T.C.
HALİÇ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
YÖNETİM BİLİŞİM SİSTEMLERİ PROGRAMI

AİLELERE YÖNELİK BİR SOSYAL BİLİŞİM AĞININ OLUŞTURULMASI VE GELİŞTİRİLMESİ

YÜKSEK LİSANS TEZİ

Hazırlayan
Hasan KAYA

Danışman
Yrd. Doç. Dr. Ülviye HACIZADE

İstanbul – 2014

T.C.

HALİÇ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ MÜDÜRLÜĞÜNE

Bilgisayar Mühendisliği Anabilim Dalı Yönetim Bilişim Sistemleri Programı Tezli Yüksek Lisans öğrencisi **Hasan KAYA** tarafından hazırlanan **"Ailelere Yönelik Bir Sosyal Bilişim Ağının Oluşturulması ve Geliştirilmesi"** adlı bu çalışma jürimizce Yüksek Lisans Tezi olarak kabul edilmiştir.

Sınav Tarihi : 26.12.2014

(Jüri Üyesinin Ünvanı , Adı , Soyadı ve Kurumu) :	İmzası :
Jüri Üyesi: Yrd.Doç.Dr.Ulviye HACIZADE Dan.-HAL.Üniv. Bilgisayar Müh. ABD Öğr.Üyesi	.....
Jüri Üyesi : Prof.Dr.Muhammet KÖKSAL HAL.Üniv. Bilgisayar Müh. ABD Öğr.Üyesi	.....
Jüri Üyesi : Yrd.Doç.Dr.Soner ÖZGÜNEL HAL.Üniv. Elektrik-Elektronik Müh.ABD Öğr.Üyesi	.....
Jüri Üyesi : Prof.Dr.Mübariz EMİNLİ HAL.Üniv. Bilgisayar Müh. ABD Öğr.Üyesi (Yedek)	.....
Jüri Üyesi : Yrd.Doç.Dr.U.Hulusi İNAN HAL.Üniv. Endüstri Müh.ABD Öğr.Üyesi (Yedek)	.....

ÖNSÖZ

Yüksek lisans eğitimim ve daha sonraki tez çalışmam da bana desteğini esirgemeyen tez danışmanım Yrd. Doç. Dr. Ulviye Hacızade'ye, bölüm hocalarım Bilgisayar Mühendisliği Anabilim Dalı Başkanı Prof. Dr. Muhammet Köksal'a ve Prof. Dr. Mübariz Eminli'ye teşekkür ederim.

Yüksek lisans eğitim sürecinde birlikte aynı bölümde eğitime devam eden arkadaşlarım Canan Keskin, Elif Özge Topal, Aydın Ertürk, Nurcan Özdemir ve Mesut Aslan'a teşekkür ederim.

Ayrıca bu süreçte desteklerini her an arkamda hissettiğim aileme de teşekkür etmek isterim.

İstanbul, 2014 Hasan KAYA

İÇİNDEKİLER

Sayfa No.

KISALTMALAR

FSN : Aile Sosyak Ağı (Family Social Network)

SQL : Yapılandırılmış Sorgu Dili (Structured Query Language)

API : Uygulama Programlama Arayüzü (Application Programming Interface)

UML : Birleşik Modelleme Dili (Unified Modelling Language)

MVC : Model View Controller

URL : Tekdüzen Kaynak Bulucu (Uniform resource locator)

REST : Temsili Durum Transferi (Representational State Transfer)

ANSI : Amerikan Standartlar Enstitüsü (American National Standards Institute)

OO : Nesne Tabanlı (Object Oriented)

TABLO LİSTESİ

Sayfa No.

ŞEKİL LİSTESİ

Sayfa No.

GENEL BİLGİLER

Adı ve Soyadı : Hasan KAYA

Anabilim Dalı : Bilgisayar Mühendisliği

Programı : Yönetim Bilişim Sistemleri

Tez Danışmanı : Yrd. Doç. Dr. Ülviye HACIZADE

Tez Türü ve Tarihi : Yüksek Lisans- Haziran 2014

ÖZET

AİLELERE YÖNELİK BİR SOSYAL BİLİŞİM AĞININ OLUŞTURULMASI VE GELİŞTİRİLMESİ

Tez konusu, kısaca aileyi temel alan ve yine ailelere yönelik olarak kullanılacak bir sosyal ağ oluşturmaktır. Bu sosyal ağda kullanıcıların tüm verilerinin saklanması, bu verilerin istenilen ölçüde ve güvenli olarak paylaşılması hedeflenmektedir.

Aileler arası etkileşimin de sağlanacağı bu ağda, aile tanımlama bilgileri, aile bireylerinin bilgileri saklanması, albümlerin oluşturulması ve paylaşılması dışında ortak etkinliklerin oluşturularak aileler arası etkileşimin artırılması da amaçlanmıştır.

Ayrıca sitenin kolay anlaşılır ve kullanılabilir olması hedeflenmiş, işlemler arası geçişler kolay ve çabuk olacak şekilde tasarlanmıştır. Kullanımı kolay ve temel sosyal gereksinim fonksiyonlarını içeren bir site hedeflenmiştir.

Diğer sosyal ağlardan temel farkı aile odaklı olması ve aileler arası ilişkileri temel almasıdır. Bu tez kapsamında mevcut sosyal ağlar incelenmiş, aileler arası paylaşım temelleri üzerine araştırmalar yapılmıştır. Bu inceleme ve araştırma sonucunda en temek fonksiyonları içeren bir sosyal ağ oluşturulmuştur. Geliştirmeye açık bir sosyal ağdır.

Burada toplanılacak verilecek ayrıca aileler arası ilişkilerin incelenmesi ve toplumsal paylaşım ve alışkanlıklar hakkında araştırma yapılmasına veri desteği sağlayacak niteliktedir.

Kullanılan teknolojilerin güncel olmasına dikkat edilmiş, genişleyebilir bir yapı oluşturulmuştur. Bu sebeple geliştirme aracı olarak Visual Studio 2013, geliştirme dili olarak c# ve veri tabanı yönetim aracı olarak ta MSSQL Server 2012 kullanılmıştır.

Anahtar Kelimeler: Aile, Sosyal ağ, Etkileşim, Veri toplama

GENERAL INFORMATION

Name and Surname : Hasan KAYA.

Fied : Computer Engineering

Program : Management Information Systems

Supervisor : Assist. Prof. Dr. . Ülviye HACIZADE

Degree Award and Date : Master - June 2014

ABSTRACT

GENERATING AND IMPROVING A FAMILY ORIENTED SOCIAL IT NETWORK

The subject of the thesis is establishing a social network which is briefly based on family and will be used focusing on to families. At this social network, it is targeted to store all the data and share these data at requested volume safely.

While interaction between families is to be established by this network; increasing interaction between families through forming common activities is aimed other than storing family members' data, creating and sharing albums.

Additionally, it is targeted to make the site easily understood and used while transitions between processes are designed to be easy and fast. A site that is easy to use and includes basic social requirements functions is targeted.

The main difference of the other social networks is being based on families and relations between families. The present social networks are examined and research based on sharing basics between families within the concept of this thesis are performed. As a result of this examine and research, a social network consisting the most basic functions is established. It is a social network open to development.

All the data to be stored here is also qualified to supply data for reseaches about relations between families, societal sharing and habits.

Paid attention to choose current technologies to use and thus formed an expandable structure. With this purpose, Visual Studio 2013 as improvement tool, c# as improvement language and MSSQL Server 2012 as database management tool have been used.

Key Words: Family, Social network, Interaction, Data collection

1. GİRİŞ

Günümüzün önemli kullanım eğilimlerinden bir tanesi de sosyal ağlardır. İnsanlar yaptıklarını günlük hayata dair bilgilerini, düşüncelerini kendilerinin kontrol edebildikleri sanal ağlar üzerinden paylaşmaktadırlar. Paylaşımlarda genel yaklaşım ve içerik yönetimi hizmeti veren kurumların odak noktasında kişisel paylaşımlar yer almaktadır. Bu tezde amaç ise aileyi merkez kabul eden, paylaşım ve kullanılan araçları bu amaçla üretilmiş bir sosyal ağ oluşturmak ve oluşturulan ağın geliştirilmesi için alt yapıyı hazırlamaktır.

Bu sosyal ağın kullanıcıları, kendi ailelerin hakkında bilgiler girebilecekler, paylaşımlarda bulunabilecekler ve bu paylaşımları istedikleri seviyede yetkilendirebileceklerdir. Bu şekilde başka ailelerle iletişime geçebilecekler ve bilgi paylaşımının yanında etkileşimde de bulunabileceklerdir.

Tez konusu seçim aşamasında Ulusal tez merkezindeki kabul edilmiş Bilgisayar Mühendisliği ana bilim dalı ve Yönetim Bilişim Sistemleri programında hazırlanan tezler, sosyal paylaşım siteleri, Facebook(2014), Twitter(2014), Linkedin(2014), MyFamily(2014), Efamily(2014) incelenmiştir. Ayrıca Sosyal Ağların Yapısı ve İçerikleri ile Sosyal Ağ Analizi isimli kitaplar araştırmaya dâhil edilmiştir. Daha sonra hazırlanacak sosyal ağ için yazılım geliştirme sürecinin nasıl olması gerektiği anlatılmıştır. Geliştirme sürecinin her adımı için çalışma yapılmıştır. Kullanılacak geliştirme ortamı ve araçlar belirlenmiştir.

Tezde öncelikler yazılım geliştirme sürecinde kullanılacak araçlar yazılım geliştirme teknolojileri ve modelleme dili başlıkları altında detaylı olarak anlatılmıştır. Yazılım geliştirme teknolojileri, kullanılacak programlama dili ve geliştirme aracı ile veri tabanı yönetim sistemi detaylandırılmıştır.

Sistemin modelleme sürecinde kullanılacak olarak modelleme dili ve bu modelleme dilinin tezin yazımında kullanılan diyagramları detaylı bir şekilde tüm özellikleri tek tek irdelenerek anlatılmıştır.

Yazılım geliştirme sürecinde kullanılan analiz, tasarım, geliştirme ve test başlıklarından oluşan genel geliştirme adımlarından bahsedilmiştir. Her adım ayrı başlıklar altında detaylı olarak anlatılmıştır.

Tezin konusu olan Ailelere Yönelik Sosyal Ağ Geliştirme süreci bölümünde ise amaç ve kapsam belirtilmiş, gereksinim listesi oluşturulmuş, sistemdeki kullanıcılar belirlenmiş, kullanıcılara ait işlevler olay listeleri oluşturularak anlatılmış, olay listeleri üzerinde kullanım senaryoları oluşturulmuş, veri yapıları belirlenmiş ve tablolar oluşturulmuştur. Son olarak kullanıcı ara yüzleri üzerinden ekranların kullanım şekli ve detayları açıklamalı olarak anlatılmıştır.

Geliştirme teknolojilerinin seçiminde hali hazırda yaygın olarak kullanılan geliştirme teknolojileri incelenmiş web sitesi ve benzeri çevrimiçi servis veren yazılım geliştiren firmaların ne tür teknolojiler kullanıldığına araştırılmıştır.

Hem ulaşılabilir kaynak desteğinin fazla olması, hem güncel teknolojileri içinde barındırması (Bulut desteği, mobil geliştirme desteği, Developer team server vs) ve geliştirme sürecine sağladığı faydalar sebebi ile geliştirme aracı olarak Visual C# programlama dili ve ASP.NET MVC 4 Web Application ayrıca veri tabanı olarak ise MS SQL Server 2012 sürümünün kullanımına karar verilmiştir

2. YAZILIMIN GELİŞTİRİLMESİNDE KULLANILACAK GEREÇLER

2.1. Kullanılacak Teknolojiler

2.1.1. Programlama Dili ASP.NET MVC Framework

ASP.NET MVC Microsoft tarafından geliştirilen ve hali hazırda geliştirilmeye devam eden, en son sürümü 5.0 olan, temel yaklaşımı geliştirme sürecinin alt parçalara bölerek geliştirme sürecinin kolay ve disiplinli olmasını sağlamak amacı ile oluşturulan bir programlama dilidir. ASP.NET MVC farklı bir programlama dili değildir, sadece yapı olarak klasik ASP.NET'ten farklı ve daha az karmaşıktır.(Microsoft. 2014)

1979 'da Trygve Reenskaug tarafından ortaya çıkarıldıktan sonra Xerox Parc'ta geliştirilmiş ve yazılım mühendisliğinde kullanılan bir mimari desendir. İlk tanımlaması Thing-Model-View-Controller olarak yapılmıştır, İlk kullanım kılavuzu Applications Programming in Smalltalk-80: How to use Model-View-Controller olarak bilinir(vikipedia, 2014).

MVC'nin açılımı Model View Controller şeklindedir. Üç ana bölümden oluşur ve katmanlı bir yapıya sahiptir. Bu katmanlı yapı, yapılacak olan geliştirmeleri birbirinden ayırarak, çok daha kolay ve hızlı bir geliştirme ortamı sağlar. Bu bölümler; Model katmanı, View katmanı ve Controller katmanlarıdır. Bu üç bölümü şu şekilde açabiliriz;

Model; ASP.NET MVC için model nesneyi ifade eder. Bu nesne veri tabanını ve iş kurallarını temsil eder. Bunlar ise uygulamanın üzerinde çalışacağı yapıyı oluşturur. Genel olarak bu bir veri tabanıdır. Veri modelleri ve veri nesneleri burada tanımlanır ve geliştirilir.

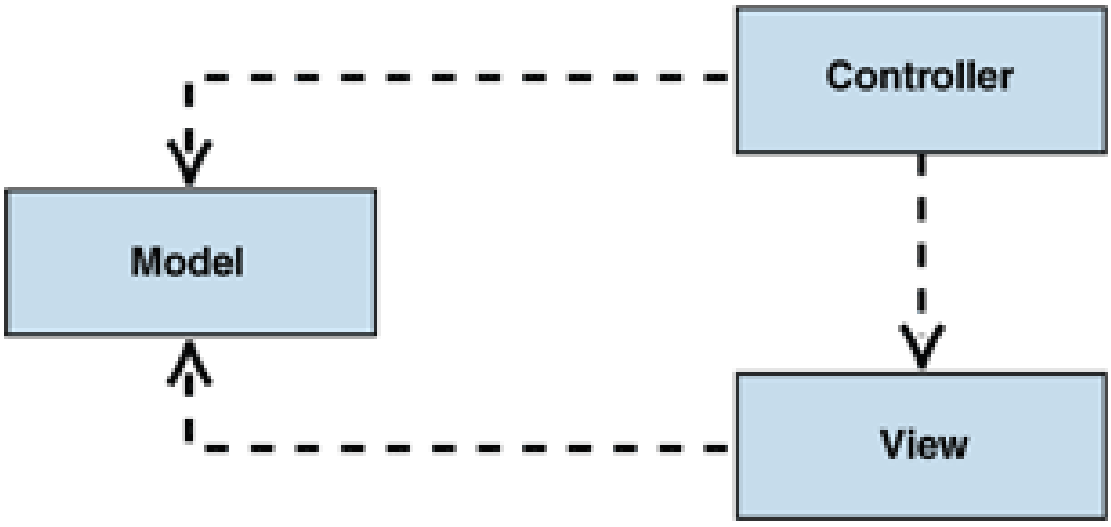
Controller; Model üzerindeki verilerin yönlendirilmesi, işlenmesi ve düzenlenmesi bu katmada yapılır. İş kuralları burada belirlenir ve Model katmanı ile View katmanı arasındaki iletişim burada kurulur ve kontrol edilir.

View; sonuç verilerinin görüntülediği verilerin girildiği hem giriş hem de çıkış katmanıdır. Verilerin görüntüleme araçları ile birleştirilerek, html çıktı dosyalarının

hazırlandığı ve sunulduğu bölümdür. Bunun dışında sitenin tasarımına ait verilerde bu katmanda yer alır.

ASP.NET MVC Framework, "interface"ler aracılığıyla yukarıda bahsedilen üç rolü destekler. View kısmı Web Formlarını destekler. Ama istenirse başka görünüm motorların da uyarlanması mümkündür. Sayfalar etkileşiminde URL Routing yapısı kullanır. Böylece REST uyumlu URL'ler tanımlanabilir. ASP.NET MVC Framework bu URL'lerin ilgili Controller metotlarıyla eşlenmesi ve bu URL'lerin oluşturulması için hazır bir altyapı sunar.

Şekil 2.1'de MVC katmanları arasında ilişkisel yapı görülmektedir.



