

SQL Veritabanına bağlanma:

```
// SqlConnection tipinden bir nesen oluşturulur ve bağlantı açılır
SqlConnection cnn = new SqlConnection("server=localhost;uid=sa");
cnn.Open();

// Transaction başlatılabilir
SqlTransaction txn = cnn.BeginTransaction();

// İşlemler gerçeklenir
if(TransferFunds(fromAct, toAct, Amount) == true) // basarili
        txn.Commit();
else
        txn.Rollback();

cnn.Close();    // herzaman bağlantiyi kapat
```

Komut çalıştırma işlemi:

Çok çeşitli komutlar vardır: Insert, Update, Delete, Stored Proc, DDL,…

```
// bir Stored prosedur icin parametreli SQLCommand tipinden bir nesne oluşturulur
SqlCommand cmd = new SqlCommand("DeleteAccount", cnn);
cmd.CommandType = CommandType.StoredProcedure;
// Parametreyi oluştur
SqlParameter param = new SqlParameter("@A_ID",typeof(string));
param.Value = accountID;
cmd.Parameters.Add(param);

// Geri dönüş değeri olmayan komutu çalıştır
Int32 RowsAffected = cmd.ExecuteNonQuery();
```

Tek bir değer sorgulama:

Bu işlem ExecuteScalar komutu ile yapılır.

```
// SqlCommand nesnesi oluşturulur
SqlCommand cmd = new SqlCommand(
        "Select Balance from Accounts where AccountID = @A_ID", cnn);

// Parametre eklenir
cmd.Parameters.Add("@A_ID",accountID);

// değer sorgulanır
Decimal AccountBalance = (Decimal) cmd.ExecuteScalar();
```

Satırların İşlenmesi:

```
// Parametreli bir SqlCommand nesnesi oluştur
SqlCommand cmd = new SqlCommand(
        "Select Desc, Amt from Activity where AccountID = @A_ID", cnn);
cmd.Parameters.Add("@A_ID",accountID);

// Sonuçta bir DataReader oluştur
SqlDataReader results = cmd.ExecuteReader();

// Her bir satırı DataReaderı kullanarak goster
while(results.Read()) {
        Console.Write("Description: " + results.GetString(0));
        Console.WriteLine("Amount: " + results.GetDecimal(1));
}
```

WebForms'da VeriTabanına Bağlantı

```
<%@ Import Namespace="System.Data.SqlClient" %>
```

```
<html><head><script language="C#" runat=server>

   public void Page_Load(Object sender, EventArgs e) {

        // Create a SqlCommand and Get a DataReader
        SqlConnection cnn = new SqlConnection("server=localhost;uid=sa;");
        cnn.Open();
        SqlCommand cmd = new SqlCommand("Select * from customers", cnn);
        SqlDataReader results = cmd.ExecuteReader();

        // Bind to Results
        ActivityList.DataSource = results;
        ActivityList.DataBind();         }
</script></head><body>
                <asp:DataGrid id="ActivityList" runat="server"/>
</body></html>
```

2.4.1.2 VERİ İLE ÇALIŞMA

Veritabanından Dataset'e Veri Aktarma:

Bu amaçla DataAdapter kullanılır.

```
        // Define the Select Command
        SqlCommand selectCommand =
                new SqlCommand("Select CategoryName from Categories",cnn);

        // Create a SqlAdapter and set the Select Statement
        SqlDataAdapter adapter = new SqlDataAdapter();
        adapter.SelectCommand = selectCommand;

        // Create and Load DataSet
        DataSet categories = new DataSet("Categories");
        adapter.Fill(categories);
```

Dataset'teki Verileri Tarama:

```
foreach(DataRow customer in myDataSet.Tables["Customer"].Rows)
{
    Console.WriteLine("Orders for customer: " + customer["Name"]);
    foreach(DataRow order in customer.GetChildRows("cust_orders") )
    {
        Console.Write("\t Order ID = " + order["OrderID"]);
        Console.WriteLine("Amount = " + order["Amount"]);
    }
}
```

Datasetteki Değişiklikleri Veritabanına Aktarma:

```
SqlDataAdapter adapter = new SqlDataAdapter();

SqlCommand delete = new SqlCommand("DeleteOrder",cnn);
delete.CommandType=CommandType.StoredProcedure;
delete.Parameters.Add("@OrderID",typeof(Int32)).SourceColumn="OrderID";
adapter.DeleteCommand = delete;

SqlCommand insert = new SqlCommand("AddOrder",cnn);
insert.CommandType=CommandType.StoredProcedure;
insert.Parameters.Add("@OrderID",typeof(Int32)).SourceColumn="OrderID";
insert.Parameters.Add("@CustD",typeof(Int32)).SourceColumn="CustomerID";
insert.Parameters.Add("@Date",typeof(DateTime)).Value = DateTime.Now;
adapter.InsertCommand = insert;

SqlCommand update = new SqlCommand("UpdateOrder",cnn);
update.CommandType=CommandType.StoredProcedure;
update.Parameters.Add("@OrderID",typeof(Int32)).SourceColumn="OrderID";
update.Parameters.Add("@CustD",typeof(Int32)).SourceColumn="CustomerID";
adapter.UpdateCommand = update;
adapter.Update(ordersTable);
```

DataSet'i Cache Olarak Kullanmak:

```
private DataSet GetCategories() {
        //See if DataSet exists in Cache
        DataSet categories;
        categories = (DataSet)Cache["CategoriesDS"];

        if (categories == null)    {              // not in cache

                //Create DataAdapter and Load DataSet
                SqlDataAdapter adapter = new SqlDataAdapter(
                "Select CategoryName from Categories",cnn);
                adapter.Fill(categories);

                // Put Categories Dataset into Cache
                Cache["CategoriesDS"]=categories;
        }
}
```

Winforms'dan VeriTabanına Bağlantı:

```
// Create a Parameterized SqlCommand
SqlCommand cmd = new SqlCommand("GetAccountInfo", cnn);
cmd.CommandType=CommandType.StoredProcedure;
cmd.Parameters.Add("@A_ID",accountID);

// Populate DataSet with results
DataSet  account = new DataSet;
DataAdapter adapter = new DataAdapter(cmd);
adapter.Fill(account);

// Create WinForm DataGrid and Bind to Results
DataGrid accountGrid = new DataGrid();
accountGrid.SetDataBinding(myDataSet, "AccountList");
```

2.4.1.3 ADO.NET ve XML

DataSet'e XML kolaylıkla veri yazılabilir, yada DataSet'deki veri XML haline dönüştürülebilir. DataSet Şeması XSD olarak kaydedilebilir yada XSD'ye yüklenebilir. DataSet belirtilen XSD'den oluşturulabilir.

XML Olarak Yükleme:

```
// DataSete XML  olarak yükleme yapılır ve DataSet oluşturulur.
DataSet ds = new DataSet();
ds.ReadXml("inventory.xml");

// Add a record to the Inventory table
DataTable inventory = ds.Tables["Inventory"];
DataRow row = inventory.NewRow();
row["TitleID"]=1;
row["Quantity"]=25;
inventory.Rows.Add(row);

// Write out XML
ds.WriteXml("updatedinventory.xml");
```

2.4.2 WEB SERVİSLERİ

Bir XML web service'i, web server'da bulunan programlanabilir bir varlıktır ve standart internet protokolleri ile kullanılmaları için sunulurlar. Bir çok açıdan, XML web service'lerinin yaratılması ve kullanılması için kullanılan programlama modeliyle, COM tabanlı uygulamalar yaratmak ve kullanmak için kullanılan programlama modeli benzerdir.

Buna karşın, COM'dan farklı olarak, XML web service'leri programlama modeli basit, genel destek gören açık standartlara dayanır. Uygulamalar arasında ikili iletişimin yerine, XML web service'leri uygulamalar arasında XML mesajları transfer etmek için SOAP protokolüne

dayanan iletişim kullanırlar. Birleşik, programlanabilir web service'leri ortamında, XML iletişim için evrensel dildir, anlaşılır ve reaksiyon verilebilir tek veridir.

Web servisleri cep telefonlarından masaüstü bilgisayarlara kadar birçok platformda çalışabilmektedir. Yani platform bağımsızdırlar. Örnek olarak, borsa, hava durumu gibi bilgi veren servisler gösterilebilir.

Web servislerinin vizyonu, bağımsız, heterojen sistemleri birbirine bağlamak, Gevşek bağlı (loosely-coupled), mesajlaşma üzerine kurulu bir sistem oluşturmak, kurumsal uygulamlarca implimente edilebilmek ve açık standartlara uymaktır.

Web Servisi (ASMX , ASP.NET), Web Servisi methodlarını harekete geçirir.

SOAP(Simple Object Access Protocol):

- Mesajlaşma için kullanılacak XML tabanlı protokol

Web Service Description Language (WSDL): Web Servisi Tanımlama Dili

- Web Servislerini Tanımlar
- Bağımlılıkları belirler (örn., DataSets / XSD)

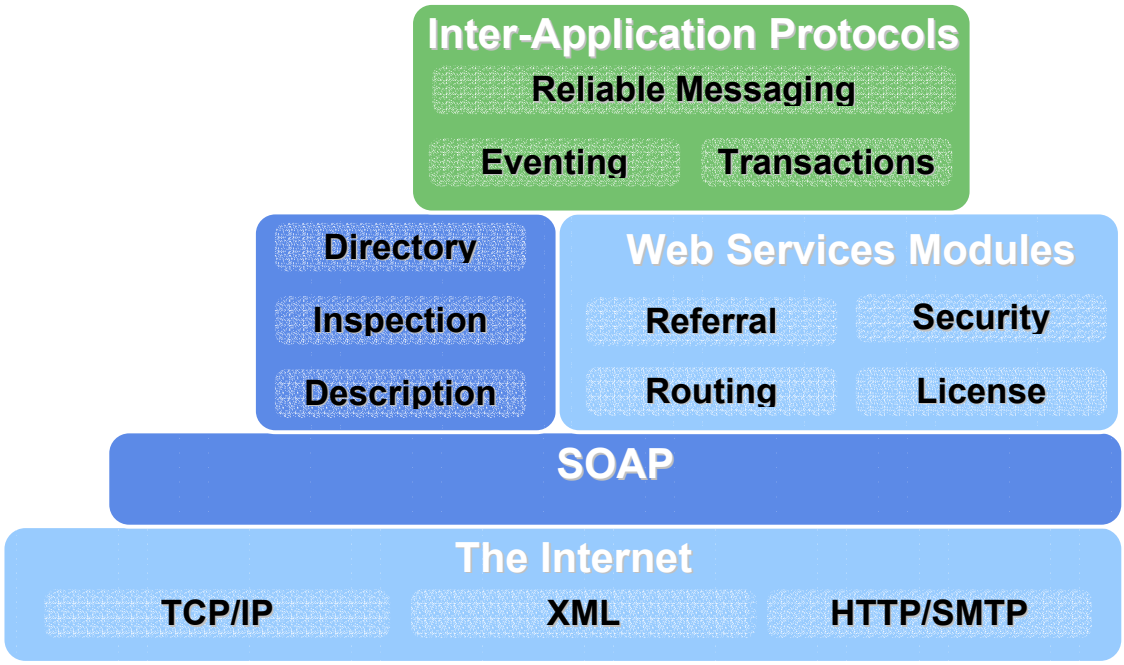
Discovery Documents (DISCO): Keşif Belgeleri

- Web servislerini yayınlar

UDDI (Universal Description Discovery and Integration)