Şekil 2.1. MVC Katmanlar arası mimari

MVC mimarisinin avantajları

- Kod okunabilirliğinin yüksek olması, anlaşılır bir yapı oluşturması
- Eski mimari yapılara göre daha hızlı proje geliştirme imkânı sağlaması
- Daha hızlı ve performanslı geliştirme için altyapı sağlaması
- Web projesi şeklinde kullanılmasını sağlaması
- UrlRouting ile seo dostu Url'ler oluşturmayı daha kolaylaştırması.
- MVC'nin Ajax kütüphanesi (System.Web.Mvc.Ajax) sayesinde json kullanımı çok kolay hale getirmesi.
- Takım çalışmasına uygun altyapı sağlaması.
- Gelişmiş hata ayıklama ve test imkânı sağlaması.

- /controllerName/functionName şeklinde sayfalara ulaşma imkânı sağlaması.

2.1.2.Microsoft SQL Server

Verilerin saklaması, işlenebilmesi ve yönetimi için kullanılan Microsoft tarafından geliştirilmiş bir DBMS aracıdır. SQL dilini destekleyen bir veri yönetim aracıdır. İşlemleri tamamı SQL konutları veya grafik ara yüz kullanılarak yapılabilir. Kullanıldığı işletim sisteminin API (Application Programming Interface) fonksiyonlarını, programcının yardımına ihtiyaç olmaksızın kullanır ve veri yönetim süreci kolay ve performanslı olarak yönetmenizi sağlar.

En yaygın kullanılan veri tabanı yönetim sistemleri:

- Microsoft SQL Server
- Oracle
- My-Sql
- Microsoft Access
- Informix

Birçok programlama dilinde olduğu gibi, SQL de standart hale getirilmiştir. ANSI tarafından belirlenen standartlar bütününe ANSI SQL denir, DBMS çözümlerinin çoğu tarafından desteklenir. Ayrıca geliştiriciler tarafından da kendi uygulamalarına özel fonksiyonlarda ilave edilmektedir.

Günümüzde veri tabanı (database), kullanımı son derece yaygındır. İnternet üzerinden yapılan her türlü arama, araştırma, veri girişleri ve benzeri uygulamalar veri tabanı işlemleri sayesinde yapılabilmektedir. Geliştirilen tüm sosyal ağlar, arama motorları ve benzeri büyük veri işleyen yapılar çok güçlü veri tabanı işleme çözümleri sayesinde mümkün olmaktadır.

Veri tabanı, işlenmek istenen verilerin sakladığı erişim şekline göre hazırlandığı ve yönetildiği bir veri depolama ve yönetim programıdır.

Veriler veri tabanında bilgileri tablo adı verilen alt birimlerde saklarlar. Bir DBMS aracında birden fazla veri tabanı ve her veri tabanında yüzlerce tablo olabilir. Ancak bunların belirli bir şekilde ayrılması gerekir. Veri tabanı ve tabloların yapıları tasarım aşamasında belirlenir. Tasarım sırasında sadece verilerin saklanması değil onlara ulaşmak için en uygun yapıda göz önünde bulundurularak işlemler

gerçekleştirilir. Veriler farklı tablolarda tutulsa dahi birbiri ile ilişkilendirilebilirler. Bir veri tabanında aynı isimde iki tablo bulunamaz. Tabi ki farklı veri tabanlarında aynı isimde tablolar bulunabilir. Veri tabanına ait tabloların özelliklerini ve yerleşim şekillerini içeren bilgiye şema (schema) denir.

Kolonlar ve Veri Türleri (Columns and Datatypes):Veri tabanının tablolardan oluşur. Tablolar da kolonlardan oluşur. Kolonlar, her birinde aynı türden veri saklayan birimlerdir. Örneğin kişi bilgilerinin verilerini saklandığı bir tabloda, bir kolon tc kimlik numarasını, bir kolon adını ve diğer bir kolon da soyadını tutar. Yani her bir farklı bilgi, bir kolonda saklanır.

Verileri ayrı ayrı kolonlarda tutulması verimli bir yapı için çok önemlidir. Örneğin aynı isime sahip kişilere ulaşmak istendiğinde, eğer isim ve soy isim bilgisi aynı kolonda tutulmuşsa işlem karmaşıklaşacaktır ve performans daha düşük olacaktır. Ancak isim ve soy isim ayrı ayrı kolonlarda tutulmuşsa, işlem oldukça kolay ve hızlı olacaktır.

Kolonlarda saklanacak verilerin özellikleri farklılık gösterir, bu farklılıklar saklanacak verinin veri türü bilgisi ile belirlenir. Bu veri türü sayesinde o kolonda saklanacak bilgi sınırlanır. Örneğin sayısal ifadelerin yer alacağı bir kolona karakter girilmesinin yasaklanması gibi.

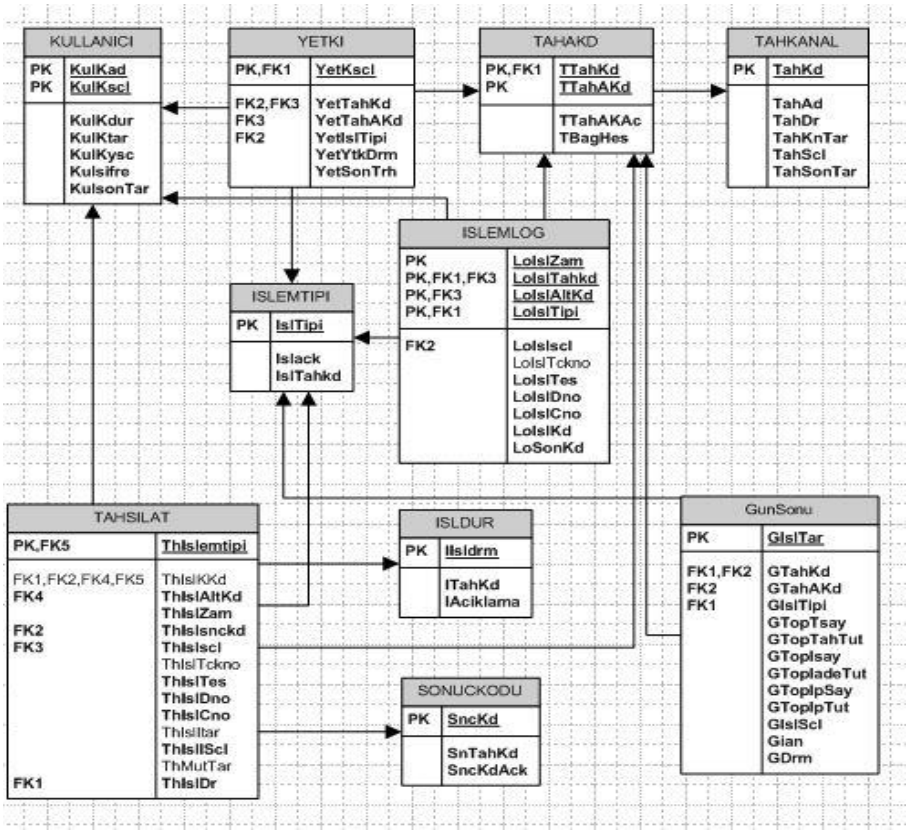
Veri tabanlarında saklanacak bilgilerin türlerini belirten veri türleri, farklı veri tabanı yönetim sistemlerinde farklı isimlere sahip olabilirler. İşte bu farklılık, veri tabanı uyumu konusunda uyumsuzluk oluşturan en temel konudur. Aslında temel veri türleri aynı isimle kullanılsa da, özel veri türleri oldukça farklıdır. Özellikle büyük verilerin saklanması ile ilgili farklı üreticiler farklı çözümler üretmekte bu sebepler veri tiplerinde farklılıklar oluşmaktadır.

Satırlar (Rows):Bir tablodaki bilgiler satırlarda saklanır. Her bir kayıt kendi sırasında tutulur. Bir satır, bir kayıt (record) demektir.

Tekil Anahtarlar (Primary Keys):Her bir tabloda, tablonun özelliğini taşıyan özel bir veya birden fazla kolon bulundurmalıdır. Örneğin öğrenci tablosunda, öğrenci numarasını tutan kolon bu tablodaki tekil bilgiyi tutar. O zaman tasarım sırasında öğrenci numarasını tutan kolon, tekil anahtar olarak belirtilmelidir. Tekil anahtar kullanarak aynı kolonda farklı satırlarda aynı bilginin girilmesi engellenmiş olur. Ve bu kolona veri girilmemesi gibi bir durum söz konusu olamaz. (NULL değer kabul edilmez.) Tekil anahtar olarak belirlenen kolonların değiştirilmemesi gerekir.

Farklı DBMS araçlarında kullanılan özelleştirilmiş SQL dili çözümleri bulunmaktadır. Microsoft tarafından geliştirilen MSSQL aracında T-SQL, Oracle firması tarafında geliştirilen araçta ise PL/SQL olarak adlandırılmakta ve geliştiricilere ait özel fonksiyonları ve yazılım terminolojisini desteklemektedirler. Bu araçlarda aynı zamanda ANSI SQL'de desteklenir.

Şekil 2.2'de veri tabanı programlarında kullanılan tabloların grafik görüntüsü ve tablolar arası ilişkilerin örnek bir şeması gösterilmektedir.



Şekil 2.2. Örnek Veri Tabanı İlişkileri Şeması

2.2 Kullanılacak Modelleme Dili

2.2.1 UML Modelleme Dili

UML nesneye dayalı programlama dilleri için uygun bir modelleme dilidir. Problemi parçalara ayırır ve bu parçalar arasında ilişki kurmak için UML kullanır.

UML ile hazırlanmış bir yazılım hem daha az maliyetli hem daha etkili ve uzun ömürlü olur. UML ile dokümantasyonu yapılmış bir programın sonradan düzenlenmesi daha kolay olur.

UML in faydaları:

- Programı kodlamaya başlamadan önce geniş bir analiz ve tasarım yapılmış olacağından kodlama işlemi daha kolay olur. Çünkü programdan ne beklediğimizi ve programlama ile neler yapacağımızı UML kullanarak profesyonel bir şekilde belirleriz.
- Kullanılan tekrar kod sayısı ayırt edilebilir bu sayede verim sağlanır.
- Mantıksal hataların minimum seviyeye düşürülmesini sağlar. Bütün sistem tasarlandığı için oluşabilecek hataların düzeltilmesi de daha kolaydır.
- Geliştirme maliyetinin düşmesini sağlar.
- UML diyagramları ile yazılım tamamını görebileceğimiz için verimli bellek kullanımı sağlanabilir.
- Karmaşık sistemlerde değişiklik yapmayı kolaylaştırır.
- UML ile dokümantasyonu yapılmış kodları düzenlemek daha az zaman alacaktır.
- UML diyagramlarını kullanan yazılımcılar aynı dili konuşacaklarından kolay iletişim sağlanır. Ayrıca müşteriler ve teknik sorumlular diyagramlar üzerinden kolaylıkla iletişim kurabilirler.

2.2.2 UML Diyagramları

Unified Modelling Language (UML) yazılım mimarisinin oluşturma sürecinin en önemli safhalarından biridir. Kurgulan sistemin tanımlanması, görsel yapının oluşturulması ve bunların dokümantasyonunu yapılabilmesi için oluşturulmuş bir standartlar dilidir. UML, büyük ölçekli ve karmaşık sistemlerin modellerinin oluşturulmasında başarısı ispatlanmış deneyimler bütününden oluşmuştur. OO yazılım geliştirme sürecinin önemli ve temel bir parçasıdır. UML'de diyagramlar yapısal ve davranışsal olmak üzere iki gruba ayrılır. Yapısal diyagramlar sistemin organizasyonunu yani genel yapısını vurgulanır, davranışsal diyagramlar ise sistemin işleyiş dinamiğine vurgu yapar. 9 temel eleman kullanılır. Bunlar:

- Use Case Diyagramları
- Class (Sınıf) Diyagramları
- Activity (Etkinlik) Diyagramları
- Sequence (Dizge) Diyagramları
- State (Durum) Diyagramları

- Object (Nesne) Diyagramları
- Collaboration (İşbirliği) Diyagramları
- Component Diyagramları
- Deployment Diyagramlar

Sıklıkla kullanılan 3 diyagramın detayı aşağıda verilmiştir.

2.2.2.1. Use Case Diyagramları

Sistemi kullanacak olan bileşenlerin sistemle ilişkili durumlarını gösteren diyagram türleridir. Kullanıcı diyagramlarında, özellik ve işlevleri aktör ile ifade edilen nesneler kullanır. Aktörler insan, araç, web servis veya başka bir yazılım veya yazılım bir parçası olabilir. Aktör olarak kurgulanan sistem veya bir sistemin alt herhangi bir parçası konumlandırılamaz. Use case diyagramları, kurgulana sistem üzerinde aktörün yapılmak istenen işin sonucuna hangi yöntem veya yöntemlerle ulaşabileceğini açıklar. Buradaki en önemli nokta aktörün isteğine uygun gerçekleştireceği adımların ve bu adımlara sistemin vereceği yanıtların açık bir şekilde ve eksiksiz olarak diyagramda bulunmasıdır. UML, bu akışı oluşturabilmeniz için gereken bir dizi hazır şema sağlar. Aktör nesnesi çöp adam şeklinde çizilmiştir. Yapılacak olan işlemler ise baloncuklar içerisinde eylem adlarıyla gösterilir. Aktör ile işlev arasındaki bağ aktörden baloncuğa doğru uzanan bir doğru ile gösterilir.

2.2.2.2. Class (Sınıf) Diyagramları

Sınıf diyagramlarında sisteme dâhil olan nesne tiplerini ve bu nesnelerin birbirleri ile olan ilişkileri tanımlamak için oluşturulur. Bu diyagramlar, sınıf yapısını ve içeriğini sınıflar, paketler ve nesneler kullanarak modeller ve bir sistemi 3 farklı açıdan; kavramsal, tanımsal ve yaşamsal açıdan ele alır. Sınıfüç bileşeni vardır: bir isim, nitelikler (property) ve işlemler (method). Sınıf diyagramları aynı zamanda içerme, kalıtım, ilişki ve diğer bağlantıları ifade eder. İlişki bağlantıları bir sınıf diyagramındaki en genel bağlantıdır. İlişki sınıfın örnek (instance)'leri arasındaki bağlantıları gösterir.

Sınıf diyagramlarında sık kullanılan bir diğer ilişki ise genelleştirmedir. Genelleştirme iki sınıfın birbirine benzer olup da sadece küçük farklarla ayrıldığı durumlarda kullanılır. Sınıf diyagramları zor çizilir diyagramlardır. Detaylı ve kullanışlı bir diyagram çizebilmek uzun süreli bir UML ve OO bilgisi gerektirir.

Sınıf diyagramı çizmeden önce, çizilecek diyagramın sistemi her üç açıdan da (kavramsal, tanımsal ve yaşamsal) göstermesi gerekir. Diyagram tek bir açıya yoğunlaşmamalı, tamamının bir arada nasıl çalışabileceğini göstermelidir.

Bir sınıfı tasarlarken ne tür özellikleri içermesi gerektiği ve ne tür işlemlere sahip olması gerektiğini göz önünde bulundurmak gerekir. Bu sınıftan türetilen örneklerin birbirleri ile nasıl etkileşeceği daha sonradan düşünülmelidir. Bunlar her ne kadar bir sınıf diyagram çizmenin ilk adımları olsa da bu temel çizim ile bile sistemin genel yapısı ortaya konulabilir.

2.2.2.3. Activity (Etkinlik) Diyagramları

Sistemin dinamik görünümünün modellenmesi etkinlik diyagramları ile yapılır. Aktiviteler arası geçişler temelde işlem akışı diyagramlarına benzerler. Aktivite sistemin işleyişini açıklamaktadır. İşlemler; ardışık, dallanma ya da eş zamanlı akışa sahip olabilir. Aktivite diyagramları tüm kontrol akışını ele alır.

Temelde sistem işleyişinin bir alt parçasıdır. Aktivite diyagramları sistemin dinamik yapısını göstermek dışında, çalıştırılabilir sistemin düzenlenmesinde de kullanılabilirler. Bu diyagramlarda mesajlaşma yoktur yani bir aktiviteden diğer aktivite akışına herhangi mesaj gösterilmez.

3. YAZILIM GELİŞTİRME SÜRECİ

Yapılacak olan yazılım için temel yazılım geliştirme süreci adımları izlenecektir. Bu adımları sırası ile;

- Gereksinimlerin toplanması (Planlama)
- Analiz
- Tasarım
- Geliştirme
- Test

Planlama, analiz ve tasarım sürecinde şablon olarak kullanılması amacı ile bir iş analizi dokümanı oluşturulabilir. Bu doküman ayrıca test sürecinde test senaryolarının oluşturulmasında da temel teşkil edecektir.

3.1. Analiz

Analizde amaç, yazılımın neyi nasıl yapacağını ve bunların kapsamının netleştirilmesi yani ihtiyacın veya problemin net olarak ifade edilmesini sağlamaktır. Genel başlıkları şunları içermelidir.

- Amaç
- Kapsam
- Beklentiler veya hedefler
- Fonksiyonel gereksinimler
- Fonksiyonel olmayan gereksinimler
- Sistem modeli
    - Aktörler
    - Olaylar
    - Senaryolar

Dokümandaki detaylar amaç ve ihtiyaca bağlı olarak çoğaltılabilir. Ancak temelde yukarıdaki başlıkları içermelidir.

3.2. Tasarım

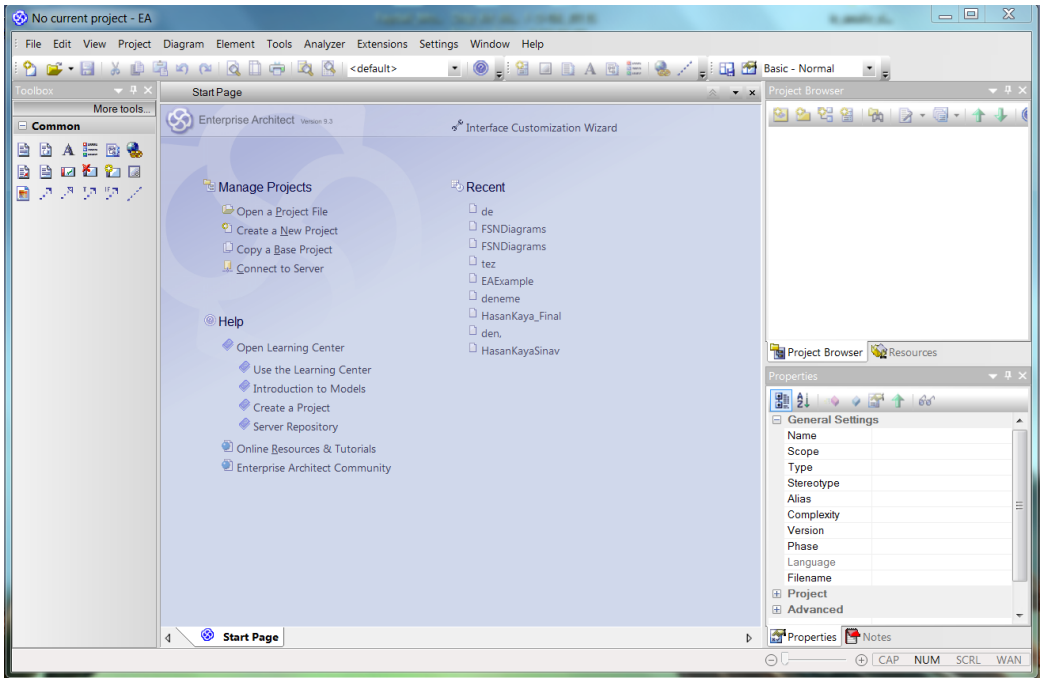
Yapılan analiz sonucunda belirlenen gereksinimleri karşılayacak olan yazılımın temelinin oluşturulmasına tasarım denir. Yazılımın bileşenleri belirlenir, bileşenler arası ilişkiler belirlenir, ara yüzler, mimarı tasarım, veri tasarımı, son kullanıcı ara yüz tasarımları, yazılım güvenliği ve performans kıstasları belirlenir. Bunlara bağlı olarak tasarım hazırlanır. Tasarımın içerisinde kullanılacak veri yapıları (Tablo ve sınıflar) hazırlanır. Fonksiyonlar giriş ve çıkış değerleri ile birlikte belirlenir. Ara yüz şablonları oluşturulur. Katmanlar arası veri akışının ve veri manipülasyonlarının detay dokümanları hazırlanır.

3.2.1 Tasarım Araçları

Nesne yönelimli analiz ve modellemenin en önemli standartlarından biri olan UML için piyasada birçok modelleme aracı bulunmaktadır. Fiyat ve performans açısından uygun bir seçenek olan Sparx Systems tarafından geliştirilen Sekil 3.1.'de kullanıcı ara yüzü görüntülenen Enterprise Architect v9.3. aracını bu süreçte kullanacaktır.

3.3. Geliştirme

Hazırlanan tasarım dokümanı üzerinden bilgisayar ortamında geliştirme ve veri işleme araçları kullanılarak modellemenin gerçeklemesi sağlanır. Bu sebeple hangi programlama dilinin kullanılacağı hangi veri yönetim aracının kullanılacağı ve hangi yazılım geliştirme araçlarının kullanılacağı bu aşamada belirlenir. Sonrasında kodlama(geliştirme) yapılır.



Şekil 3.1. Enterprise Architect v9.3. Kullanıcı Ara yüzü

3.4. Test

Geliştirme sürecinin ardında gerçekleştirilen bu süreçte doğrulama ve kontrol işlemleri yapılır. Geliştirilen uygulama gereksinimleri ne derece karşılıyor, yapılan kodlama ne kadar düzgün çalışıyor tespitleri yapılır. Test süreçleri yazılıma, platforma ve gereksinimlere bağlı olarak farklılık gösterebilir.

Bu süreç kodlamaya başlandıktan sonra başlar ve kodlama ile iç içe devam eder. Kullanılacak test yöntemi süreç içerisinde belirlenir. Belli başlı test yöntemleri şunlardır;

- Birim testi
- Tümleyim testi
- Regresyon testi
- Performans testi
- Kullanım kabul testi
- Beyaz kutu testi
- Kara kutu testi

4. AİLELERE YÖNELİK SOSYAL AĞ GELİŞTİRME SÜRECİ (FSN)

4.1. Amaç

Aile, aile içi ilişkiler ve aileler arası komşuluk ilişkilerini temel alan sosyal bir iletişim ağını tasarlamak ve oluşturmak.

4.2. Kapsam

Aileye ait genel tanımın yapılabileceği, aile bireylerinin bilgilerinin olduğu, farklı kullanıcıların ve ailelerin yazılı ve görsel olarak iletişime geçebileceği, etkinliklerin düzenlenebileceği, aile resimlerinin saklanıp paylaşılabileceği bir ağ oluşturmak.

4.3. Gereksinimler

Gereksinimler analiz sırasında sosyal ağdan beklenen işlevler temel alınarak oluşturulmuştur.

- Kullanıcı adı veya mail adresi ile kayıt olabilecekler
- Kullanıcı bilgilerine istedikleri zaman detay bilgi ve resim ekleyebilecekler, güncelleyebilecekler.
- Aile oluşturabilecekler, resim ve isim ekleyip güncelleyebilecekler.
- Parolalarını güncelleyebilecekler.
- Aile ye birey ekleyebilecekler, silebilecekler ve güncelleyebilecekler.
- Albüm oluşturabilecekler, paylaşabilecek ve güncelleyebilecekler.
- Etkinlik oluşturabilecekler, güncelleyebilecekler.
- Başka ailelere komşu veya akraba olarak istek gönderebilecekler.
- Birey olarak aileye istek gönderebilecekler.
- Aile ağacı oluşturabilecekler.
- Mesaj oluşturabilecek ve yönetebilecekler.

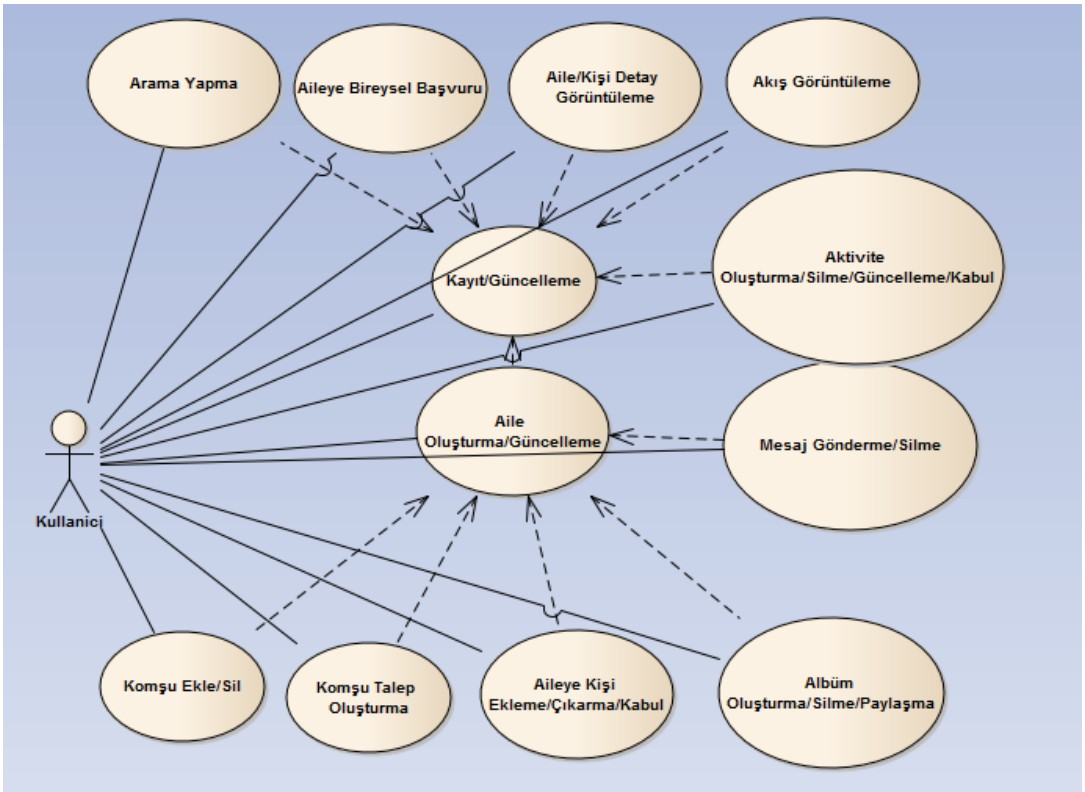
- Kişiler, aileler ve komşuları arasında arama yapabilecekler ve detaylarını görüntüleyebilecekler.
- Bildirim ekranlarından kendilerine gelen talepleri yönetebilecekler
    - Komşuluk talebi
    - Aile bireyi talebi
    - Etkinlikler
- Komşularını listeleyip, yönetip, detaylarına bakabilecekler.
- Yaklaşan doğum günlerini görebilecekler

4.4. Kullanıcılar

Kullanıcılar sosyal ağı kullanacak olan kullanıcı gruplarını ve bu kullanıcı gruplarının işlem yetkilerini belirlemek amacı ile oluşturulmuştur. Bu sosyal ağ için sadece tek bir kullanıcı belirlemiştir.

- Standart Kullanıcı

Standart kullanıcı hem kullanıcı hem de aile için yönetici hesaplardır. Ailelere katılabilirler ve aile içinde yetkilendirmeye bağlı olarak mevcut fonksiyonları kullanabilirler. Kendilerine aile oluşturabilirler. Kullanıcının sahip olduğu işlevler ve bunları diğer işlevler ile ilişkileri Şekil 4.1'deki gibidir.



Şekil 4.1. Kullanıcı İşlevleri Diyagramı

4.5. Olaylar

Olay listesi sistem için belirlenmiş olan kullacıların gerçekleştirebilecekleri işlemleri ifade eder. Her kullanıcı yetkisi dâhilinde bu işlemleri sistem üzerinde gerçekleştirebilir.

Olay Listesi

- Kayıt Olma
- Giriş
- Kullanıcı bilgisi güncelleme
- Aile oluşturma
- Aile bilgisi güncelleme
- Komşu talebi oluşturma
- Birey talebi oluşturma
- Bağımsız birey ekleme
- Bağımsız birey bilgi güncelleme

- Bağımsız birey silme
- Komşu talebi kabul etme
- Komşu silme
- Birey talebi kabul etme
- Birey silme
- Albüm oluşturma
- Albüm silme
- Etkinlik oluşturma
- Etkinlik silme
- Etkinlik kabul etme
- Aile ağacı oluşturma
- Arama yapma

4.6. Senaryolar

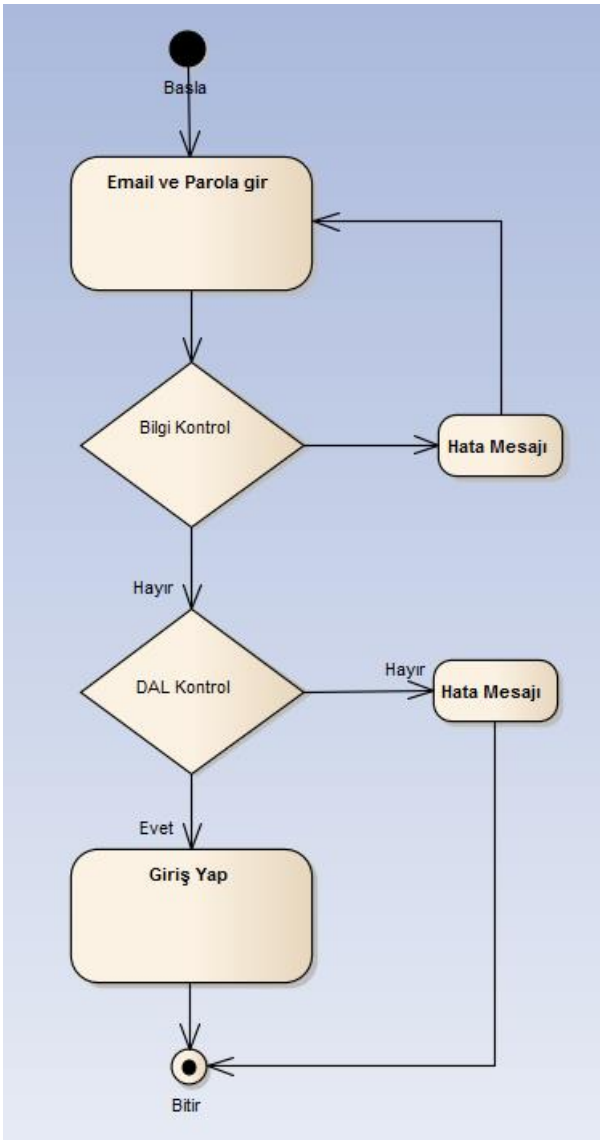
Senaryolarda kullanıcılar tarafında gerçekleştirilecek olaylar(işlemler) adım adım anlatılmıştır. Geliştirilecek olan sosyal ağda en çok kullanılacak olan işlevlerin senaryoları oluşturuldu.

Web sitesine giriş işleminin senaryo adımları Tablo 4.1'deki gibi oluşturulmuştur.

Tablo 4.1. Web Sitesi Giriş Senaryosu

Senaryo adı : Giriş Yap	
Kullanıcı : Standart Kullanıcı	
İşlem Adımları :	
	- Email ve Parola gir
	- Giriş butonuna bas
	- Bilgileri kontrol et
	- Hata varsa Hata Mesajı başa dön
	- Girişi tamamla

Web sitesine giriş senaryosunun aktivite diyagramı Şekil 4.2'deki gibidir.



Şekil 4.2. Kullanıcı Giriş Aktivite Diyagramı

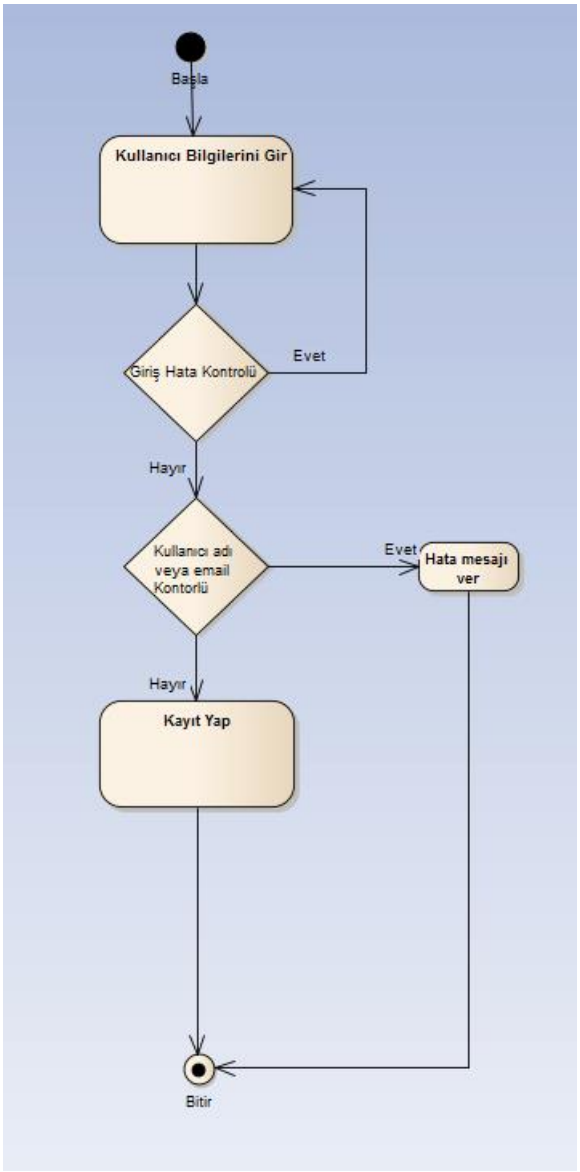
Web sitesine yeni kayıt olacak kullanıcıların kayıt işlemi senaryo adımları Tablo 4.2'de gibi oluşturulmuştur.