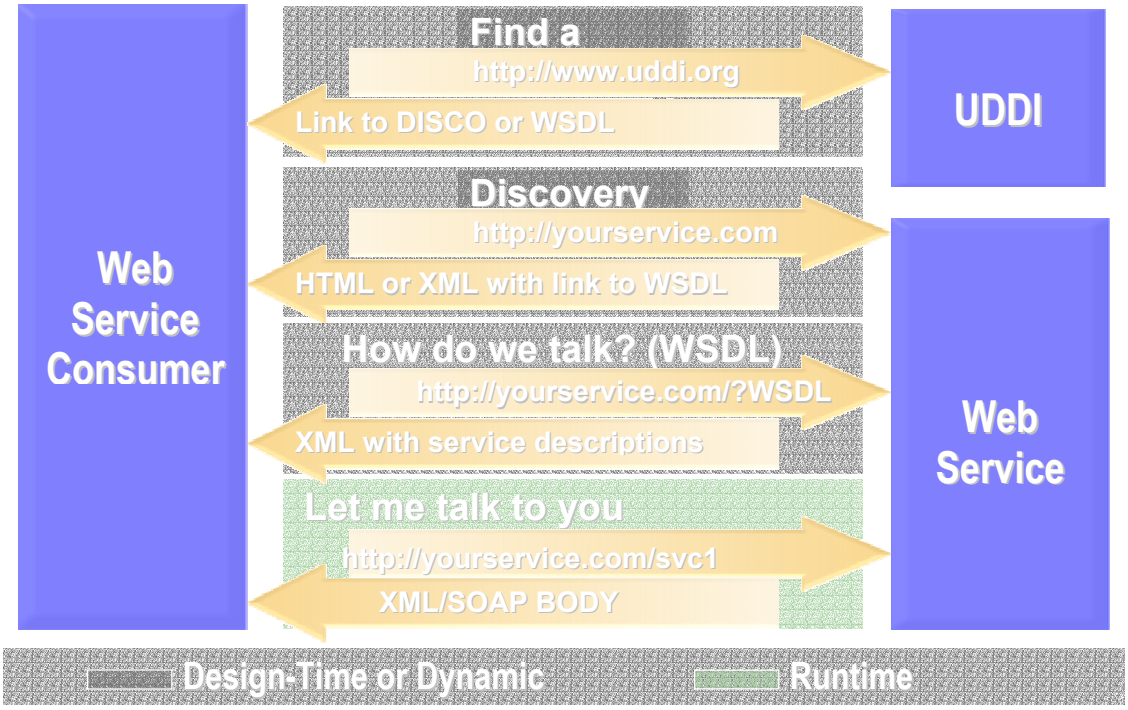
- Servisler için bir dizindir

Web Servislerinin Genel Yapısı şu şekildedir:



2.4.2.1 Web Servislerinin Yapısı

Web Servislerinin Çalışma mantığı şu şekildedir:



2.4.2.2. Web Servislerinin Çalışma Mantığı

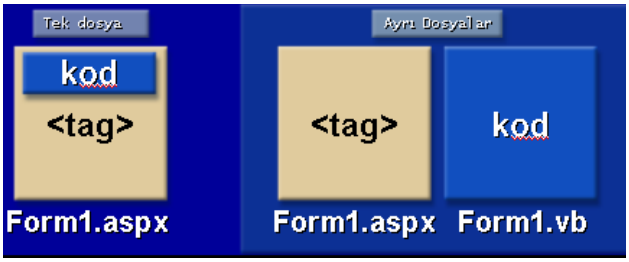
Web servisi örnegi:

```
<%@ WebService Language="VB" Class="alan_hesapla" %>
Imports System.Web.Services
Public Class alan_hesapla : inherits WebService
  <WebMethod()>
      Public Function alan_hesapla(ByVal deger1 As Integer, _
          ByVal deger2 As Integer) As Integer
          Return (deger1 * deger2)
      End Function
End Class
```

2.4.3 ASP.NET

ASP.NET, common language runtime üzerinde kurulmuş bir programlama anaçatısıdır (framework). ASP.NET bir server üzerinde kullanılarak güçlü web uygulamaları geliştirilebilir. ASP.NET web formları, dinamik web kullanıcı arayüzleri geliştirmek için kolay ve güçlü bir yol sağlar. Microsoft'un yeni vizyonu .NET ile duyurmuş olduğu internet uygulamaları ve web servisleri için sunucu taraflı yazılım geliştirmeyi kolay, güvenli ve genişleyebilir yapıda sağlayan teknolojidir.

Bir Web Form program kodu tag'ler içerir (HTML, ASP directive'leri, Sunucu Kontrolleri ve statik metin) . ASP.NET ile içeriğin program kodundan ayrışması sağlanmıştır.



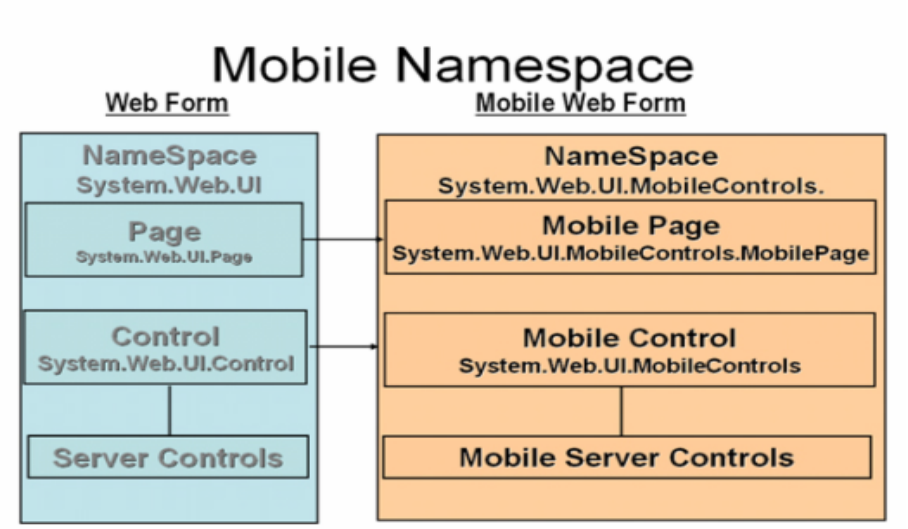
2.4.3.1 Web Forms yapısı

2.4.4 MOBİL TEKNOLOJİLER

Mobil Teknolojiler için .NET şu araçları sunmaktadır:

- Mobile Controls
- ASP.NET sayesinde tek geliştirme

- Mobil Web Formları



3.3.1: Mobil Namespace

Mobile Örnek:

```
<%@ Page Inherits="System.Web.UI.MobileControls.MobilePage" Language="VB" %>
<%@ Register TagPrefix=“mobile" Namespace="System.Web.UI.MobileControls" %>
<mobile:Form runat="server">
  <mobile:Label runat="server">                    Merhaba Dünya !
  </mobile:Label>
</mobile:Form>
```

WML Kodu:

```
<?xml version='1.0'?>
<!DOCTYPE wml PUBLIC'-//WAPFORUM//DTD WML 1.1//EN' 'http://www.wapforum.org/DTD
/wml_1.1.xml'><wml>
<head>
        <meta http-equiv="Cache-Control" content="max-age=0" />
</head>
 <card>
  <p>
        Merhaba Dünya !
        <br/>
  </p>
 </card>
</wml>
```

HTML Kodu:

```
<html>
<body>
 <form id="ctrl1" name="ctrl1" method="post“
action="helloworld.aspx?631183709659516346">
<INPUT type="hidden" name="__VIEWSTATE"
value="TJk3Y2YwNzQtNzY4OS00ODBhLTlmMjUtYjE2ZTViYzA4YjIyLDA=c7b8
0628">
<input type="hidden" name="__EVENTTARGET" value="">
<input type="hidden" name="__EVENTARGUMENT" value="">
<script language=javascript>
  <!--
  function __doPostBack(target, argument){
    var theform = document.ctrl1
    theform.__EVENTTARGET.value = target
    theform.__EVENTARGUMENT.value = argument
    theform.submit()
   }
  // -->
</script>
<div>
                Merhaba Dünya !
</div></form></body></html>
```

2.5 KAYNAK KODUN DERLENMESİ VE ÇALIŞTIRILMASI

.NET Framework'te bir uygulama geliştirildiğinde, kullanılan dilin derleyicisi, common language runtime'a uygun olduğu sürece istenilen programlama dili kullanılabilir. Derleme işlemi sonucunda bir "managed module" üretilir. Managed module fiziksel olarak bir dosyada saklanır ve bu dosya aynı zamanda "portable executable file" olarak da bilinir.

Portable executable file şu öğeleri içerebilir :

- Microsoft Intermediate Language (MSIL) : derleyici, kaynak kodu CPU bağımlı komutlardan oluşan, kolaylıkla makine diline çevrilebilen MSIL diline dönüştürür.
- Type Metadata : bu bilgi tamamen tipleri, üyeleri ve diğer referansları tanımlar ve çalışma anında common language runtime tarafından kullanılır.
- Kaynak kümesi : mesela .bmp, .jpg gibi dosyalar.

Kullanıcı bir uygulamayı çalıştırdığı zaman işletim sistemi, MSIL kodu çalıştıracak olan common language runtime'ı yükler. CPU kendisi direk olarak MSIL kodu çalıştıramayacağı için common language runtime öncelikle MSIL kodu makine koduna çevirir.

Common language runtime, bütün kodu yükleme aşamasında CPU koduna çevirmez. Bunun yerine fonksiyonlar çağırıldıkça çevirme işlemini yapar. MSIL sadece gerektiğinde derlenir. Bu işlemi gerçekleştiren common language runtime komponenti just-in-time (JIT) compiler olarak isimlendirilir. JIT derleme işlemi, hafıza ve zaman kazancı sağlar.

3 C# DİLİ

C# Windows uygulamaları geliştirmek için C++ ve Visual Basic programlama dillerine alternatif olarak Visual Studio .NET ile birlikte gündeme gelen bir uygulama geliştime dilidir. C# orta seviye bir programlama dilidir. Bir sıralama yapacak olursak C#'ın bu sıralamadaki yeri aşağıdaki gibi olacaktır.

- Assembly
- C,C++
- C#,Java
- Visual Basic
- VBA, Scripting Dilleri

C# tam anlamıyla bir Nesne Yönelimli Programlama dilidir. Bu özellik sadece C#'ın taşıdığı bir özellik değildir. C++ ve VB de Nesne Yönelimli Programlama yapısına sahiptir. Buna rağmen C++ ve VB açısından olaya yaklaşıldığında birtakım problemlerle karşılaşılmaktadır
C#, Visual Basic ile C++ arasında bulunan boşluğu gidermek için dizayn edilmiştir. C#'ın C++ ve Visual Basic arasında bir dil olduğuna iyi bir örnek olarak bellek yönetimi gösterilebilir. Visual Basic'de bellek yönetimi için yapılabilecek şey, nesnelerin kullanımı bittiğinde "set nesne=nothing" kodu ile ortadan kaldırmaktır. Bu işlemin yapılmadığı durumda olabilecek tek kötü sonuç sistem kaynakları ve belleğin gerektiğinden biraz daha uzun bir süre kullanılması olacaktır.

Diğer taraftan C++, uygulama geliştiricilere hangi değişkenin hafızada ne kadar kalacağının ayarını yapma imkanı vermek ve böylelikle hafızanın daha etkin kullanımını sağlamakla kalmaz, bunu neredeyse zorunlu kılar. C++'da yazılmış programların bellek yönetimi hatalarına maruz kalmaları ve hafızada kayıplara yol açması Visual Basic'de gerçekleşmez.

C# bellek yönetimi konusunda Visual Basic ile benzerlik gösterir, yani sorumluluk dilin kendisindedir. Ancak VB'nin aksine C#'da belleği uygulama geliştirici sınıflandırmak isterse yazdığı kodun bir bölümünü güvensiz (unsafe) olarak belirtebilir ve sonra da C++'da olduğu gibi bellek için yer ayırıp daha sonra bu kısmı boşaltabilir.

3.1 NAMESPACELER

C#, .NET taban sınıf kütüphanesinde tanımlanmış veritipleri kullanır. Bu sınıflar namespace adı altında gruplanmıştır. Namespace diğer bir tanımlama ile sınıflar kullanılırken fonksiyonel bir şekilde ulaşılmasını sağlayan mantıksal saklayıcılar denilebilir. Örnek olarak System.Winforms namespace'i WinForm uygulamaları için gerekli kontrol ve sınıfları içerir. Genellikle namespace isimlendirmesinde aşağıdaki kurala dikkat edilir.