Tablo 4.2. Kullanıcı Kayıt Senaryosu

Senaryo adı	: Kayıt Yap
Kullanıcı	: Standart Kullanıcı
İşlem Adımları	:
	- Kayıt ol linkine tıkla
	- Kullanıcı adı gir

- Parola gir
- Tekrar parola gir
- Kayıt butonuna bas.

Web sitesi kayıt senaryosunun aktivite diyagramı Şekil 4.3'deki gibi oluşturulmuştur.



Şekil 4.3. Kayıt Aktivite Diyagramı

Kullanıcıların bilgilerini güncellemesi için izleyecekleri senaryo adımları Tablo 4.3'deki gibi olacaktır.

Tablo 4.3. Kullanıcı Kayıt Güncelleme Senaryosu

Senaryo adı	: Kayıt Güncelle
Kullanıcı	: Standart Kullanıcı
İşlem Adımları	:
	- Girişi linkine tıkla
	- Kullanıcı adı gir
	- Parola gir
	- Kayıt güncelleme butonuna bas.
	- İsim gir
	- Soy isim gir
	- Eposta adresi gir
	- Doğum tarihi gir
	- Cinsiyet gir
	- Resim seç
	- Güncelleme butonuna bas

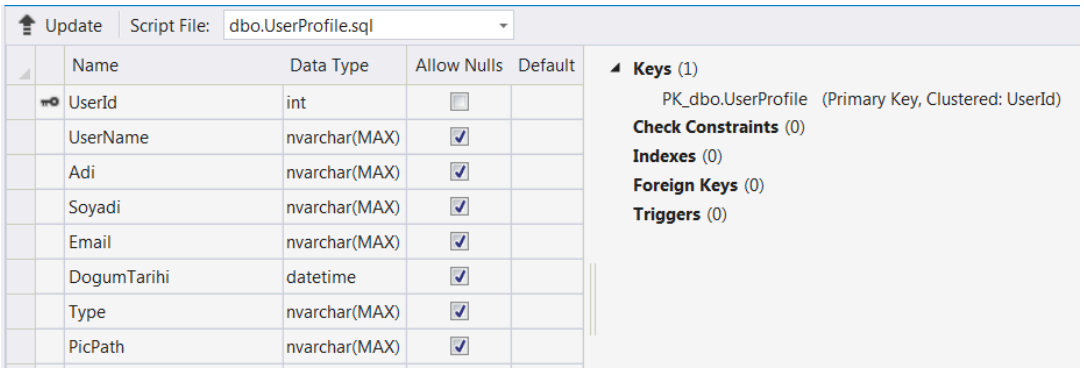
Her bir kullanıcı işlevi için senaryo adımları belirlenir ve senaryo adımlarına uygun olarak aktivite diyagramları oluşturulur.

4.7. Veri Yapısı

Sosyal ağın tanımı ve tanıma bağlı yapılan analiz ile oluşturulan gereksinimleri karşılanması, kullanıcılar için belirlenen işlemlerinin gerçekleştirilebilmesi için veri tabanı yapısı oluşturulmuştur.

Tablo Adı : KullanıcıProfili

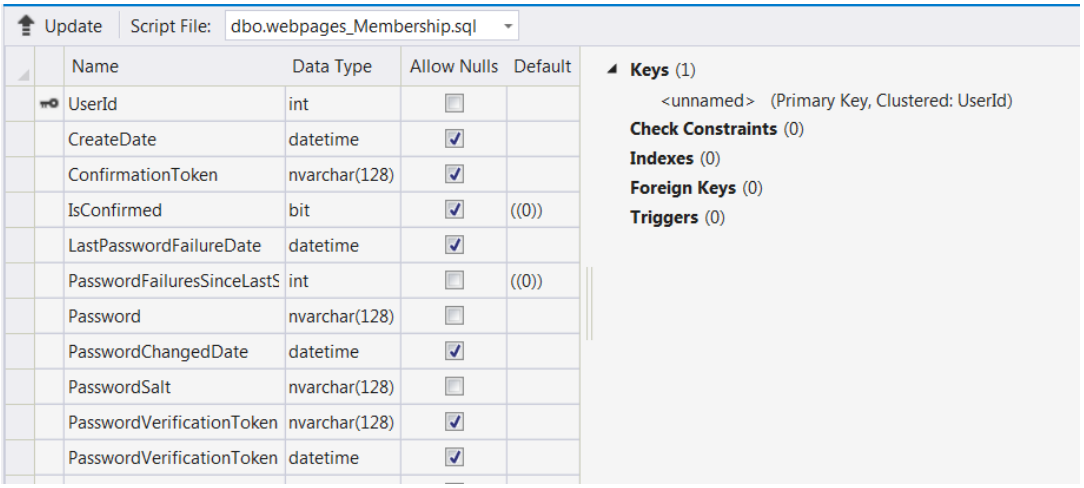
Tablo Açıklama : Kullanıcı kayıt ve bilgi güncelleme işlemlerinde oluşan veriler Şekil 4.4'de ki tablo yapısında tutulmaktadır.



Şekil 4.4. KullanıcıProfili **(**UserProfile) Tablosu

Tablo Adı : VebSayfasi_Üyeleri

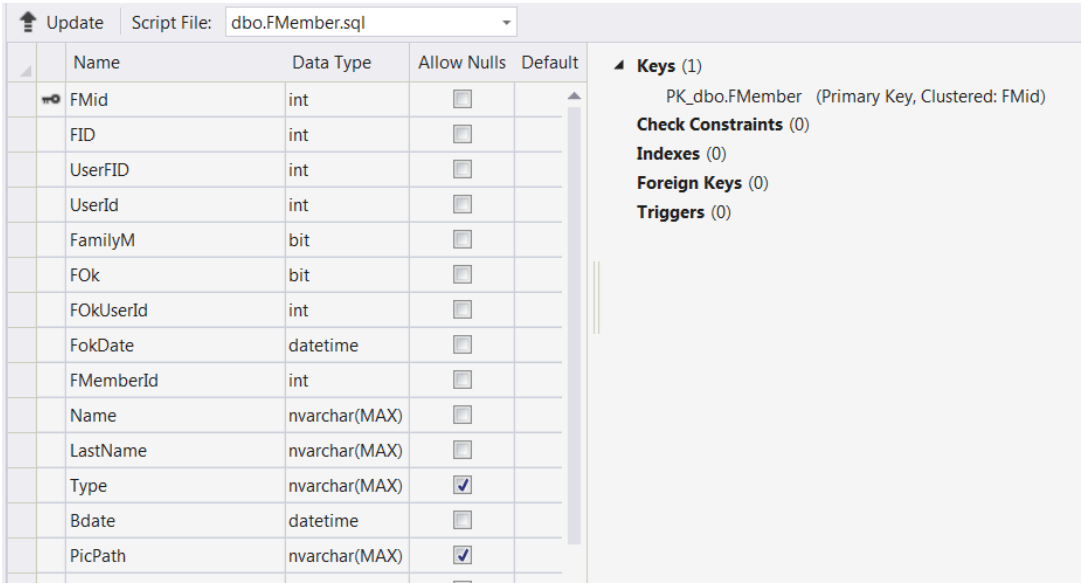
Tablo Açıklama : Kayıtlı kullanıcılarında kayıt bilgileri onaylama ve parola değişiklikleri işlem bilgilerinin tutulduğu tablo yapısı Şekil 4.5'deki gibidir.



Şekil 4.5. VebSayfasıUyeleri **(**webpages_Membership) tablosu

Tablo Adı : AUyeler

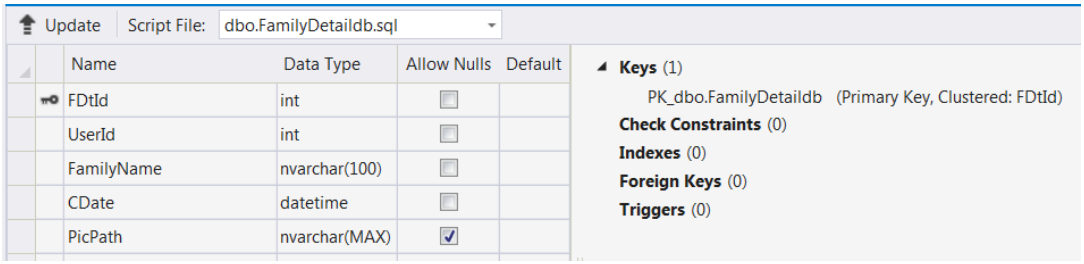
Tablo Açıklama :Aileye üye olan dış kullanıcıların ve yönetici hesap tarafından eklenmiş aile üyelerinin verilerinin tutulduğu tablo yapısı Şekil 4.6'daki gibidir.



Şekil 4.6. AUyeler **(**FMember) Tablsu

Tablo Adı : AileDetaydb

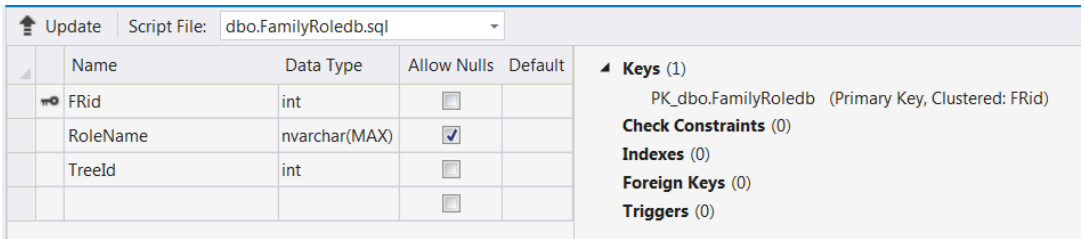
Tablo Açıklama :Aileye ait tanımlama verilerinin tutulduğu tablo yapısı Şekil 4.7'deki gibidir.



Şekil 4.7. AileDetaydb **(**FamilyDetailDb) Tablosu

Tablo Adı : AileRoldb

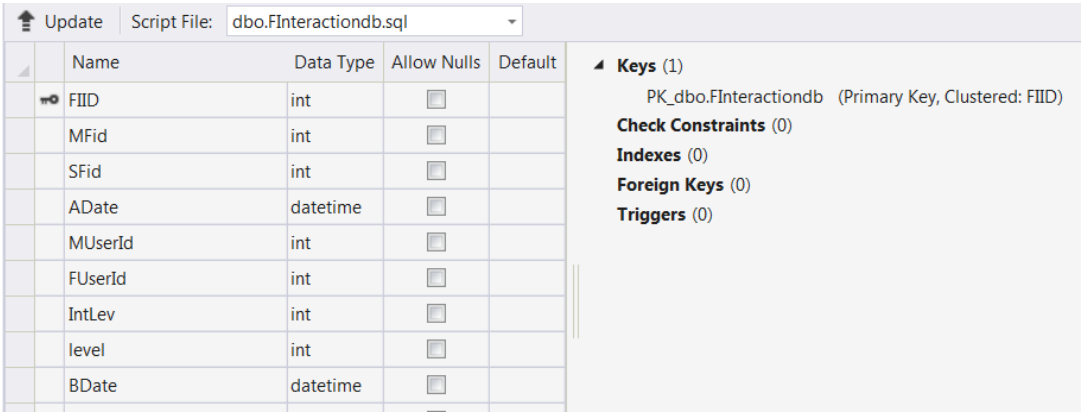
Tablo Açıklama : Aile üyelerinin aile içindeki rol tiplerine ait verilerin tutulduğu tablo yapısı Şekil 4.8'deki gibidir.



Şekil 4.8. AileRoldb **(**FamilyRoledb) Tablosu

Tablo Adi : Aİlişkidb

Tablo Açıklaması : Aileler arası etkileşim düzeyine ait verilerin tutulduğu tablo yapısı Şekil 4.9'daki gibidir.



Şekil 4.9. Aİlişkidb **(**FInteractiondb) Tablosu

Tablo Adi : AkisLog

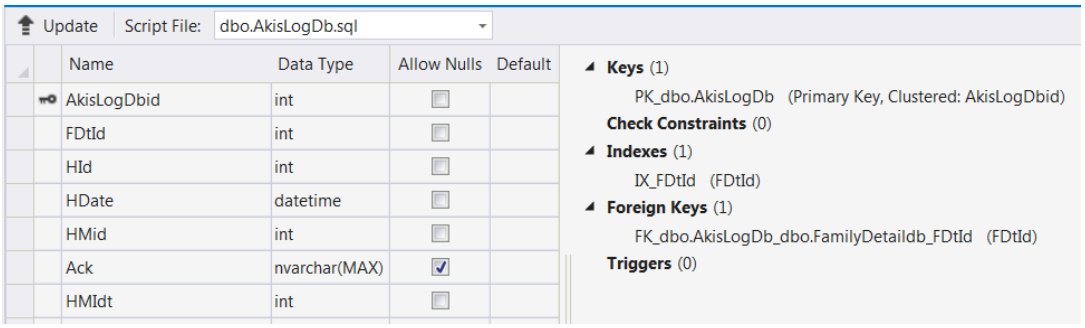
Tablo Açıklaması : Kullanıcıların yaptığı işlemlere ait verilerin ve ana sayfada görüntülenecek olan olay akışlarının tutulduğu tablo yapısı Şekil 4.10'daki gibidir.

Tablo Adi : Albumler

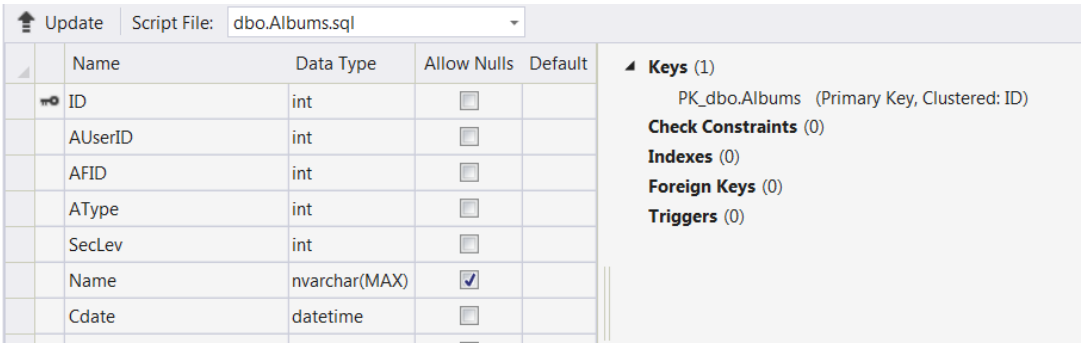
Tablo Açıklaması : Kullanıcı resimlerinin saklandığı albümlere ait verilerin tutulduğu tablo yapısı Şekil 4.11'deki gibidir.

Tablo Adi : AlbumDetay

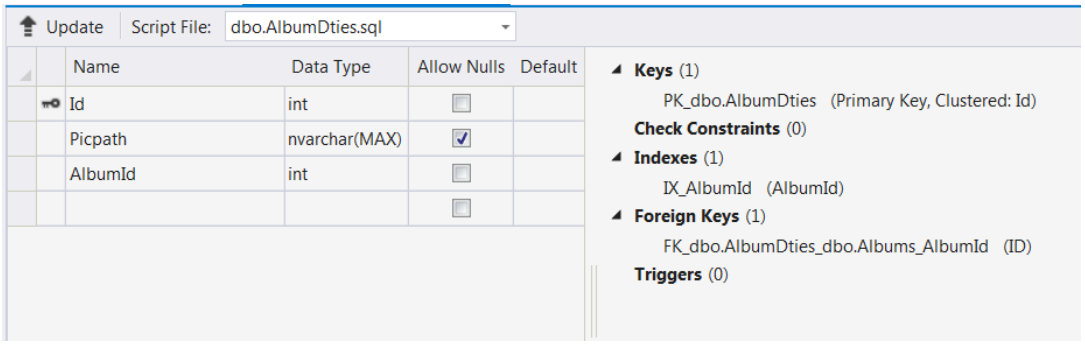
Tablo Açıklaması : Albümlerin içerisindeki resimlerin her birine ait saklama ve albüm ilişkisi verilerinin tutulduğu tablo yapısı Şekil 4.12'deki gibidir.