Companyname.Technologyname

Örneğin :

```
namespace Microsoft.Office
{
    //class tanımlamaları
}
```

using keyword'ü ile de namespaceler içerisinde bulunan sınıflar çalışmaya çekilir. Namespace, using keyword'ü ile çekildikten sonra artık herhangi bir period belirtmeden namespace içerisinde bulunan tanımlamalara ulaşılabilir.

```
using Microsoft.Office;
```

3.1.1 SYSTEM NAMESPACE'İ

Common Laguage Specification (CLS) bir programlama dili tarafından desteklenmesi kaçınılmaz olan bir çok vertipini tanımlayan zengin bir kütüphaneye sahiptir. Bu kütüphanede bulunan boolean, string, integer, kronolojik ve finansal değerlerin hepsi System namespace'inin içerisinde yer alır. Bütün tipler object tipinden türerler. Böylece bir object'in referansı herhangi bir tipdeki değişkeni tutmak için kullanılabilir.

Method Tanımlama

```
class IlkKod
{
  public static void main()
  {
      Console.WriteLine("Bir Sonraki Yazida Gorusmek Uzere");
  }
}
```

public static void main() satırında bir method tanımlama söz konusudur. Yalnız bu main() isimli özel bir methoddur. Eğer her hangi bir uygulamada main() isimli bir method varsa ilk önce o icra edilir. Yani uygulama ilk önce bu method içindeki işlemleri yaparak çalışmaya başlar.

```
class Test
{
  public static void main()
  {
        Console.WriteLine("Bir Sonraki Yazida Gorusmek Uzere");
        Test baglan = new Test();
        baglan.EkranaBas();
  }
  public void EkranaBas()
  {
        Console.WriteLine("Mesajı Artık Görebilirsiniz.")
  }
  private void Deneme()
  {
         Console.WriteLine("AspNedir.com")
  }
}
```

Burada "new" keyword'u aracılığı ile Test class'ına ait bir baglan nesnesi oluşturuluyor ve EkranaBas() methodu çağrılıyor. Yalnız burada Deneme() isimli bir method daha var.

EkranaBas ile Deneme isimli bu iki method'un tanımlanması arasında farklılıkların olduğu görülebilir. EkranaBas methodu public ile başlarken Deneme methodu private ile başlıyor.

Aradaki fark:
PUBLIC: public ile başlayan methodlar dışarıdan (yani başka bir class'ın içinden) çağırılabilir.
PRIVATE: private ile başlayan methodlar ise sadece bulunduğu class içinde çağırılabilir.

3.2 C#'DA TİPLER

C#'da temelde iki tip vardır: value tipleri ve reference tipleri

3.2.1 VALUE TİPLERİ VE TANIMLANMASI

C#'daki basit value tipleri, bunların MSIL assembler ve class library'deki karşılıkları aşağıdaki tabloda verildiği gibidir:

MSIL assembler'daki adı	C#'daki adı	Class library'daki adı	Açıklama
bool	bool	System.Boolean	True/false value
char	char	System.Char	Unicode 16-bit char
float32	float	System.Single	IEEE 32-bit float
float64	double	System.Double	IEEE 64-bit float
int8	sbyte	System.Sbyte	Signed 8-bit integer
int16	short	System.Int16	Signed 16-bit integer
int32	int	System.Int32	Signed 32-bit integer
int64	long	System.Int64	Signed 64-bit integer
unsigned int8	byte	System.Byte	Unsigned 8-bit integer
unsigned int16	ushort	System.Uint16	Unsigned 16-bit integer
unsigned int32	uint	System.Uint32	Unsigned 32-bit integer
unsigned int64	ulong	System.Uint64	Unsigned 64-bit integer

3.2.1.1 C# Value tipleri

Anlaşırlılık sağlamak amacıyla her tanımlama için bir deyim kullanılabilir veya alternatif olarak aynı anda birden fazla tanımlama yapılabilir.

Basit Tanımlama	Bileşik tanımlama
int i;	int i,j,k;
int i=5;	int x=3,y=8,z=-1;

C# bellek hatalarını önlemek amacıyla tanımlanan değişkenin bir değer atamasına tabi tutulup tutulmadığını derleme sırasında inceler. Bu işlem sayesinde atanan bu değerin işletim sistemi tarafından programın akışı sırasında belleğe iade edilmesi engellenmiş olur.

```
using System;
public class ForcedInit
{
public static int main()
{
    int k;
    Console.WriteLine(k);
    return 0;
}
}
```

Yukarıdaki kod derlendiğinde bir derleyici hatası ile karşılaşılır: "use unassigned local variable k". Programı hatasız bir şekilde derleyebilmek için k değişkenine ilk değer verilmesi gerekir.

3.2.2 REFERANS TİPLERİ VE TANIMLANMASI

Referans tiplerinin bir instance'ının yaratılması sırasında new keyword'ü kullanılır. new keyword'ü kullanıldığında C# referans tipinin bir instance'ını heap'de yaratır ve ilgili instance'ın referans değerini döndürür; bu sayede geri dönen referansı bir değişkende tutabiliriz.

Bir objeye referans verildiğinde referans'ı silmek için referans'ın tanımladığı nesneye null değeri atamak yeterlidir. Bu önemli bir işlemdir çünkü Common Language Runtime (CLR) ile yönetilen Garbage Collector (GC) ilgili nesneyi otomatik olarak memory'den release etme işini yapabilmek için bunu bilmek zorundadır.

```
using System;
public class ReferenceType
{
   public int MyData;
}
public class SimpleClass
{
public static int Main()
```

```
{
        ReferenceType objA;
            //new keyword'ü kullanılarak ReferenceType instantiate ediliyor
        objA=new ReferenceType();
        objA.MyData=5;
            //Aşağıdaki kod ise ilgili objeyi Garbage Collector(GC)'i
            //kullanarak memoryden de-allocate etmek için hazırlıyor.
         objA=null;
        return 0;
}
}
```

Periyodik olarak GC (Garbage Collector), referans'ına null değeri atanmış objeleri tespit eder ve onların otomatik olarak memory'den de-allocate edilmesini sağlar.

3.2.3 DEĞİŞKEN TİPLERİ DÖNÜŞÜMLERİ

Program buglarını ortadan kaldırmak için (bug-free programs), C# çok güçlü bir dildir. Değişkenler arasındaki dönüşümler sadece ihtiyaç duyulduklarında kullanılmalıdır. Örnek verecek olursak bool tanımlı bir değişkeni string değişken tipindeki bir değere dönüştürmek için bunun belirtilmesi gerekmektedir. Bu sayede program otomatik olarak değişken tiplerini dönüştürmekle uğraşmayacaktır, bu da programın daha hızlı çalışmasını sağlayacaktır.

```
using System;
public class Donusum
{
    public static int Main()
    {
        bool MyBool;
        string i,j;
        MyBool=true;
         i=MyBool.ToString();
        MyBool=false;
        j=MyBool.ToString();
        Console.WriteLine(i);
        Console.WriteLine(j);
        Console.ReadLine();
        return 0;
    }
}
```

4 MICROSOFT SQL SERVER 2000

SQL Server, OLTP ve OLAP ortamlarını birleştiren bir teknoloji ve ürünler ailesidir. SQL Server bir ilişkisel veritabanı yönetim sistemidir. SQL Server,

- Transaction'lar ve analizler için data saklama işini yönetir.
- Client uygulamalarından gelen isteklere cevap verir.
- Bir client ve SQL Server arasında istekleri göndermek için Transact-SQL, Extensible Markup Language (XML), multidimensional expressions (MDX), veya SQL Distributed Management Objects (SQL-DMO)'leri kullanır.

4.1 VERİ SAKLAMA MODELLERİ

SQL Server iki tip veritabanı yönetir : online transaction processing (OLTP) and online analytical processing (OLAP) veritabanarı.

4.1.1 OLTP VERİTABANLARI

OLTP veritabanındaki veriler genel olarak ilişkisel tablolarda saklanırlar. Böylece gereksiz bilgiler azaltılır ve update hızı arttırılır. SQL Server, çok sayıda insanın transactionlar yapmasına ve aynı anda gerçek zamanlı verileri değiştirmesine olanak sağlar.

4.1.2 OLAP VERİTABANLARI

OLAP teknolojisi büyük miktarda verileri, bir analizcinin gerçek zamanlı olarak hızlı bir şekilde veriye erişebileceği şekilde organize eder ve özetler. SQL Server 2000 Analysis Services bu veriyi, işletme raporlamaları ve analizlerden veri modelleme ve karar desteğe kadar bir dizi işletme çözümlerini desteklemek için organize eder.

4.2 SQL SERVER VERİTABANLARI

Her SQL Server iki tip veritabanına sahiptir: system databases (sistem veritabanları) ve user databases (kullanıcı veritabanları). System Database'leri bir bütün olarak SQL Server hakkında bilgi tutarlar. SQL Server, sistem veritabanlarını kullanarak sistemi yönetir. User database'leri kullanıcıların oluşturduğu veritabanlarıdır.

SQL Server yüklendiği zaman SQL Server Setup, system database'leri ve örnek user database'leri yaratır. Aşağıdaki tablo bu database'leri açıklıyor :

Database	Açıklama
master	Kullanıcı account'ları, ayarlanabilir çevre değişkenleri ve sistem hata mesajları gibi bilgileri saklayarak kullanıcı veritabanlarını ve SQL Server'ı bütün olarak kontrol eder.
model	Yeni kullanıcı veritabanları için örnek ve prototip sağlar.
tempdb	Geçici tablolar veya diğer geçici saklama ihtiyaçları için saklama alanı sağlar.
msdb	Scheduling bilgilerinin ve iş tarihinin saklanması için alan sağlar.
distribution	Replikasyonda kullanılan transaction verilerini ve tarihi saklar.
pubs	Öğrenme aracı olarak bir örnek veritabanı sağlar.
northwind	Öğrenme aracı olarak bir örnek veritabanı sağlar.

4.2.1. SQL Server Veritabanları

4.3 SQL SERVER' DA GÜVENLİK

SQL Server kullanıcılarına iki seviyede güvenlik sunar, ilk adım login olmadır. Diğer seviyede güvenlik, kullanıcı rollerine göre sağlanmaktadır. Kullanıcı SQL Server' a iki türlü login olabilir:

- Windows Authentication: Kullanıcı Windows 2000'den direkt olarak sunucuya bağlanabilir. Sistem yöneticisinin Windows 2000'den bağlanacak kişiyi daha önceden SQL Server için geçerli bir kullanıcı olarak tanımlaması gerekmektedir.
- SQL Server Authentication: Kullanıcı, SQL Server'a daha önceden sistem yöneticisinin tanımladığı kullanıcı adı ve şifresi ile bağlanır.

4.4 TRANSACT-SQL PROGRAMLAMA DİLİ

Transact-SQL ANSI-SQL ISO standartının SQL Server için uyarlanmış halidir.

4.4.1 VERİ KONTROL DİLİ İFADELERİ

Veri kontrol dili ifadeleri (Data Control Language Statements), veritabanı kullanıcılarının sahip olduğu yetkilerde değişiklik yapmak için kullanılır.

- GRANT : Kullanıcıya belirlenen veri ile çalışma ve Transact-SQL ifadelerini çalıştırma yetkisi verir.
- DENY : Kullanıcının, tanımlanmış grup ve rollerin belirtilen veriye ulaşmasını engeller.
- REVOKE : Daha önceden verilmiş izni ya da kısıtlamayı kaldırır.

4.4.2 VERİ TANIMLAMA DİLİ İFADELERİ

Veri Tanımlama Dili İfadeleri (Data Definition Language Statements), Veritabanları, tablolar ve kullanıcı tarafından oluşturulan veri tipleri tanımlamalarının yapılarak veritabanlarının oluşturulması için kullanılır.

- CREATE *obje_tipi obje_adı*
- ALTER *obje_tipi obje_adı*
- DROP *obje_tipi obje_adı*

4.4.3 VERİ KULLANIM DİLİ İFADELERİ

Veri kullanım dili ifadeleri (Data Manipulation Language Statements) ile veriyi sorgulayıp ve üzerinde değişiklikler yapabiliriz.