Şekil 4.10. AkisLog Tablosu



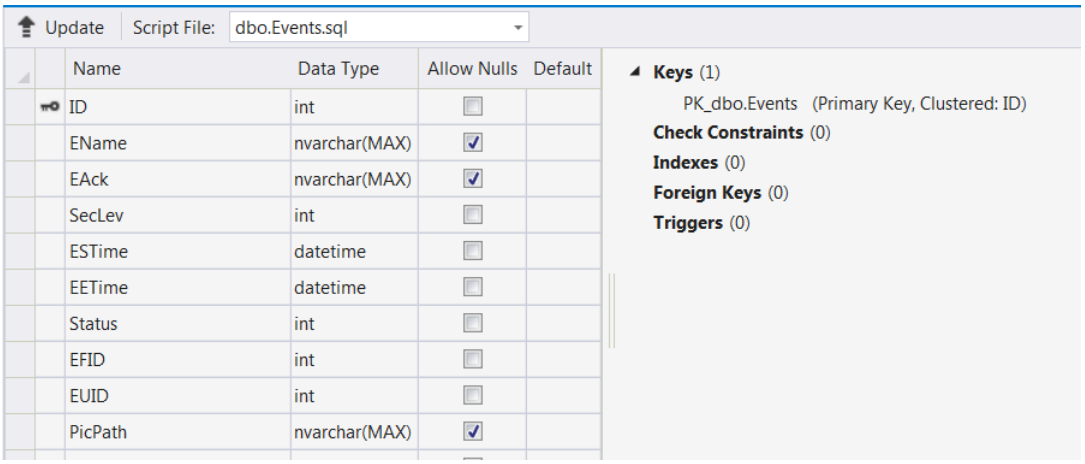
Şekil 4.11. Albumler **(**Albums) Tablosu



Şekil 4.12. AlbumDetay **(**AlbumDties) Tablosu

Tablo Adi : Etkinlikler

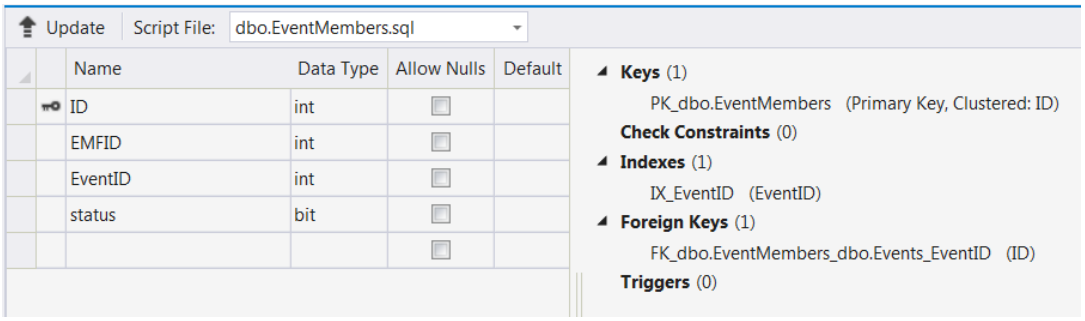
Tablo Açıklaması : Kullanıcılar tarafından oluşturulan etkinliklere ait verilerin tutulduğu tablo yapısı Şekil 4.13'deki gibidir.



Şekil 4.13. Etkinlikler **(**Events) Tablosu

Tablo Adi : EtkinlikUyeleri

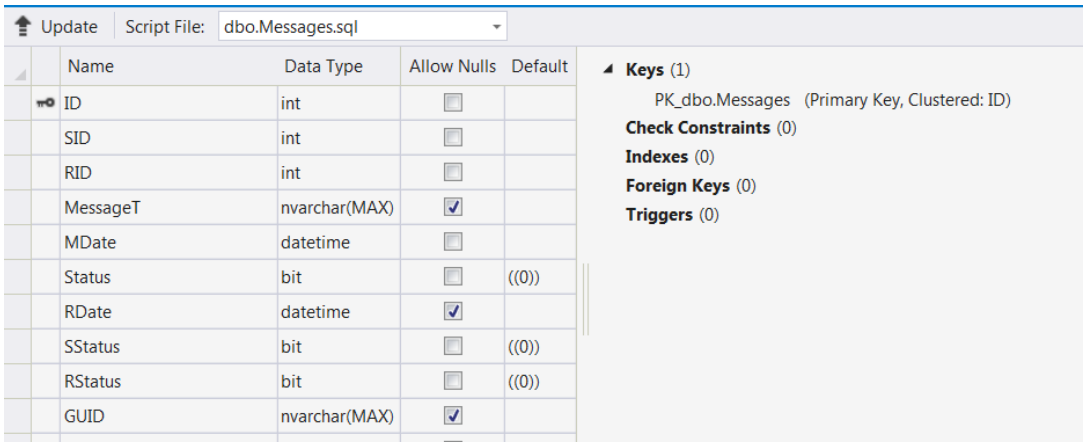
Tablo Açıklaması : Düzenlenen etkinliklere katılacak olan kullanıcılara ait verilerin tutulduğu tablo yapısı Şekil 4.14'deki gibidir.



Şekil 4.14. EtkinlikUyeleri **(**EventMembers) Tablosu

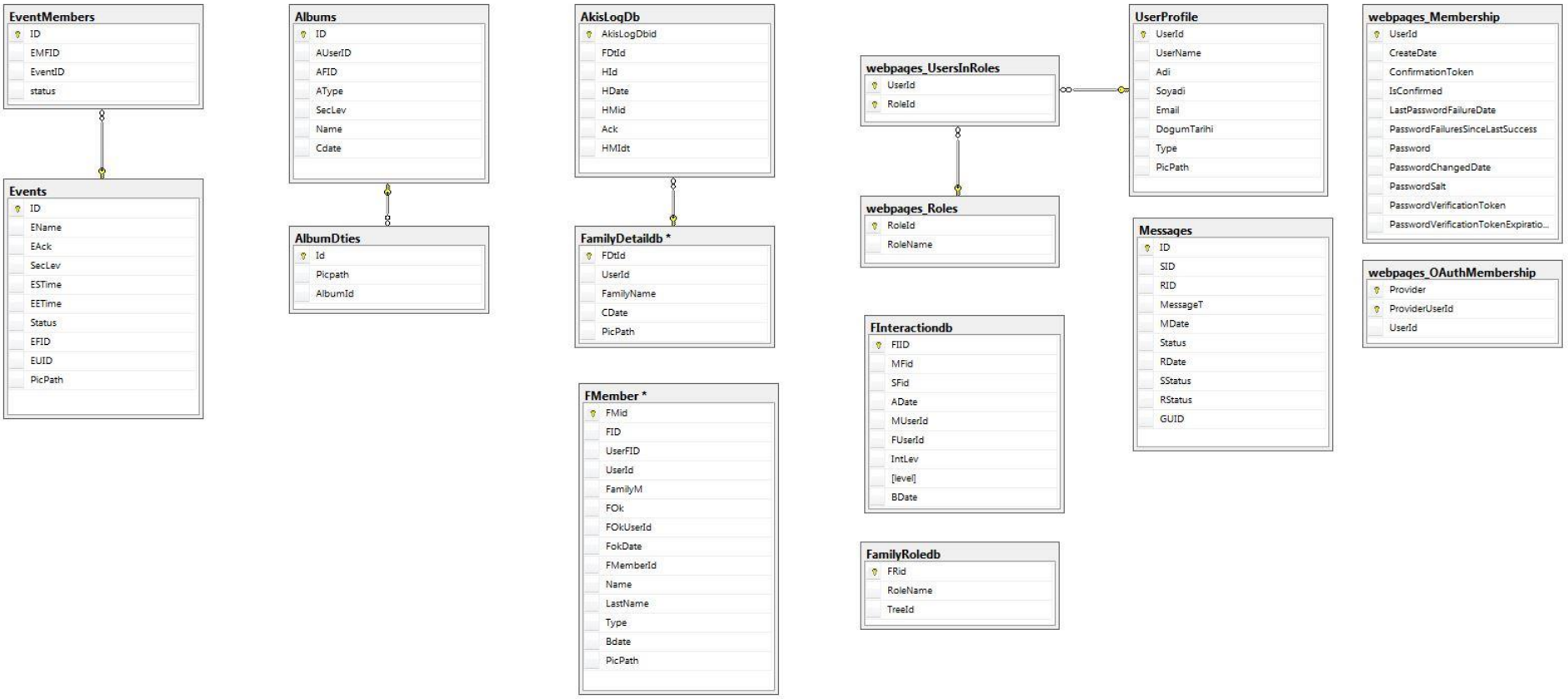
Tablo Adi : Mesajlar

Tablo Açıklaması : Kullanıcılar arası mesajlaşma verilerinin tutulduğu tablo yapısı Şekil 4.15'deki gibidir.



Şekil 4.15. Mesajlar **(**Messages) Tablosu

Veri tabanındaki tablolar arası ilişkilerin genel görünümü Şekil 4.16'daki gibidir.



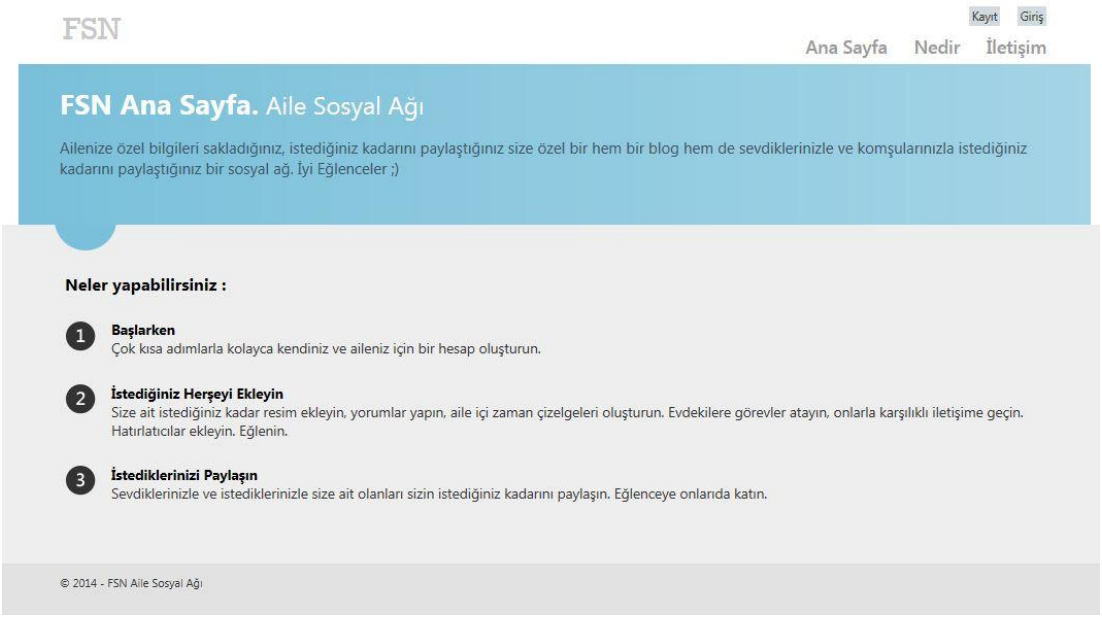
Şekil 4.16. Veri tabanı modeli genel görünümü

4.8. Kullanıcı Ara yüzleri

Kullanıcıların işlem yapabilecekleri ekranlara ait ekran görüntüleri ve detaylı açıklamaları bu kısımda yer almaktadır. Anlatım sırası ilk kayıt ve sonrasında yapılacak işlemler göz önüne alınarak oluşturulmuştur.

Web sitesi kullanıcı karşılama ekranı Şekil 4.17'deki gibidir, bu sayfa site hakkında genel bilgileri barındırır. Aynı zamanda kullanıcılar burada verilen "Kayıt" linkine tıklayarak isterlerse siteye kayıt olabilir, "Giriş" linkine tıklayarak ta önceden kayıt olmuş kullanıcılar ise kendileri için özelleştirilmiş ana sayfalarına giriş yapabilirler.

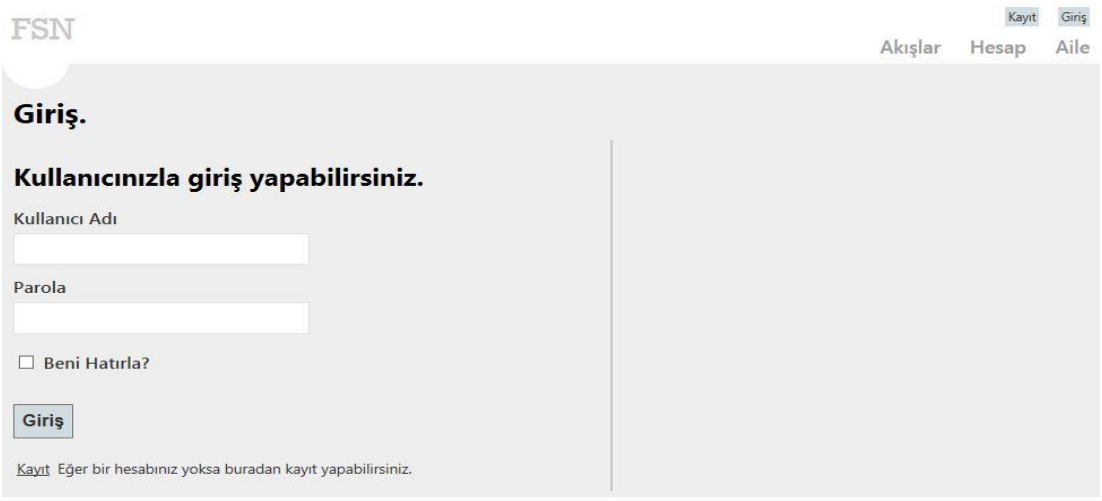
Ayrıca "Nedir" ve "İletişim" linklerine tıklayarak siteye ait diğer bilgilere ulaşabilirler.



Şekil 4.17. Web sitesi ana ekranı

Kullanıcılar daha önceden kayıtlı ise Şekil 4.18'deki Kullanıcı Giriş Ekranını kullanarak sisteme giriş yapacaklardır.

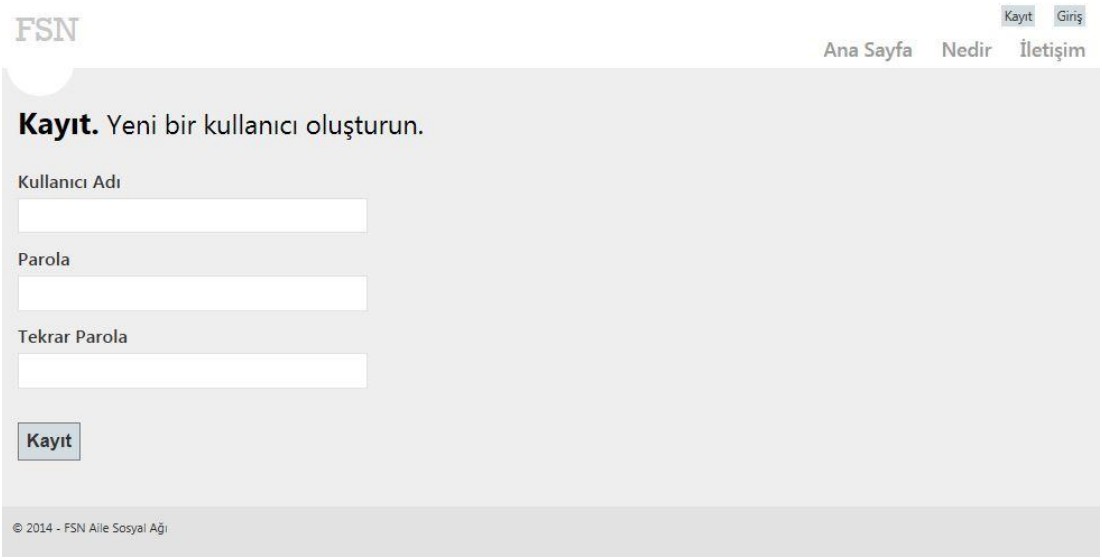
Bu ekranda kullanıcı daha önceden kayıt işlemi tamamlanmış olan hesabına Kullanıcı Adı ve Parola bilgilerini girip, "Giriş" butonuna basarak giriş işlemini tamamlayabilir. Ayrıca Beni Hatırla seçeneğini kullanıp daha sonraki girişlerinde browserların otomatik doldurma özelliğini kullanarak daha kolay bir giriş yapabilir.



Şekil 4.18. Kullanıcı giriş ekranı

Siteye yeni kayıt olmak isteyen kullanıcılar kayıt işlemini Şekil 4.19'daki Kullanıcı Kayıt Ekranı üzerinden yapacaklardır.