- SELECT
- INSERT
- UPDATE
- DELETE

4.4.4 LOCAL DEĞİŞKENLER

Local değişkenler, şu şekilde deklere edilir :

```
DECLARE { @local_değişkenin_adı data_tipi } [ ,…n ]
SET @local_değişkenin_adı = ifade
```

4.4.5 TRANSACTİON KULLANIMI

Transactionlar, küme olarak düşünülebilen prosesler grubudur. SQL Server, veri bütünlüğünü sağlamak için değişikliklerin tümünü aynı anda yapar, eğer bu sırada bir problem oluşursa yapılan değişikliklerin tümü iptal edilir.

Örnek :

```
BEGIN TRANSACTION
UPDATE savings
    SET balance = ( amount - 100 )
    WHERE custid = 78910
IF @@ERROR < > 0
    BEGIN
      RAISERROR ( ' Transaction not complete due to savings accounts problem. ', 16, -1 )
      ROLLBACK TRANSACTION
    END
UPDATE checking
    SET balance = ( amount + 100 )
    WHERE custid = 78910
IF @@ERROR < > 0
    BEGIN
      RAISERROR ( ' Transaction not complete due to savings accounts problem. ', 16, -1 )
      ROLLBACK TRANSACTION
    END
COMMIT TRANSACTION
```

4.5 VERİTABANLARININ OLUŞTURULMASI VE YÖNETİLMESİ

Veritabanlarının oluşturulma formatı genel olarak aşağıdaki gibidir :

```
CREATE DATABASE Sample
ON
    PRIMARY ( NAME = SampleData,
    FILENAME = ' C:\ Program Files \ .. \ .. \ Data \ Sample.mdf ',
    SIZE = 10MB,
```

```
    MAXSIZE = 15MB,
    FILEGROWTH = 20% )
LOG ON
    ( NAME = SampleLog,
    FILENAME = ' C:\ Program Files \ .. \ .. \ Data \ Sample.ldf ',
    SIZE = 3MB,
    MAXSIZE = 5MB,
    FILEGROWTH = 1MB )
COLLATE SQL_Latin1_General_Cp1_CI_AS
```

SIZE : Bu parametre, veri ya da log dosyasının büyüklüğünü tutar. Verilebilecek maximum değer 512 KB' dır.
MAXSIZE : Bu parametre, dosyanın büyüyebileceği maximum büyüklüğü tutar. Eğer bu değer girilmemişse dosya disk dolana kadar büyüyebilir.
FILEGROWTH : Bu parametre, veritabanının büyüme miktarını tutar.
COLLATION : Bu parametre, veritabanının dil yapısını tutar.

Veritabanı büyüdükçe, veritabanı dosyası otomatik ya da manuel olarak büyütülebilir. Eğer bir veritabanı artık kullanılmayacaksa bu yapıyı sistemden kaldırabiliriz.

Örnek : Aşağıdaki örnekte SampleLog dosyasının boyutu 15MB olarak büyütülmüştür.

```
ALTER DATABASE Sample
MODIFY FILE ( NAME = ' SampleLog ',
SIZE = 15MB )
GO
```

Örnek : Sample veritabanın %25 küçülmesi,
DBCC SHRINKDATABASE (Sample, 25) şeklindedir
DROP DATABASE veritabanı_adı [,…n] ile yaratılmış veritabanlarını kaldırabiliriz.

4.6 VERİ TİPİ VE TABLO TANIMLANMASI

Veri tipleri, her kolon için girilebilecek veri değerlerini belirler. SQL Server, bir çok veri tipi sağlamaktadır. SQL Server'ın sağladığı veri tiplerinden ANSI uyumlu olanları aşağıdaki tabloda gösterilmiştir :

genel data tipi	SQL Serverdaki data tipi	byte sayısı
İnteger	int bigint Smallint	4 8 2, 1
exact number	decimal Numeric	2-17
approximate numeric	float Real	8 4
Monetary	money smallmoney	8 4
date and time	datetime smalldatetime	8 4
Character	char varchar Text	0-8000 0-2 GB
unicode character	nchar nvarchar Ntext	0-8000 4000 0-2 GB
Binary	binary Varbinary	0-8000
İmage	İmage	0-2 GB
global identifier	uniqueidenfier	16
Special	bit cursor uniqueidentifier timestamp sysname table sql_variant	1 0-8 8 256 0-8016

4.6.1 SQL Server Veri Tipleri

4.6.1 VERİ TİPİ OLUŞTURULMASI VE KALDIRILMASI

sp_addtype prosedürü ile kullanıcı kendine özgü veri tipleri oluşturabilir, sp_drop prosedürüylede yarattığı bu tipleri kaldırabilir.

```
EXEC sp_addtype city, ' nvarchar ( 15 ) ', NULL
EXEC sp_addtype region, ' nvarchar ( 15 ) ', NULL

EXEC sp_droptype city
```

4.6.2 TABLO OLUŞTURULMASI VE KALDIRILMASI

Veritabanlarında tablo oluşturma formatı genel olarak aşağıdaki gibidir :

```
CREATE TABLE dbo.Categories
    ( CategoryID int IDENTITY ( 1, 1 ) NOT NULL,
    CategoryName nvarchar ( 15 ) NOT NULL,
    Description ntext NULL,
    Picture image NULL )
```

Veritabanlarında tablo kaldırma formatı genel olarak aşağıdaki gibidir :

DROP TABLE *table_adı [,...n]*

4.7 VERİ BÜTÜNLÜĞÜ (DATA INTEGRİTY)

Veritabanı oluşturulurken dikkat edilmesi gereken en önemli noktalardan birisi de veri bütünlüğünün sağlanmasıdır. Veri bütünlüğü, veritabanında saklanan verinin tutarlılığı ve geçerliliği anlamına gelir. Veri bütünlüğünün değişik çeşitleri vardır:

Kolon Bütünlüğü (Domain Integrity) : Bir kolon için geçerli değerler kümesini ve bu değerlerin null olup olamayacağını ve belirler.

Tablo Bütünlüğü (Entity Integrity) : Bir tabloda primary key olarak belirtilen kolon mutlaka unique olmalıdır.

Referans Bütünlüğü (Referantial Integrity) : Primary key ile foreign key ilişkisine dayanır. Foreign keyin referans ettiği tabloda mutlaka primary key değerinin karşılığı bulunmalıdır.

4.8 KISITLAR (CONSTRAINTS)

Kısıtlar, veri bütünlüğünü sağlayan standart methodlardır. Her veri bütünlüğü çeşidinin kendine has kısıtları vardır. Bu aşağıdaki tabloda gösterilmiştir:

Bütünlük tipi	**Kısıt tipi**
Kolon	default
	check
	referential
tablo	primary key
	unique
refential	foreign key
	check

4.8.1. Kısıt Tipleri

Default : Bir kolona INSERT ifadesi ile herhangi bir değer girilmemiş ise SQL Server default olarak belirtilmiş değeri bu kolona atar.

Check : Bir kolona girilebilecek değerleri belirler.

Referential : Başka tablolara referans veren kolonların güncellenip güncellenemeyeceğinin kontrolünü yapar.

Primary Key : Her satırdaki değer unique olmalıdır ve hiç bir satır null olmamalıdır.

Unique : Satırlarda tekrarlanan değerler olmamalıdır, fakat satırlar null değerler içerebilir.

Foreign Key : Aynı ya da farklı bir tabloda primary key ile örtüşen kolon ya da kolonları tanımlar.

4.9 INDEX

Index kullanmak veritabanı performansını oldukça yükseltir. Tablolardaki satırlar veri sayfalarında tutulur. Her veri sayfası 8 KB bilgi içermektedir, sekiz veri sayfası grubuna da extent adı verilir. Heap ise bir tablonun veri sayfalarının tümüdür. SQL Server, veriye iki şekilde ulaşabilir :

- Tablonun başlangıcından başlayarak her satırı veri sayfaları ile gözden geçirir ve sorgulama kısıtlarına uyan satırları görüntüler.

- Index tree yapısını sorgulama kısıtlarına uyacak şekilde arar ve sorgulama sonucunu görüntüler.

Index kullanımı, veriye erişim hızını arttırır ve satırların unique olmasını zorunlu kılar. Fakat, insert, update, delete gibi işlemlerde index de güncellendiğinden dolayı hız düşer. Indexler diskte yer tutarlar, o yüzden sık kullanılmayacak indexler kullanılmamalıdır. Indexler, "sysindex" adlı sistem tablosunda tutulurlar. Text alanı için kurulmuş indexler varsa bunlar veritabanına kurulmaz. Index yaratıldıktan sonra mutlaka veritabanının backup'ı alınmalıdır.

4.9.1 KÜMELENMİŞ İNDEXLER (CLUSTERED INDEXES)

Kümelenmiş İndexler, sık erişilen kolonlara ya da içeriklerine erişim için kullanılırlar. Kümelenmiş İndexler ile ilgili temel özellikler şöyledir :

- Tüm tablolar sadece bir tane Kümelenmiş İndexe sahip olabilirler.
- Tablonun fiziksel satır sırası ile index'teki satır yapısı aynıdır.
- Bir kümelenmiş index'in ortalama büyüklüğü tablonun %5'i kadardır.
- Index yaratılırken veritabanın bulunduğu disk kapasitesinin belirli bir miktarı kullanılır.(geçici olarak)

4.9.2 KÜMELENMEMİŞ İNDEXLER (NONCLUSTERED INDEXES)

Kümelenmemiş İndexler, kullanıcıların veriye erişmeleri için birden fazla yöntem olduğunda kullanılırlar. Kümelenmemiş İndexler ile ilgili temel özellikler şöyledir :

- Index tipi belirtilmemişse, default olarak Kümelenmemiş İndex yatatılır.
- Kümelenmemiş İndex sayfalarının yapısı tablonun fiziksel yapısından farklıdır.
- Her tablo için 249 tane Kümelenmemiş İndex yaratılabilir.

4.9.3 HANGİ KOLONLAR INDEXLENMELİDİR?

- Primary ve Foreign key'ler
- Sık olarak arama yapılan kolonlar

- Sık olarak sıralı erişim yapılan kolonlar

4.9.4 HANGİ KOLONLAR INDEXLENMEMELİDİR?

- Çok az arama yapılan kolonlar
- Unique değeri az olanlar
- Data tipleri text,ntext ya da image olanlar

4.10 VİEW

Bir view, yapılan sorgulama sonucu bilgilerin daha sonraki kullanımlar için veritabanında nesne olarak tutulduğu yapıdır.

4.10.1 VIEW'İN AVANTAJLARI

- Önemli ve uygun verilere odaklanma imkanı sağlar.
- Kişilerin istenmeyen verilere ulaşmasını engeller.
- Kompleks veritabanı dizaynını basite indirger.
- Kişilerin izinlerinin yönetimini kolaylaştırır.
- Performansı arttırır.

4.11 STORED PROCEDURE

Stored prosedürler, Transact SQL ifadelerinin sunucuda toplu halde tutulmasından oluşurlar. Stored prosedürler ilk oluşturulduklarında sırasıyla sentaks konrolü, obje kontrolü, optimizasyon, derleme ve çalıştırma aşamalarından geçer. Bir stored prosedür ikinci defa çalıştırıldığında ilk dört aşamadan geçmesine gerek kalmaz, o yüzden daha hızlı çalışır. SQL Server beş tip stored prosedürü destekler:

1. Sistem Stored Prosedürleri (System Stored Procedures - sp): Master veritabanında tutulan bu prosedürler, sistem tablolarından bilgi almak için kullanılırlar. Sistem yöneticileri tarafından sistem tablolarının güncellenmesini sağlarlar.
2. Lokal Stored Prosedürler (Local Stored Procedures): Bireysel kullanıcı veritabanında yaratılan prosedürlerdir.

3. Geçici Stored Prosedürler (Temporary Stored Procedures): # işareti ile başlayanlar lokal, ## işareti ile başlayanlar global olarak tanımlanmıştır. Lokal olanları tek kullanıcı, global olanları tüm kullanıcıların kullanımı açıktır.

4. Uzak Stored Prosedürler (Remote Stored Procedures): SQL Server'ın eski bir özelliğidir. Dağınık sorgulamalarda artık bu özelliği desteklemektedir.

5. Genişletilmiş Stored Prosedürler (Extended Stored Procedures - xp) : Dinamik link libraryleri (DLLs), SQL Server' ın dışında çalıştırıldığında bu prosedürler çalışır.

4.11.1 STORED PROSEDÜRLERİN AVANTAJLARI

- Kullanıcının veritabanındaki tablolara direkt olarak erişmesine gerek kalmaz, bunu stored procedürler yapar.
- Güvenlik mekanizmasını sağlarlar. Eğer kullanıcının bazı veritabanlarına ve tablolara erişim hakkı yoksa işlemini stored prosedürler yardımıyla yapabilir.
- Performansı arttırırlar.
- Ağ trafiğini azaltırlar. Bir çok Transact -SQL ifadesini ağ üzerinden yollamaktansa tek bir ifade ile istenilen işlem yapılabilir.

4.12 TRİGGER

Triggerlar, veri tablolarında bir değişiklik olduğunda devreye giren stored prosedürlerdir. Trigger tabloları olarak adlandırılan özel tablolarda tanımlanırlar. Triggerlar direkt olarak çağrılmazlar, insert, update, delete gibi çeşitli işlemler sonunda otomatik olarak devreye girerler.

5. YAPILAN ÇALIŞMA

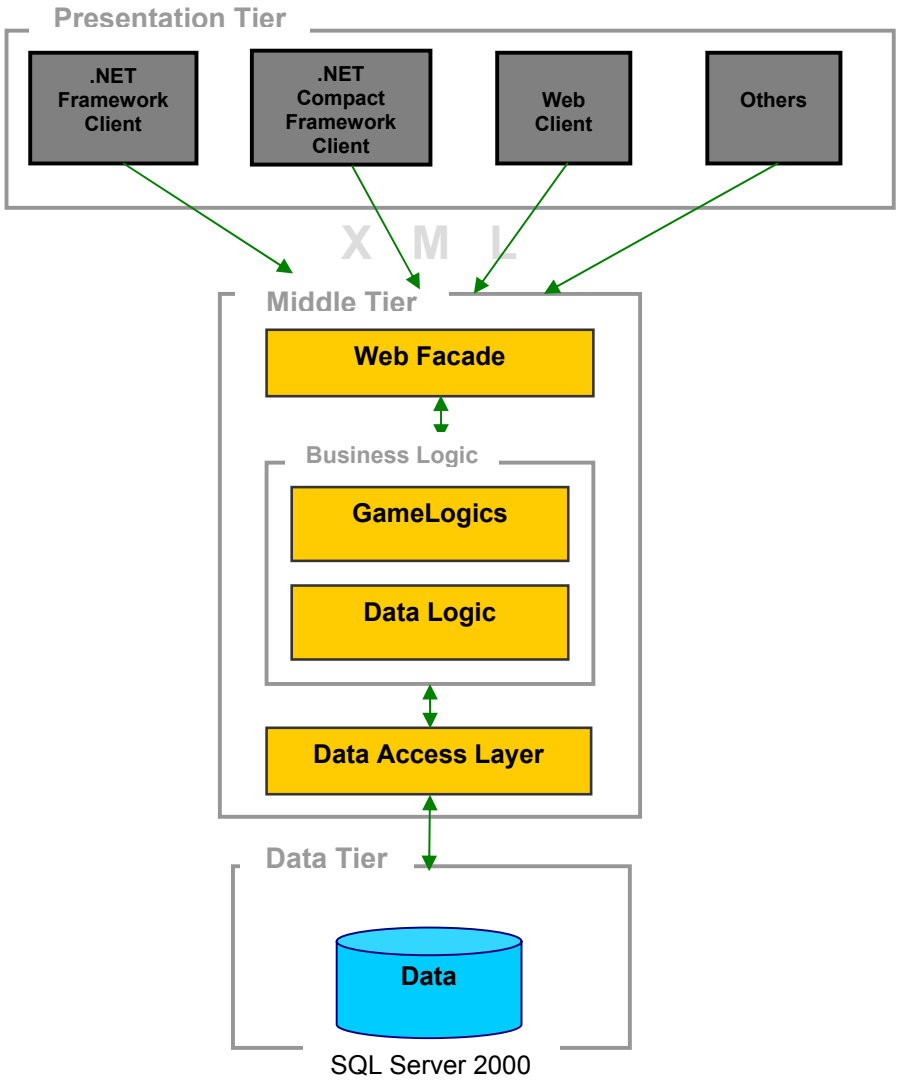
5.1. GameNETwork PROJESİ

Bu bölümde GameNETwork'ün tasarımı alt başlıklar halinde incelenecektir. Öncelikle projenenin mimarisi tanıtılacak, akabinde projenin ölçeklenebilirliğini, güvenliğini ve genişletilebilirliğini artıran konular irdelenecektir.

5.1.1. GAMENETWORK'ÜN MİMARİSİ

GameNETwork, bir istemci sunucu uygulaması olduğundan istemci ve sunucunun mimarileri ayrı ayrı ele alınacaktır.

5.1.1.1 SUNUCU MİMARİSİ



Yukarıdaki şekilden de görüldüğü gibi GameNETwork n - katmalı (n - tier) bir mimariye sahiptir. Bu katmanları 3 ana katman grubunda toplayabiliriz. Bu katmanlar sunum katmanı, orta katman ve veri katmanıdır.

5.1.1.1.1. SUNUM KATMANI (PRESENTATİON TİER)

Bu katman mimarinin en üstünde yer alır ve kullanıcıya sistemi kullanmak için bir arayüz sağlamakla görevlenidirilmişitir. Bu katman farklı platformlardan almış olduğu girdileri sunucuya gönderir ve sunucudan istenen hizmeti çalıştırmasını ister. Bu katmanda birden çok platforma ilişkin istemci arabirimleri vardır. Bizim projemiz çerçevesinde gerçekleştirmiş olduğumuz arayüzler şunlardır:

.NET Framework arayüzü (PC'ler için)

.NET Compact Framework arayüzü (PDA'ler ve diğer mobil cihazlar için)

Web arayüzü (İnternet gözatıcıları (browser'lar) için)

WAP arayüzü (cep telefonalrı için)

Bu arayüzler de artırılabilir. Örneğin bir JAVA arayüzü yazılarak bütün JAVA'yı destekleyen cihazlar sisteme entegre edilebilir. Web arayüzü yazılmış olduğu için hemen hemen bütün cihazlar sistemi kullanılabilir haldedir. Sunum katmanında bu kadar çok platform için arayüz olması ve bu arayüzlerin artırılabilir olması GameNETwork'ü platform bağımsız bir yazılım haline getirmektedir. Bu katmanda ASP.NET, Smart Device Programmability, Mobile Controls teknolojikerinden yararlanılmıştır.

5.1.1.1.2. ORTA KATMAN (MİDDLE TİER)

Orta katmanda kendi içerisinde 3 alt katmana ayrılmaktadır: Web Cephesi (Web Facade), Ticaret Mantığı (Business Logic), Veri Erişim Katmanı (Data Access Layer).

5.1.1.1.2.1. WEB CEPHESİ KATMANI (WEB FACADE LAYER)

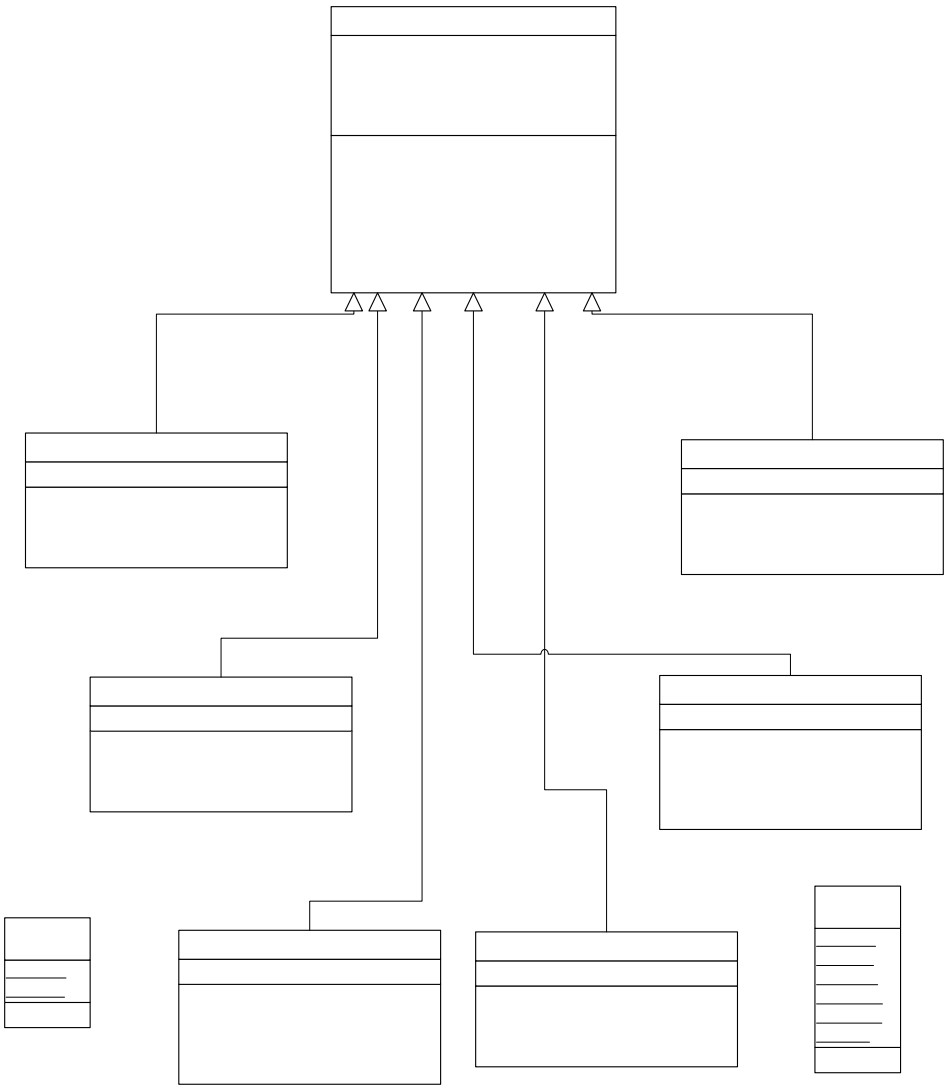
Bu katmanda web servisleri bulunmaktadır. Web servisileri, SOAP, XML, HTTP protokol yığını ile haberleştiğinden hemen her platformdan gelen verileri işleyebilirler. Farklı ortamlardan web cephesi katmanına gelen istekler doğrultusunda, bu katman iligili nesneyi ticaret mantığında oluşturur ve ondan ilgili hizmeti talep eder. Hizmeti aldıktan sonra o nesne yok edilir.

5.1.1.1.2.2. TİCARET MANTIĞI KATMANI (BUSİNESS LOGİC LAYER - BLL)

Bu katman da kendi içinde iki alt katmana ayrılır: Oyun Mantığı ve Veri Mantığı

5.1.1.1.2.2.1. OYUN MANTIĞI KATMANI (GAME LOGİCS LAYER)

Bu katmanda hizmeti verilen oyunların algoritmaları tutulur. Web servisleri bu algoritmaları kullanırlar. Bu katmandaki algoritmalardan komponent mantığı ile
çalıştığından istemci tarafında elde edilmiş esneklik sunucu tarafında da elde edilmektedir. Aşağıda satranç oyununa ait sınıf hiyerarşisi ve sınıf yapılarını gösteren UML diyagramı yer almaktadır.



5.1.1.1.2.2.2 VERİ MANTIĞI KATMANI (DATA LOGİC LAYER)

Bu katman üst katmanların veri erişim katmanı ile tutarlı bir iletişim kurmasından sorumludur. Veri erişim katmanının parametrelerin geçilecek argümanların doğruluk kontrolü bu katmanda yapılır. Bu katmanın diğer bir görevi de verinin bütünlüğünü sağlamktır. Veri katmanında meydana gelebilecek hatalar (Exception'lar) bu katmanda yakalanarak üst katmana GameNETwork için yazılmış özel bir exception sınıfı türünden nesneyi iletirler. Bu exception nesnesi de üst katmanlarda yakalanır ve ilgili kullanıcı sistemden logout edilir. Aynı zamanda bu kullanıcının veritabanında yapmış olduğu state management ile ilgili değişiklikler de silinir. Bu kısım state management ve exception handling başlıklarında detaylı olarak incelenecektir. Sonuç olarak geliştirilmiş bu mekanizma ile veri katmanının bütünlüğü sağlanmış olur.