Bu ekranda kullanıcı siteye kayıt için gerekli olan alanları doldurur. Kullanıcı Adını doldurduktan sonra doğrulama için iki kez parola bilgisi istenir. Kayıt butonuna basılır, hatalı girişin önüne geçmek için parolalar karşılaştırma işlemi yapılarak kontrol edilir. Kullanıcı adı yine sistemde daha önceden kayıtlı mı diye kontrol edilir. Her iki kontrolde hata alınmaz ise kayıt işlemi tamamlanır.



Şekil 4.19. Kullanıcı kayıt ekranı

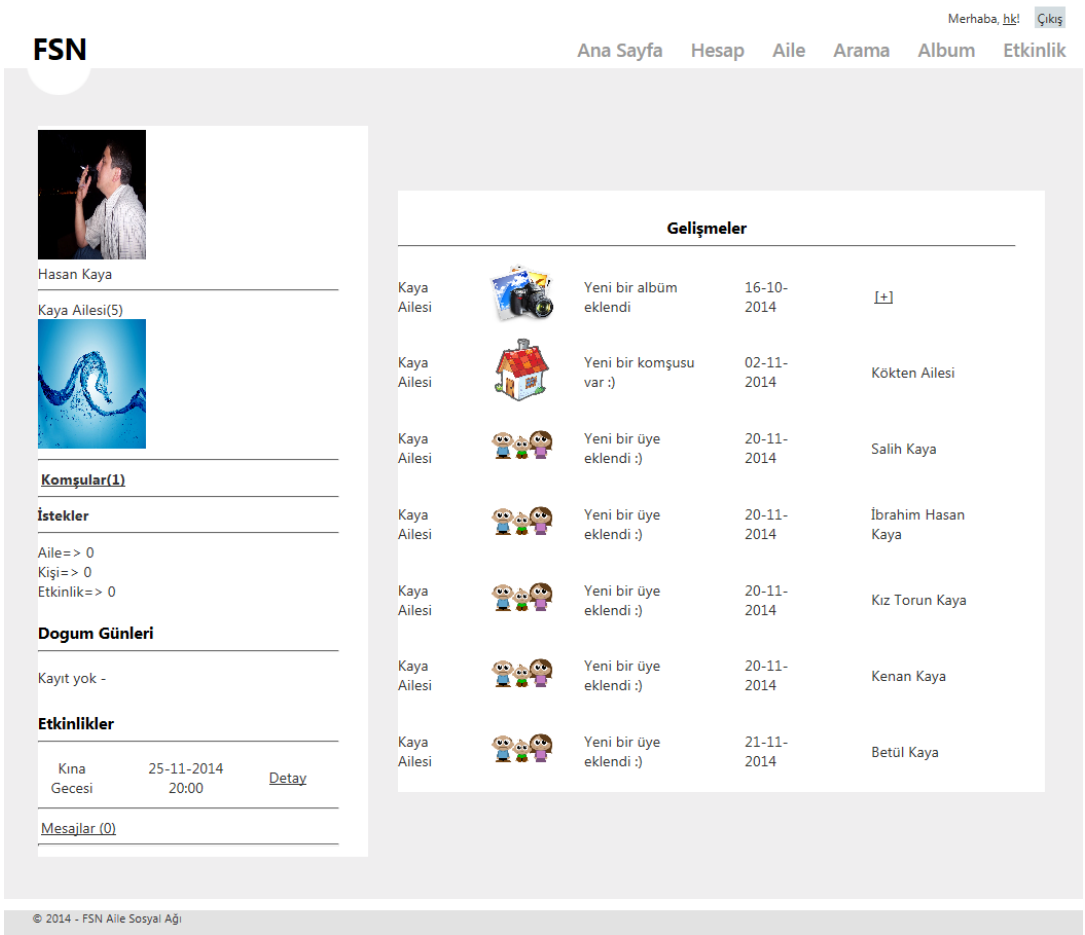
Kullanıcı karşılama ana ekranı Şekil 4.20'deki gibidir.

Bu ekranda kullanıcı için gerekli tüm bilgilendirmeler yer alır. Kullanıcı yönetim linkleri, kullanıcı bilgisi, kullanıcı aile bilgisi, komşular listesi, istekler bilgi ekranı, etkinlik ve doğum günleri hatırlatma ekranı, diğer kullanıcıların işlem bilgilendirme ekranı ve mesaj ekranına ulaşım linkleri bu ekrandadır.

Ayrıca ekranın sağ üst köşesinde karşılama mesajı ve giriş yapmış kullanıcı bilgisi yer alır. İstenildiğinde çıkış yapılabilmesi için "Çıkış" linki de bu bölümdedir.

Ana işlemler arası hızlı geçiş yapmak ve kolay kullanım için yatay işlemler menüsü de tüm ekranlarda olduğu gibi bu ekranda da yer alır. Kullanıcı işlem yapmak istediği linke tıklayarak istediği bölüme kolayca geçebilir.

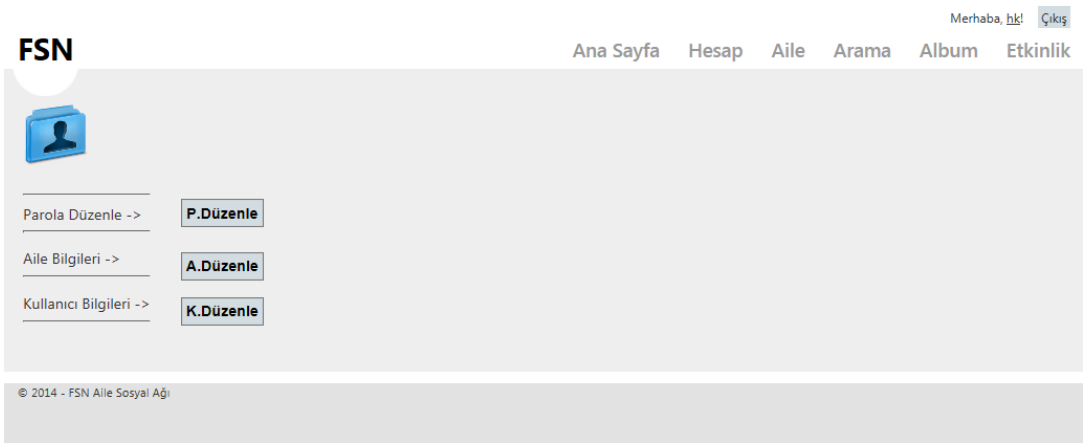
Site ile ilgili işlemlerin tamamına bu ekran üzerinden ulaşabilir. Sayfanın kaynak kodu ekler bölümünde detaylı olarak verilmiştir.**(EK 1)**



Şekil 4.20. Kullanıcı karşılama ana ekranı

Kullanıcı hesap yönetim ekranı Şekil 4.21'daki gibidir.

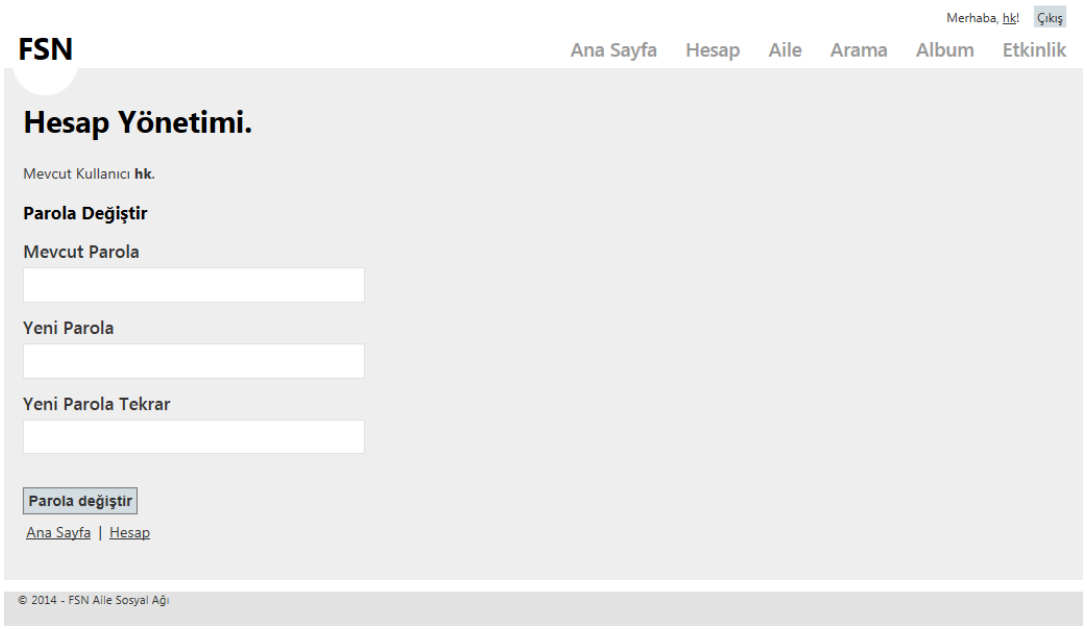
Bu ekranda kullanıcı hesabı ile ilgili olarak parola değiştirme, aile bilgileri görüntüleme ve düzenleme ile kullanıcı bilgileri görüntüleme ve düzenleme ekranlarına ulaşabilir. Kullanıcı işlemlerine kolayca ulaşılması için kullanıcı kontrol linkleri tek bir ekran üzerinde tasarlanmıştır.



Şekil 4.21. Kullanıcı hesap yönetim ekranı

Kullanıcı parola değiştirme ekranı Şekil 4.22'deki gibidir.

Bu ekranda kullanıcı parolasını istediği zamanda eski parola bilgisini girerek kontrollü bir şekilde değiştirebilir.



Şekil 4.22. Kullanıcı parola değiştirme ekranı

Kullanıcı aile detay ekranı Şekil 4.23'deki gibidir.

Kullanıcı burada daha öncede tanımı yapmış olduğu aile bilgilerini ve ne zaman işlem yaptığı bilgisine ulaşabilir ve isterse "Düzenle" linkine basarak güncelleme ekranına yönelebilir.

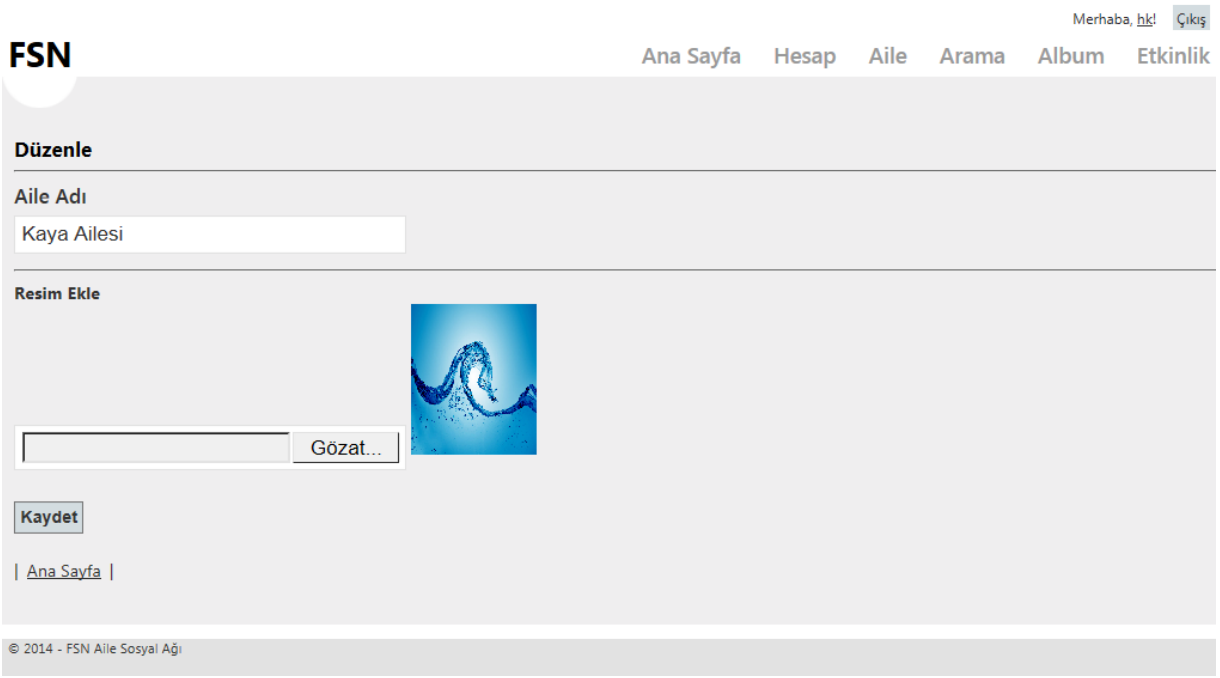
Şekil 4.23. Kullanıcı aile detay ekranı

Kullanıcı aile detay düzenleme ekranı Şekil 4.24'deki gibidir.

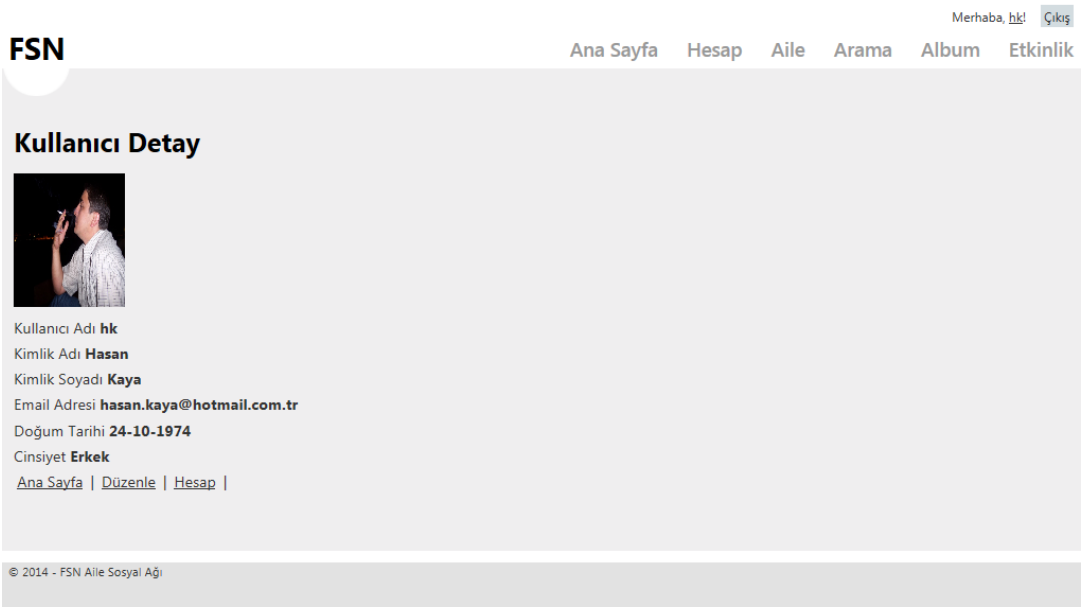
Kullanıcı aile detay düzenleme ekranında kullanıcı aile adını ve resmini değiştirebilir, eğer ilk kez girmiş ise tanımlama yapıp bilgileri kaydedebilir.

Kullanıcı detay görüntüleme ekranı Şekil 4.25'deki gibidir.

Bu ekranda kullanıcı kendine ait önceden girilmiş bilgileri görüntüleyebilir. İsterse "Düzenle" linkine tıklayarak değişiklik ekranına yönelebilir.