5.1.1.1.2.3. VERİ ERİŞİM KATMANI (DATA ACCESS LAYER)

Bu katman bütün sistemin veritabanı ile olan ilişkilerini düzenler. Bu katmanın veri tabanı ile etkileşimi ya sql ifadelerini icra ederek olur veyahut ilgili stored procedure'un icra edilmesi ile olur. Bu katmanda ADO.NET teknolojisinden yararlanılmıştır.

5.1.1.1.3. VERİ KATMANI (DATA TİER)

Veri katmanında veri tabanı bulunmaktadır. Veri tabanında GameNETwork'e ait tablolar, view'lar, stored procedureler ve triggerlar yer almaktadır. GameNETwork platformunun veri tasarımı, veriler arası ilişkileri, ve verilerin kendisinin saklanması için Microsoft SQL Server 2000 kullanılmıştır. Microsoft SQL Server 2000'i seçmemizin sebebi ilişkisel bir veritabanı olması ve tasarımı ve tablolar arası ilişkilendirmeleri kolay bir şekilde yapmamızı sağlamasından dolayıdır. Ayrıca Microsoft SQL Server 2000 genişleyebilir bir yapıda ve Internet uygulamaları için XML desteğine verebilen bir veritabanıdır. Aynı zamanda, çok sayıda kullanıcıya aynı anda hizmet verebilmeside seçim nedenlerinden birisidir.

5.1.1.1.3.1 TABLOLAR

GameNETwork veritabanında yedi tablo bulunmaktadır. Bu tablolar: Users, OnlineUsers, SavedGame, Score, CurrentGames, Game, PurchasedGame tablolarıdır.
Aşağıda her bir tabloya ilişkin detaylı açıklama bulunmaktadır.

5.1.1.1.3.1.1 USERS TABLOSU

	Column Name	Data Type	Length	Allow Nulls
🔑	UserID	int	4	
	UserName	nvarchar	50	
	Name	nvarchar	50	
	Surname	nvarchar	50	
	Password	nvarchar	50	
	email	nvarchar	50	
	Age	smallint	2	

5.1.1.1.1 Users Tablosu

Users Tablosu, GameNETwork sistemine üye olan kullanıcılar hakkındaki bilgileri tutar. Tablo yukarıdaki şekilde belirtilen alanlardan oluşmaktadır. Bu alanlara ilişkin veri tipleri ve NULL olup olamayacakları hakkındaki bilgide yukarıdaki şekilde mevcuttur.

Her kullanıcı, tekil olan kullanıcı ismi ile sistem tarafından ayırt edilir. Sistem işlemleri hızlandırmak için kullanıcıya ait olan ve yine tekil olan UserID'i işlemlerinde kullanır. Kullanıcıya ait şifrede, kullanıcının sisteme tekrar griebilmesi için veritabanında tutulur.

5.1.1.1.3.1.2 GAME TABLOSU

	Column Name	Data Type	Length	Allow Nulls
🔑	GameID	smallint	2	
	GameName	nvarchar	50	
	GamePrice	decimal	9	
	GameDesc	nvarchar	100	
	AssemblyName	nvarchar	50	
	DefaultBoardLayout	nvarchar	4000	✓
	DefaultGameStateFlag	nvarchar	1000	✓
	DefaultGameLogicFlag	nvarchar	1000	✓

5.1.1.2.1 Game Tablosu

Bu tabloda, GameNETwork sisteminin sağlamış olduğu oyunlara ilişkin bilgiler tutulmaktadır. Bu oyunlara ilişkin başlangıç değerleri ve oyunların ucretleri hakkındaki bilgilerde burada tutulmaktadır. Tablodaki alanlara ilişkin ayrıntılı bilgi yukarıdaki şekilde bulunmaktadır.

5.1.1.1.3.1.3 ONLİNEUSERS TABLOSU

	Column Name	Data Type	Length	Allow Nulls
🔑	UserID	int	4	
	AvailableInfo	smallint	2	
	RequesterID	int	4	✓
	GameID	smallint	2	✓
	SaveName	nvarchar	50	✓
	StartFlag	smallint	2	✓
	DenyFlag	smallint	2	✓

5.1.1.3.1 OnlineUsers Tablosu

Bu tabloda GameNETwork sistemine bağlanmış olan kullanıcıların listesi tutulmaktadır. Ayrıca, bu kullanıcıların başka kullanıcılar ile bağlantı kurup kurmadıkları hakkında bilgide bu tabloda tutulmaktadır. Tablodaki alanlara ilişkin ayrıntılı bilgi yukarıdaki şekilde bulunmaktadır.

5.1.1.1.3.1.4 CURRENTGAMES TABLOSU

	Column Name	Data Type	Length	Allow Nulls
🔑	ConnectionID	bigint	8	
	UserID1	int	4	
	UserID2	int	4	
	GameID	smallint	2	
	Side1	nvarchar	50	✓
	Side2	nvarchar	50	✓
	BoardLayout	nvarchar	4000	✓
	GameStateFlags	nvarchar	1000	✓
	GameLogicFlags	nvarchar	1000	✓
	SaveName	nvarchar	500	✓

5.1.1.4.1 CurrentGames Tablosu

Bu tabloda, GameNETwork sisteminde şu anda üyeler arasında oynanan oyunlar hakkındaki bilgiler tutulur. Oyunlardaki nesneler hakkındaki bilgiler burada tutulur. Tablodaki alanlara ilişkin ayrıntılı bilgi yukarıdaki şekilde bulunmaktadır.

5.1.1.1.3.1.5 PURCHASEDGAME TABLOSU

	Column Name	Data Type	Length	Allow Nulls
🔑	PurchaseID	int	4	
	GameID	smallint	2	
	UserID	int	4	
	PurchaseDate	datetime	8	

5.1.1.5.1 PurchasedGame Tablosu

Bu tabloda, GameNETwork sistemine üye olmuş olan kullanıcıların, satın almış olduğu oyunlara ilişkin bilgiler tutlmaktadır. Herhangi bir üyenin, ilgili oyunu satın alıp almadığı bilgisi bu tablodan öğrenilebilir. Ayrıca, oyunların satın alınma tarihleride tutulmaktadır. Tablodaki alanlara ilişkin ayrıntılı bilgi yukarıdaki şekilde bulunmaktadır.

5.1.1.1.3.1.6 SAVEDGAME TABLOSU

	Column Name	Data Type	Length	Allow Nulls
🔑	SaveID	int	4	
	GameID	smallint	2	
	UserID1	int	4	
	UserID2	int	4	
	SaveName	nvarchar	50	
	BoardLayout	nvarchar	500	
	GameStateFlags	nvarchar	100	✓
	GameLogicFlags	nvarchar	100	✓

5.1.1.6.1 SavedGame Tablosu

Bu tabloda, üyelerin kendi aralarında oynayıp, daha sonra tekrar oynamak üzere kaydettikleri oyunlara ilişkin bilgiler tutulmaktadır. Oynanmış oyunlara ilişkin nesnelerin değerleri gibi bilgiler bu tabloda tutulur. Tablodaki alanlara ilişkin ayrıntılı bilgi yukarıdaki şekilde bulunmaktadır.

5.1.1.1.3.1.7 SCORE TABLOSU

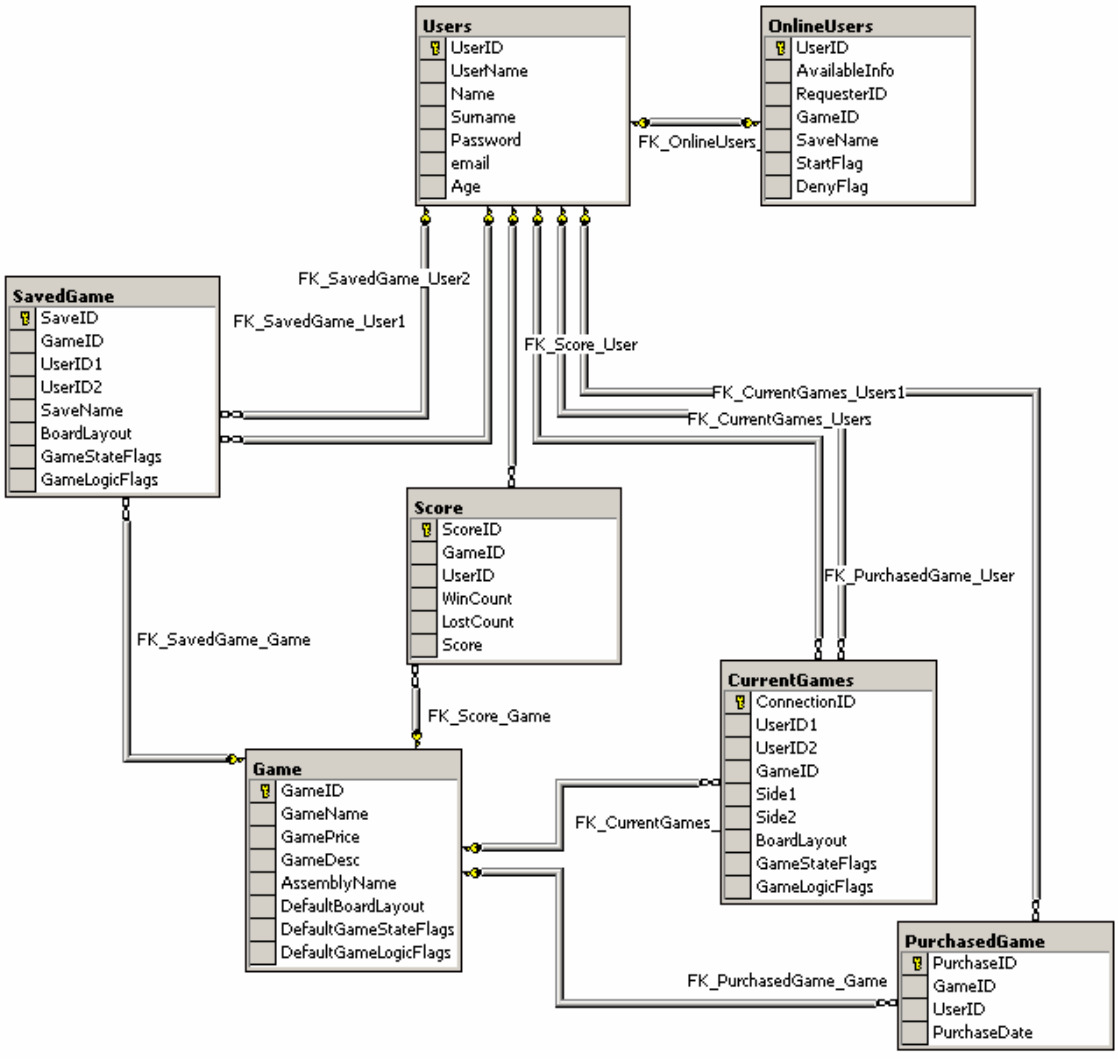
	Column Name	Data Type	Length	Allow Nulls
🔑	ScoreID	int	4	
	GameID	smallint	2	
	UserID	int	4	
	WinCount	int	4	✓
	LostCount	int	4	✓
	Score	int	4	✓

5.1.1.7.1 SavedGame Tablosu

Bu tabloda, GameNETwork sistemine üye olan kullanıcıların sistemde kazandıkları ve kaybettikleri oyun durumlarına göre puan durumları tutulmaktadır. Böylece, üyeler rakiplerini, puan durumlarını araştırarak seçebilir. Tablodaki alanlara ilişkin ayrıntılı bilgi yukarıdaki şekilde bulunmaktadır.