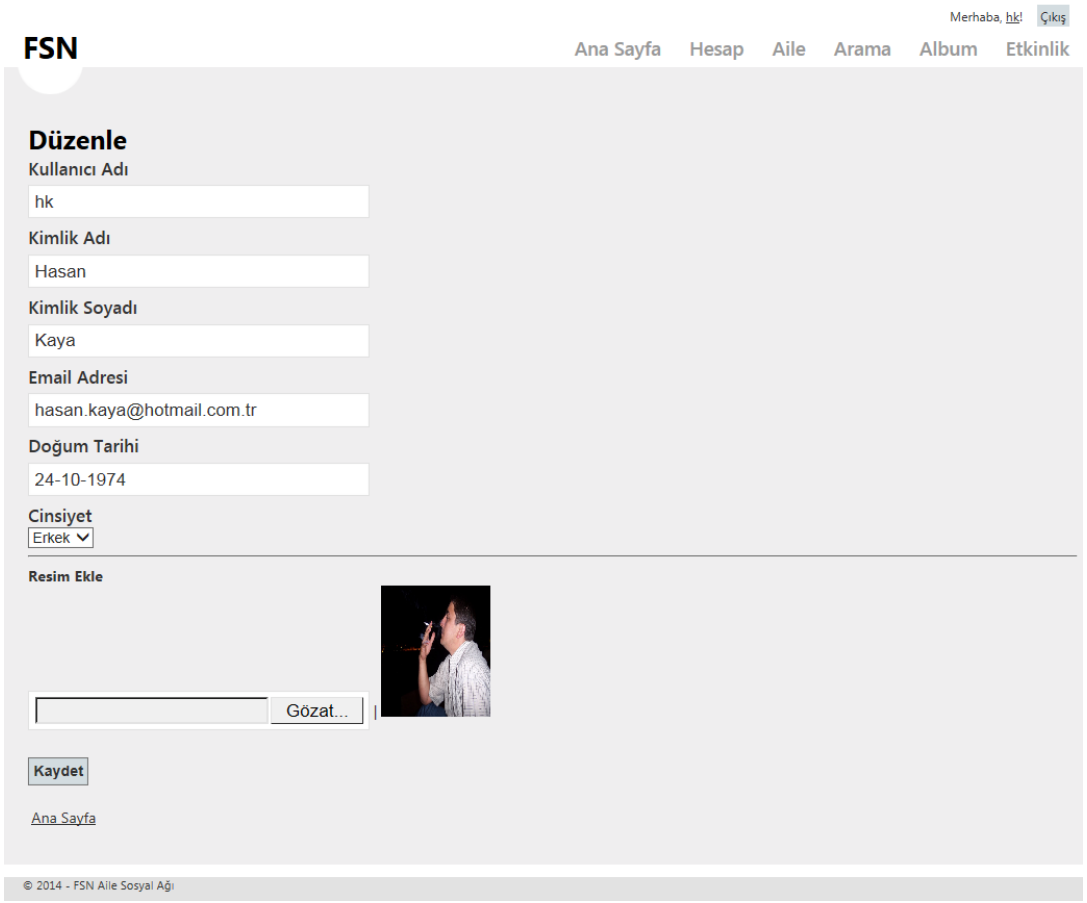
Şekil 4.24. Kullanıcı aile detay düzenleme ekranı



Şekil 4.25. Kullanıcı detay görüntüleme ekranı

Kullanıcı bilgileri detay düzenleme ekranı Şekil 4.26'deki gibidir.

Bu ekranda kullanıcı kendine ait bilgileri güncelleyebilir, resim ekleyebilir. İlk kez girdi ise bilgileri girip kayıt işlemi yapabilir.



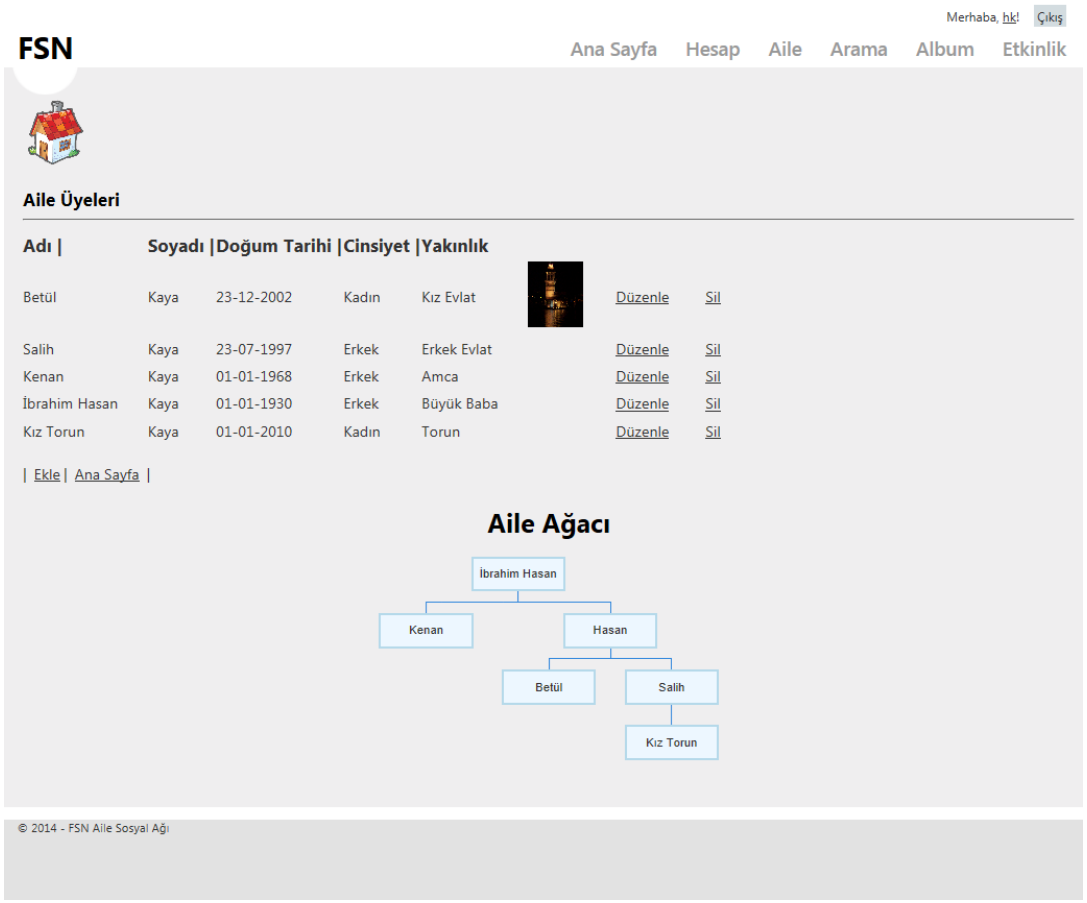
Şekil 4.26. Kullanıcı detay düzenleme ekranı

Kullanıcı aile işlemleri ekranı Şekil 4.27'deki gibidir.

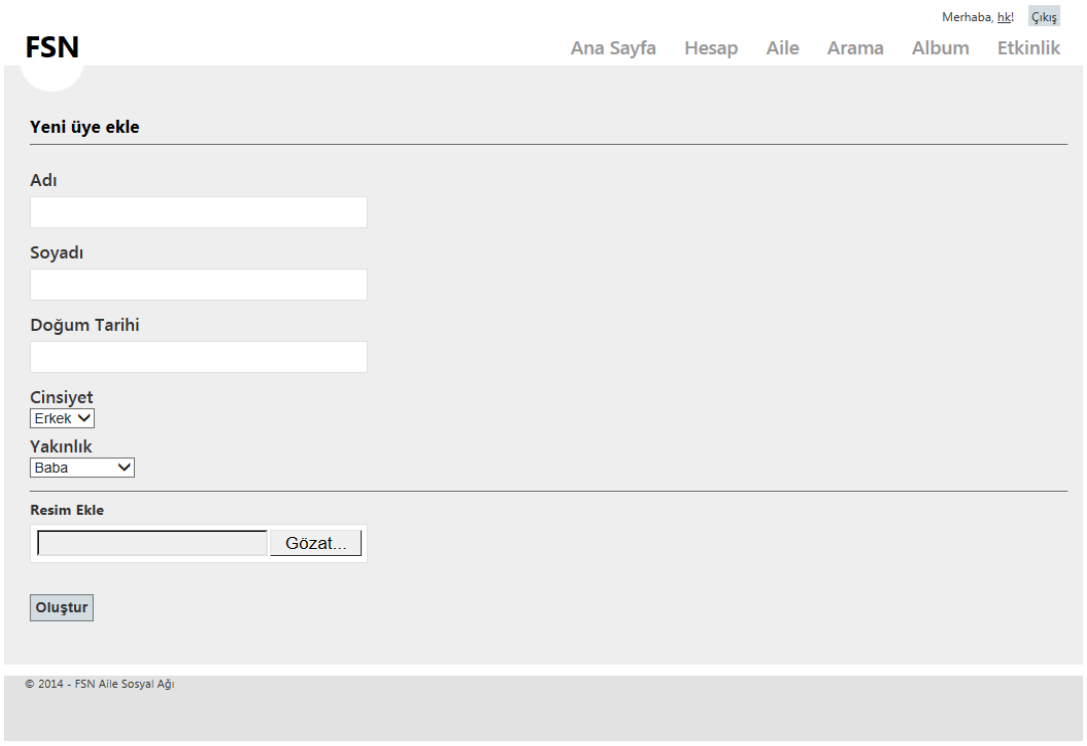
Kullanıcı bu ekranda aile bireylerinin listesini detayları ile birlikte listeli olarak görüntüleyebilir, "Ekle" linki ile yeni üye ekleme ekranına yönlenebilir, mevcut üye üzerinde değişiklik yapmak için "Düzenle" linki ile işlem ekranına gidebilir veya "Sil" linkine tıklayarak silebilir. Bu işlemlerin sonucunda aile ağacı görüntüsü dinamik olarak oluşturulur ve görüntülenir. Ekrana ait kaynak kodu ekler bölümünde detaylı olarak verilmiştir.**(EK 2)**

Aileye yeni üye ekleme ekranı Şekil 4.28'deki gibidir.

Bu ekranda kullanıcı aileye yeni bir üye ekleyebilir, gerekli bilgileri doldurup kullanıcıya özel resim ekleyerek kayıt işlemi yapabilir.



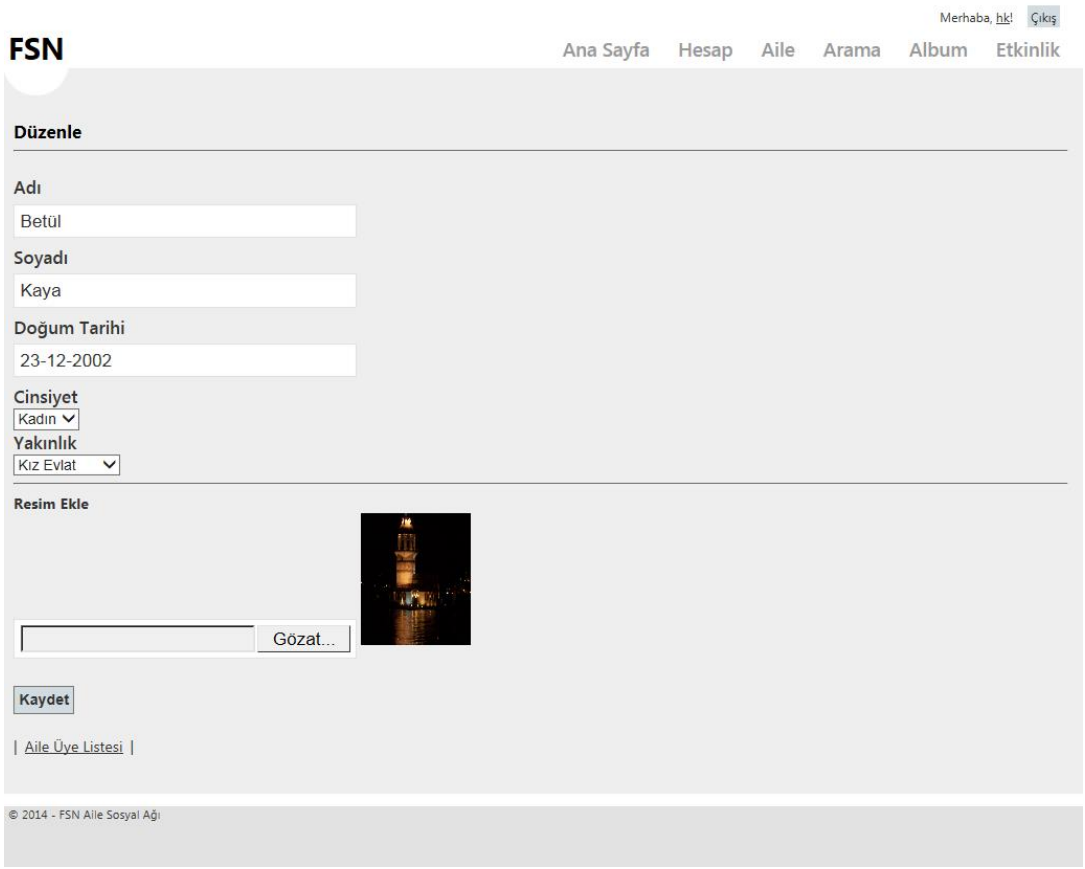
Şekil 4.27. Aile işlemleri ekranı



Şekil 4.28. Yeni üye ekleme ekranı

Aileye üye düzenleme ekranı Şekil 4.29'deki gibidir.

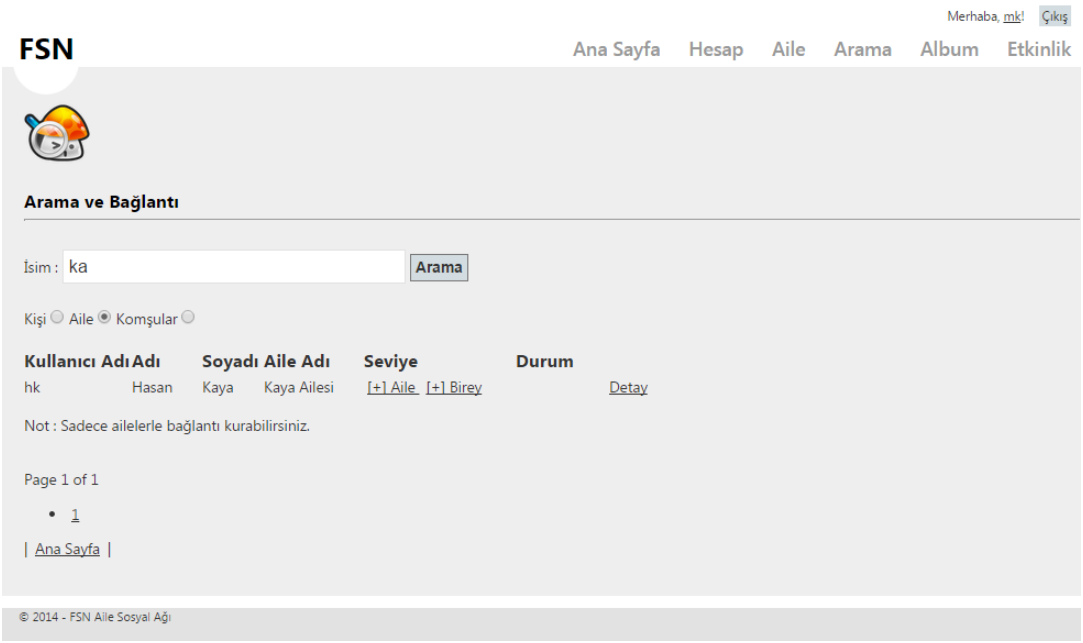
Bu ekranda kullanıcı daha önceden kendisi tarafından kayıt işlemi yapılmış aile üyesine ait bilgileri güncelleyip kayıt işlemi yapabilir.



Şekil 4.29. Üye güncelleme ekranı

Arama ve bağlantı ekranı Şekil 4.30'daki gibidir.