5.1.1.1.3.1.8 GAMENETWORK E-R DİYAGRAMI

Sistemde varolan tablolar arası ilişki şu şekildedir:



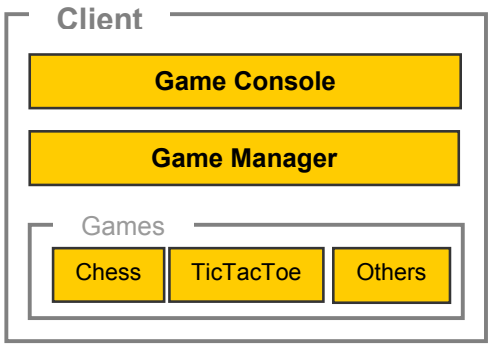
5.1.1.8.1 E-R Diyagramı

5.1.1.1.3.2 STORED PROSEDÜRLER

DataLogic Katmanından gelen istekleri karşılamak için, çok çeşitli stored prosedürler yazılmıştır. Bu stored prosedürler, istemci istekleri doğrultusunda, veritabanı üzerinde işlemler yaparlar. Örneğin, kullanıcıyı sisteme eklemek için AddUser stored proseduru kullanılmıştır.

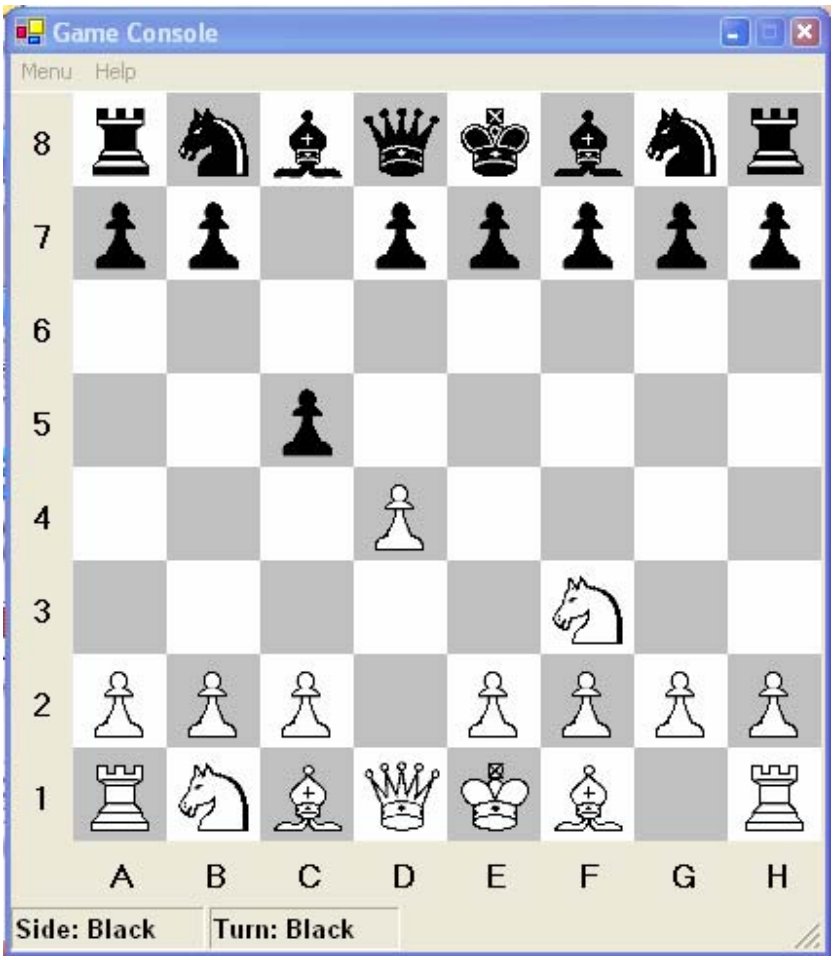
5.1.1.2. İSTEMCİ MİMARİSİ

Aşağıdaki şekilde istemci arabirimlerinin mimarisi gösterilmiştir:



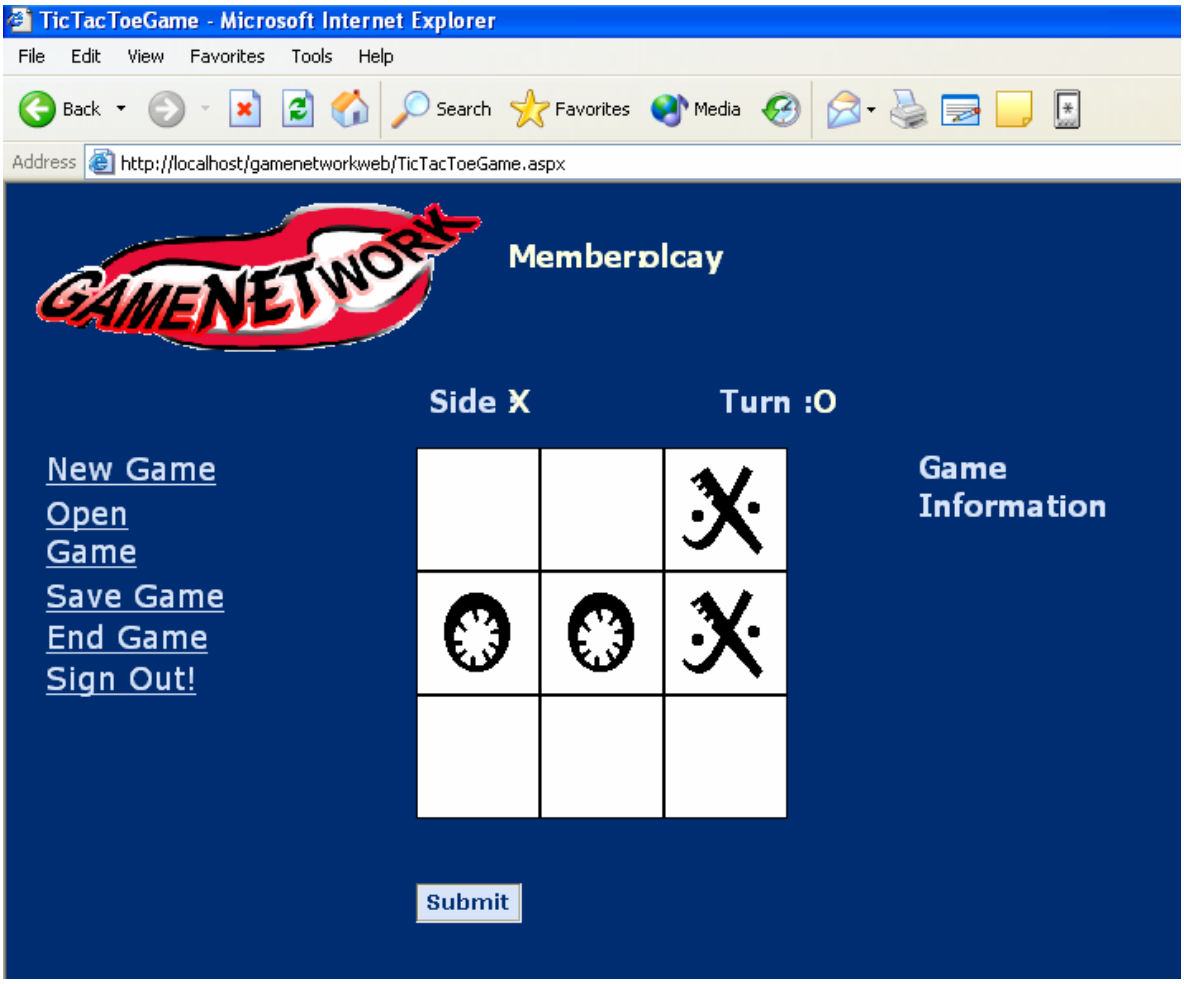
İstemci mimarisinin en üstünde Console katmanı yer alır. Kullanıcıya sunulan asıl arayüz bu katmandadır. Bu katmanla kullanıcının istekleri alınır ve kullanıcıya cevaplar sunulur. Bu katmanın altında komponentlerin işletilmesinden sorumlu olan Game Manager katmanı yer alır. Game manager kullanıcının isteği doğrultusunda ilgili komponenti çalışma-zamanında (run - time) konsola ekler ve çıkartır. Game Manager katmanının altında kullanıcının satın almış olduğu ve internetten download etmiş olduğu komponentler yer alır. Bu komponentlerin her biri birbirinden bağımsız olarak çalışırlar. Dolayısıyla birden fazla komponentin aynı anda çalışması mümkündür. Bu da projenin esnekliğini ve geliştirilebiliriğini artırmaktadır. Komponentlerin çalışma zamanında eklenip çıkarışabilmesinde .NET Reflection teknolojisinden yararlanılmıştır. Çalışma zamanında programın genişletilebilir olmasının vermiş olduğu özgürlük sayesinde GameNETwork'ün genişletilmesi için bir yandan da yeni komponentler yazılmaya devam edilebilir. Bu komponentlerin yazımı tamamlandığında, bu komponentlerde sistem üzerinde çalışabileceklerdir. Böylelikle teorik olarak sistemin

genişletilmesinde bir sınır yoktur. Aşağıda sistemin farklı platformlarda çalışmasını gösteren çeşitli ekran resimleri (screenshot) yer almaktadır.

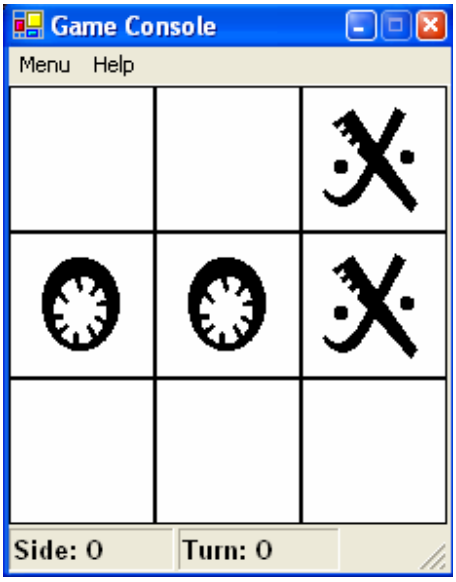


PC üzerinde satranç oynayan oyuncunun konsolundan bir görünüş.

PDA üzerinde satranç oynayan oyuncunun konsolundan bir görünüş.



Browser üzerinde tictactoe oynayan oyuncunun konsolundan bir görünüş.



PDA üzerinde tictactoe oynayan oyuncunun konsolundan bir görünüş.

WAP arayüzünden GameNETwork'e üye olan bir kullanıcının konsolundan bir görünüş.

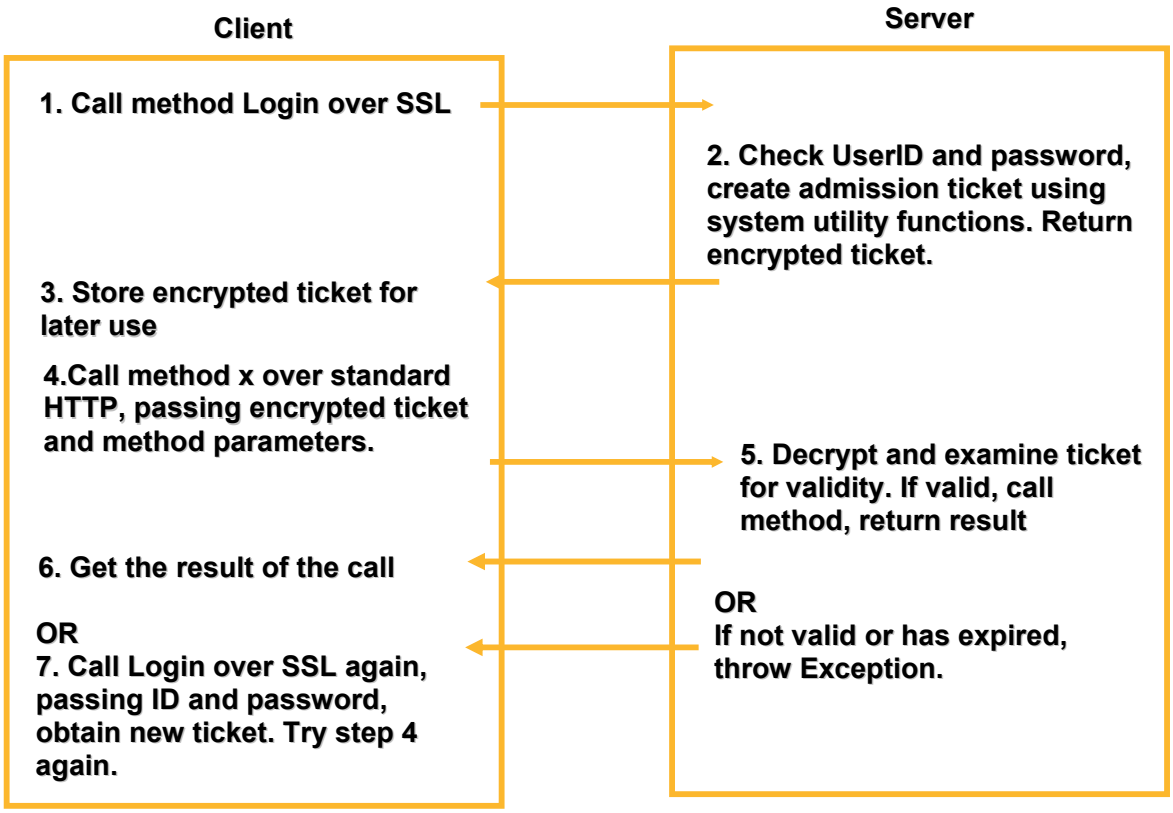
5.1.1.3. GÜVENLİK MEKANİZMASI (SECURİTY)

GameNETwork üyelik sistemine göre çalıştığından sistemin güvenliğinde sağlanmalıdır. Üye olmayan sistemi kullanamamalı ve üyelere ait özel bilgiler gizlenmelidir. Güvenlik mekanizmasını açıklamadan önce kullanılmakta olan güvenlik sistemlerinden bahs etmek yerinde olacaktır. Windows authentication ile gerçeklenen güvenlik active directory tabanlı olmaktadır. Yani active directory'de bulunan kullanıcı hesapları kontrol edilerek kullanıcı onaylanır (authentication). Bu yöntemin avantajı active directory gibi güvenli herkes tarafından kabul görmüş bir sistemin kullanılmasıdır. Bir diğer avantajı da kullanıcı hesaplarında tutulan izin (permission) bilgilerinin kullanılmasıdır. Fakat bu yöntemin en

büyük dezavantajı sadece windows platformlarında çalışmasıdır. Biz projeyi geliştirmeden önce koyduğumuz hedeflerden biri yazılımın bütün platformlarda çalışması idi. Dolayısıyla bu yöntemin kullanılması doğru olmayacaktır. Windows authentication'ın dışında basic ve custom authentication denen iki farklı yöntem daha vardır. Bu iki yöntemde de verilerin şifrelenmesinden ve diğer güvenlik unsurlarından yazılım geliştirici sorumludur. Şifrelemede kullanılan algoritmalar simetrik ve asimetrik algoritmalar olarak ikiye ayrılabilir. Simetrik algoritmalarda public key kullanılır. İletişimde bulunan iki durak da public key'i bilmektedir. Veriyi gönderecek olan public key'i kullanarak veriyi şifreler. Alıcı durak da public key'i kullanarak şifrelenmiş veriyi çözer. Asimetrik algoritmalarda ise hem public key hem de private key kullanılır. Alıcı durak göndericilere public key'ini duyurur. Göndericiler bu key'i kullanarak veriyi şifrelerler. Alıcı durak da private key'i kullanarak şifrelenmiş veriyi çözer. Private key'i sadece kendisi bildiği için simetrik algoritmalara göre daha güvenlidir. Asimetrik algoritmaların bir başka üstün tarafı ise kırılmalarının zor olmasıdır. Kırılmaları mümkündür. Fakat kırmak için geçecek süre en iyi bilgisayarda dahi çok uzun zaman alacağından güvenlik otoritelerince asimetirk algoritmalar güvenli olaral kabul edilirler. Asimetrik algoritmaların dezavantajı ise yoğun matematiksel işlemler yaptığından performansı düşürmeleridirler. Genel olarak custom authentication yöntemi kullanılır. Bu yöntemde simetirk ve asimetrik algoritmalar birlikte çalışırlar. Veri iletişiminde çok özel veriler hariç simetirk algoritmalar kullanılır. Özel verilerin gönderilmesinde, örneğin public key2in güncelleştirilmesi durumunda, asimetirk alogoritmalar kullanılır.

Biz projemizde bu yöntemleri bir adım daha ileri götürerek hem yeterli güvenliği sağladık hem de performans düşmesini engelledik. Geliştirmiş olduğumuz güvenli mekanizması şu şekilde çalışmaktadır:
Kullanıcı login olurken parola bilgisi gönderildiği için bu bilgi SSL (Secure Socket Layer) ile gönderilmiştir. SSL yöntemi direk olarak web sunucuları tarafından çalıştırlan, güvenli herkes arafından kabul görmüş, 128-bit şifreleme yapan bir şifreleme yöntemidir. Sertifikalar ile çalışmaktadır. Biz ticari bir ürün geliştirmediğimiz için sertifikayı verisign gibi bir şirketten almak yerine windows'un sağlamış olduğu ceritification authority serivisini kullandık. SSL güvenli bir yöntemdir fakat uygulamaların performansını ciddi bir oranda düşürmektedir. Bu yüzden her veri aktarımında SSL'i kullanmak doğru olmaz. Biz de bu sebebten dolayı SSL'i

sadece login ve sign up işlemlerinde kullandık. Kullanıcı login olduktan sonra cookie'ler ile çalışan bir şifreleme algoritması kullanılarak kulanıcının isimi şifrelenmiş ve şifrelenmiş veri istemciye oturum anahtarı (session key) olarak gönderilmiştir. İstemci bu anahtarı kendisinde kayıtlı olarak tutar. Dolayısıyla her seferinde yeniden key üretmek için zaman harcanılmayacaktır ve key'in çözülmesi işlemi sadece sunucu tarafında yapılacaktır. Aynı zamanda sunucu sadece oturum anahtarını deşifre edecek diğer verileri deşifre etmek için zaman harcamayacaktır. Oturum anahtarı bilgisi web servislerinin semantiği ile ilişkili olmadığından bu veri SOAP başlığı içerisinde sunucuya gönderilmektedir. Sunucu SOAP balşığındaki anahtarı deşifre ederek kullanıcının sisteme online olup olmadığını kontrol eder. Online değilse güvenlik hatası (Security Exception) üretir.



Şifreleme algoritmasının cookieler ile çalışmasından dolayı hat dinlense bile web servisleri başka bir kaynaktan çağrılacağı için deşifreleme aşamasında kötü veri (bad data – crytographic exception) hatası oluşacaktır. Tasarlanımş olan güvenlik mekanizması hem yeterli güvenliği sağlamaktadır hem de iyi performans göstermektedir.

5.1.1.4. HATA KONTROL MEKANİZMASI (EXCEPTİON HANDLİNG)

Exception'lar program çalışması sırasında meydana gelen hatalrdır. Bunların handle edilmesi için diğer OOP dillerinde de bulunan try – catch – throw deyimleri kullanılır. İleride de anlatılacağı üzere ölçeklenebiliriliği artırmak için durum bilgilerini (state management) veri tabanında tutmamız gerekti. Bu da veri tabanın aktivitesini artırmaktadır. Veri tabanının bütünlüğünü ve tutarlılığını sağlamak için kendi exception sınıfımızı (class GameNETworkException) yazdık. Veri tabanında meydana gelebilecek bir hata Data Logic katmanında yakalnmaktadır ve bu hata üst katmana GameNETworkException nesnesi olarak throw edilmektedir. Üst katman hangi kullanıcının veya kullanıcıların veri tabanınına erişitiğini bildiğinden catch bloğunda bu hatayı yakalayarak ilgili kullanıcının sadece state tablolarında yapmış olduğu kayıtları siler ve kullanıcıyı logout eder. Böylelikle veri tabanının tutarlılığı sağlanmış olur.

5.1.1.5. DURUM YÖNETİMİ (STATE MANAGEMENT)

Sistemimizde hizmet verebilecek komponentler arasında durum bilgilerine ihtiyaç duyabilecek olanlar olabilir. Örneğin satranç ve tictactoe oyunları bu bilgiye ihtiyaç duymaktadır. Bu bilgiler iki türlü saklanabilir; bellekte veya sabit diskte. Hash Table ile veyahut ASP.NET'in state management için sağlamış olduğu tablolar kullanılarak durum bilgileri bellekte tutulabilir. Bu da oyunlar gibi sürekli etkileşim isteyen komponentler de iyi performans sağlayacaktır. Fakat biz bir internet uygulaması geliştirdiğimiz için ölçeklenebilirlik performansdan daha önemlidir. Sisteme binlerce kullanıcı bağlandığında bellek yeterli gelmeyecek ve sistem çökecektir. Bunun için durum bilgilerini sabit diskte tutmak daha doğru olur. Veri organizasyonu sağladığı için biz de durum bilgilerini veri tabanındaki tablolarda tuttuk.

6. SONUÇLAR VE ÖNERİLER

Kodları, metodları verilen ve adım adım yöntemi anlatılan bu projede Visual Studio .NET etkin bir şekilde kullanarak, katmanlı yapıda ve dağıtık bir uygulama geliştirilmiştir. Uygulama çeşitli koşullarda test edilmiş ve başarılı olmuştur. Projenin en önemli özelliği olan farklı platformlara hizmet verme kabiliyeti, genişleyebilir, ölçeklenebilir ve güvenli olması projenin en önemli özellikleridir. Projenin bir diğer çarpıcı özelliği ise, kurulmuş olan altyapı ile teorik olarak sonsuz sayıda internet uygulamasının sistem üzerinde host edebilebilir olmasıdır. Diğer bir özellik ise projenin OOP programlama tekniği ile yazılmış olmasdır. Bundan dolayı aynı kodun tekrar kullanımı, kod bütünlüğü, kod okunabilirliği gibi yazılım geliştiriciye yönelik özellikler de projenin diğer bir üstünlüğüdür.

Projenin altyapı olarak, geliştirilmeye müsait olduğundan, bir çok özellik projeye eklenebilir. Bizce eklenebilecek en önemli özellik JAVA platformlarını desteklemek için bir JAVA arabiriminin projeye eklenmesidir. Bunun dışında projenin güvenliğini artırmak için evidence-based security özelliği de projeye eklenebilir.

7. KAYNAKÇA

[1] Robinson, S. ,Professional C#, Wrox Press, ISBN : 1-861004-99-0,2001

[2] Morris, C., .NET Compact Framework, Wrox Press, ISBN: 1-861007-00-0,2001

[3] Sutton, M., Microsoft .NET Compact Framework, MS Press, ISBN: 0-7356-1725-2, 2002

[4] Sharp, J., Microsoft Visual C# .NET, Step by Step, MS Press, ISBN: 0-7356-1289-7, 2002

[5] Gibson, M., Microsoft Visual Basic .NET, Step by Step, MS Press, ISBN: 0-7356-1374-5, 2002

[6] Reilly, D., Designing Microsoft ASP.NET Applications, MS Press, ISBN: 0-7356-1348-6, 2002

[7] Riordam, R., Microsoft ADO .NET Step by Step, MS Press, ISBN: 0-7356-1236-6,2001

[8] Haneshi, M., Microsoft .NET XML Web Services, Step by Step, MS Press, ISBN: 0-7356-1720-1,2002

[9] Pillard, K., Programming a Microsoft SQL Server 2000 Database, MSDN Training,2001

[10] Pillard, K., Administering a Microsoft SQL Server 2000 Database, MSDN Training,2001

[11] Richmond, S., Programming with the Microsoft .NET Framework, MSDN Training,2001

[12] Kitten, B., Developing Microsoft .NET Applications for Windows(Visual C# .NET), MSDN Training,2001

[13] MSDN Library for Visual Studio .NET 2003,2003

[14] Turtschi, A., C#.NET Web Developers Guide, Syngress Press, ISBN: 1-928994-50-4,2002

[15] Mesbah, A., ASP.NET Web Developers Guide, Syngress Press, ISBN: 1-928994-51-2,2002