Bu ekrandan kullanıcılar ağ üzerindeki kişileri, aileleri ve kendi komşuları içindeki aileler arasında arama işlemi yapabilirler, listelenen ailelere bireysel veya aile olarak etkileşim talebinde bulunabilirler ve ailelere ait bilgilerin detaylarını görüntüleyebilirler. Bu ekranda eğer kişi ve aileler sizinle etkileşim içinde ise "Durum" alanında bu bilgi görüntülenir. Ekrana ait detaylı kaynak kodları ekler bölümünde verilmiştir.**(EK 3)**



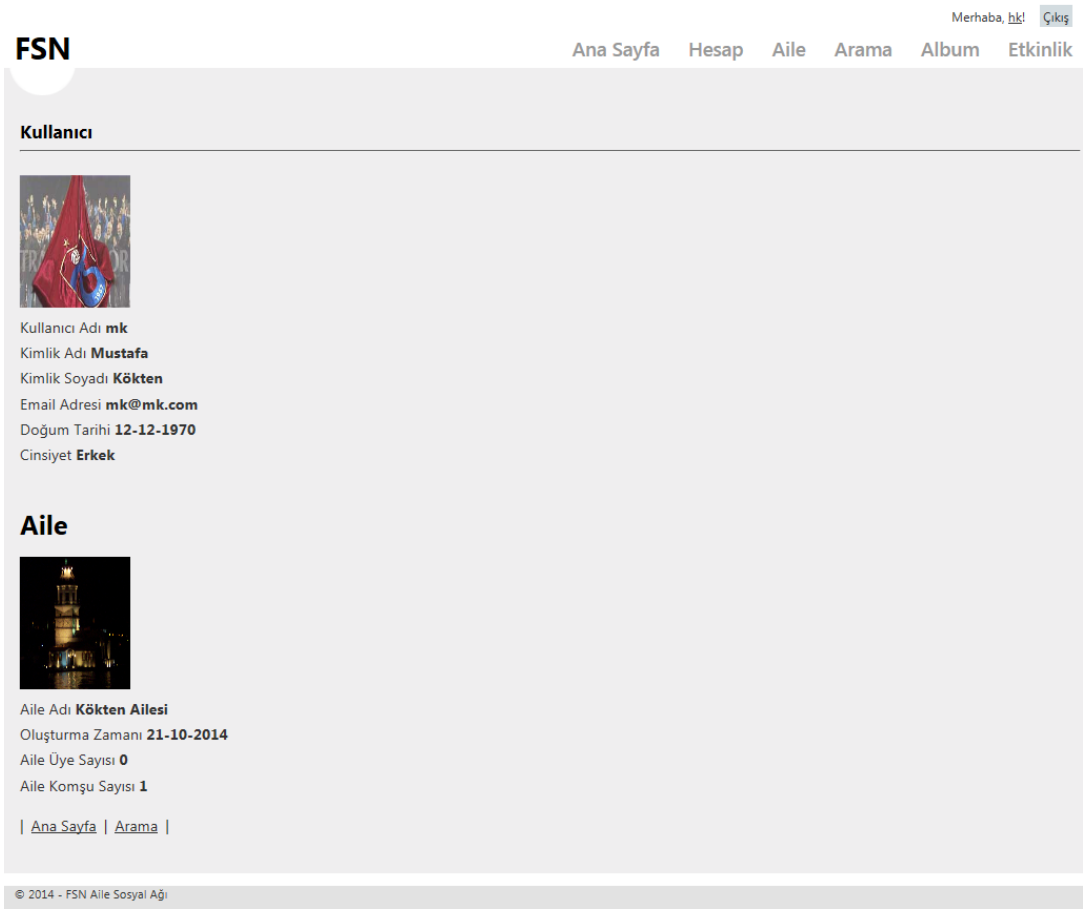
Şekil 4.30. Arama ve bağlantı ekranı

Aile detay ekranı Şekil 4.31'deki gibidir.

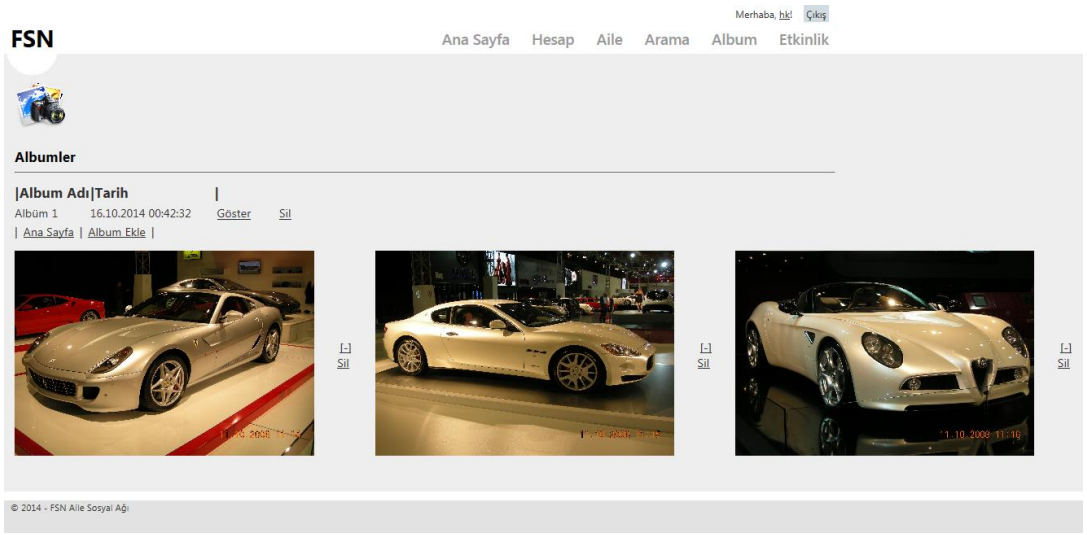
Bu ekranda kullanıcılar arama sonuçlarında çıkan ailenin detaylarını görüntülemek için "Detay" linkine tıkladıklarında aileye ve ailenin yönetici hesabı olan kullanıcıya ait bilgileri görüntüleyebilirler.

Albüm yönetim ekranı Şekil 4.32'deki gibidir.

Bu ekranda kullanıcılar kendilerine ait albümlerin listesini, oluşturma zamanı, albüm ismi gibi detayları da içerir şekilde görebilirler, albüm içinde resimleri görmek için göster linkine tıklayarak aynı ekranda detaylarına ulaşabilirler, isterlerse albümün tamamını veya istedikleri herhangi bir resmi silebilirler. Yeni bir albüm oluşturmak istediklerinde "Albüm Ekle" linkine tıklayarak oluşturma ekranına yönlenebilirler.



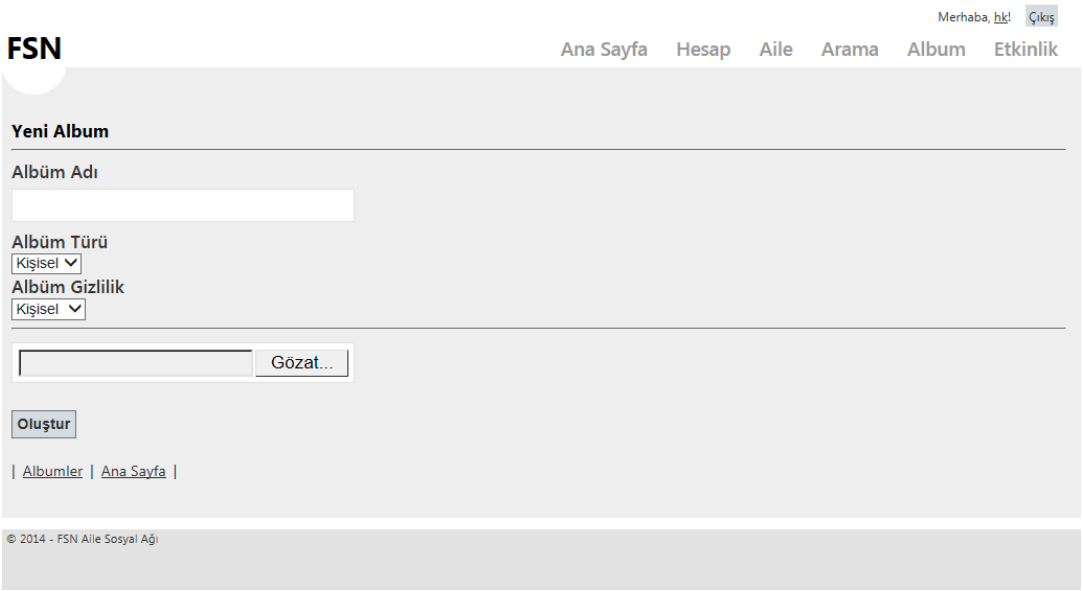
Şekil 4.31. Aile ekranı



Şekil 4.32. Albüm yönetim ekranı

Albüm oluşturma ekranı Şekil 4.33'deki gibidir.

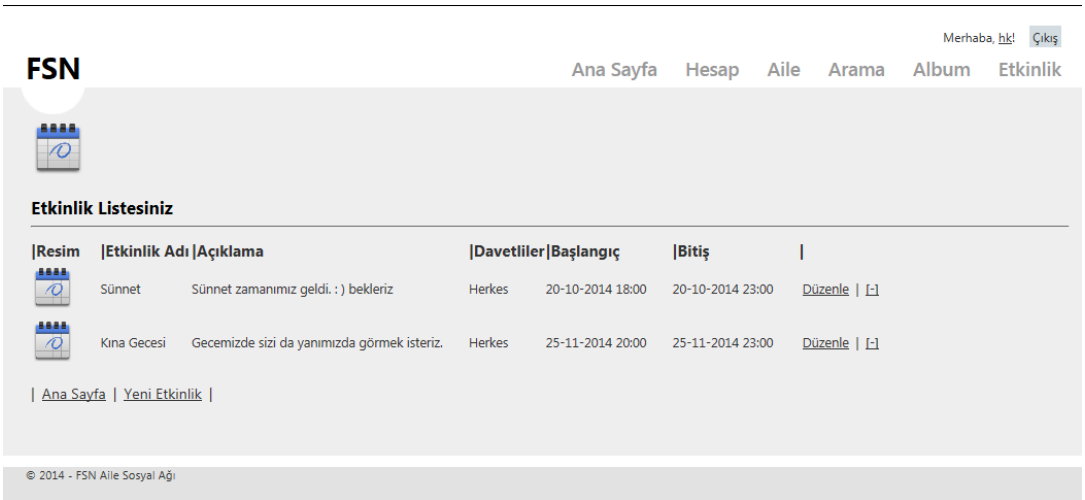
Bu ekranda kullanıcılar yeni albümlerini paylaşım seviyesini belirleyerek hem aileleri için hem de kişisel albümlerini oluşturabilirler. Aynı anda birden çok resmi seçerek tek seferde birçok resim ekleyebilirler.



Şekil 4.33. Albüm oluşturma ekranı

Etkinlik yönetim ekranı Şekil 4.34'deki gibidir.

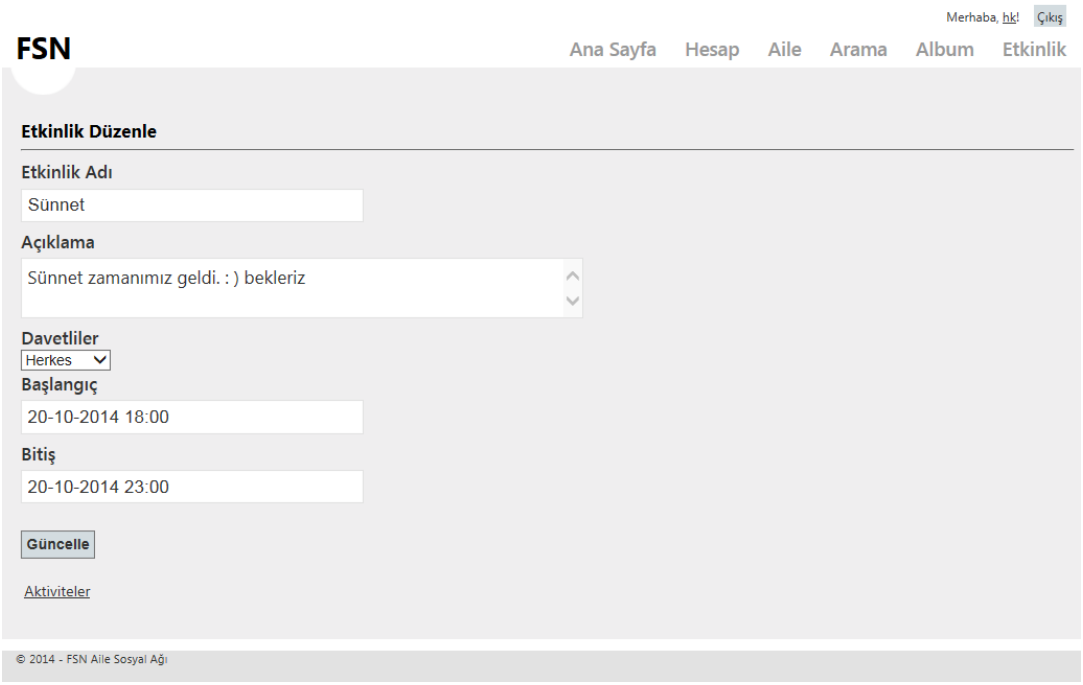
Bu ekranda kullanıcılar daha önceden oluşturdukları etkinlikleri listelenmiş bir şekilde görüntüleyebilir, silebilir veya düzenle linki ile değişiklik ekranına gidebilir. Yeni etkinlik linki ile etkinlik oluşturabilirler.



Şekil 4.34. Etkinlik yönetim ekranı

Etkinlik güncelleme ekranı Şekil 4.35'deki gibidir.

Bu ekranda kullanıcılar daha önceden oluşturdukları etkinlikleri ismini, açıklamasını, davet detayını ve tarih bilgilerini güncelleyebilirler.



Şekil 4.35. Etkinlik güncelleme ekranı

Etkinlik oluşturma ekranı Şekil 4.36'deki gibidir.

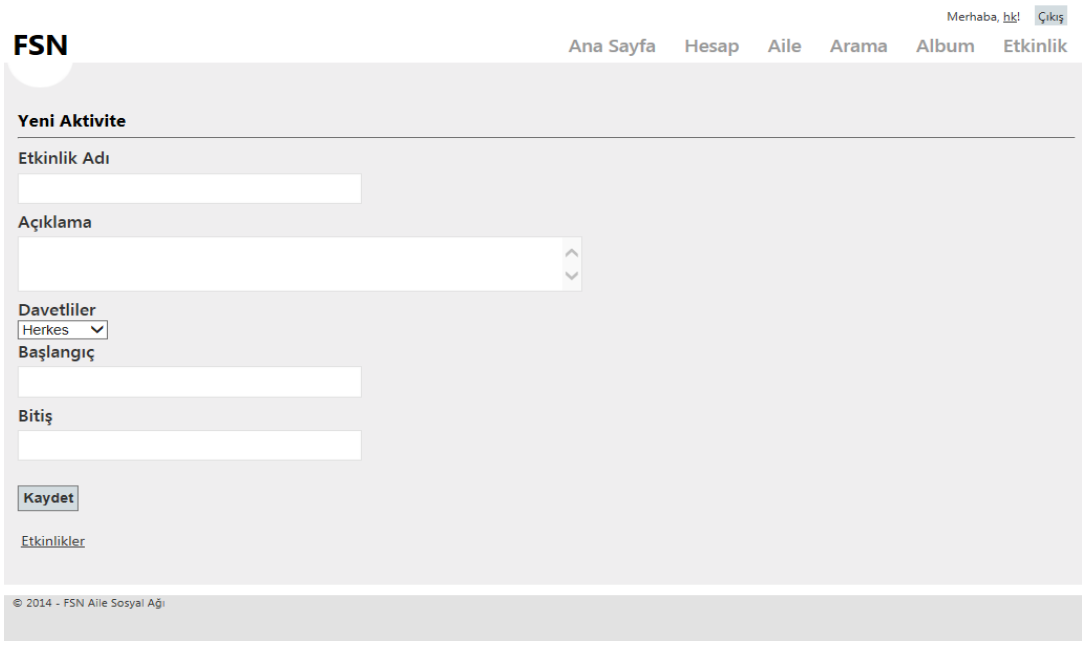
Bu ekranda kullanıcılar yeni düzenleyeceği etkinliklerini diğer kullanıcılara duyurmak için girişlerini yapabilirler ve davetli grubunu seçerek işlemleri gerçekleştirebilirler.

Etkinlik talep ekranı Şekil 4.37'deki gibidir.

Bu ekranda kullanıcılar davet edildikleri etkinlikleri listeleyebilir ve kabul veya ret işlemleri yapabilirler.

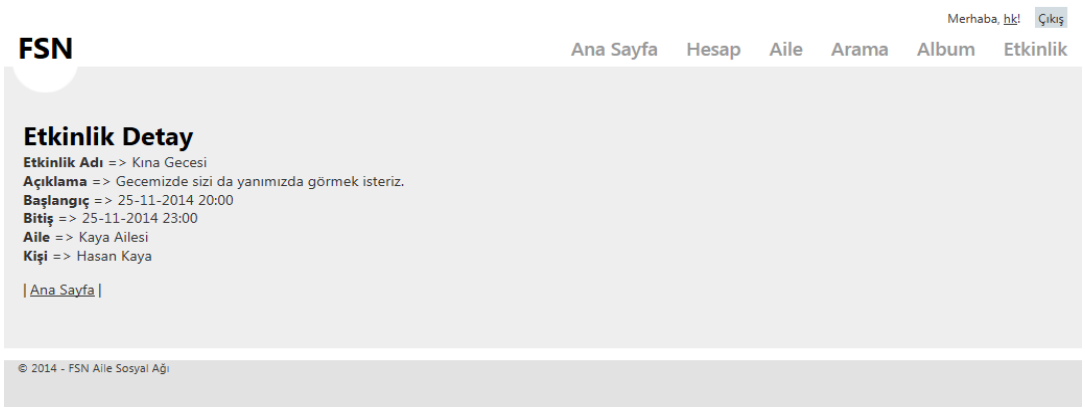
Etkinlik detay ekranı Şekil 4.38'deki gibidir.

Bu ekranda kullanıcılar ana sayfada görüntülenen ve yaklaşan etkinliklerin detaylarını görüntüleyebilirler.



Şekil 4.36. Etkinlik oluşturma ekranı

Şekil 4.37. Etkinlik talep ekranı



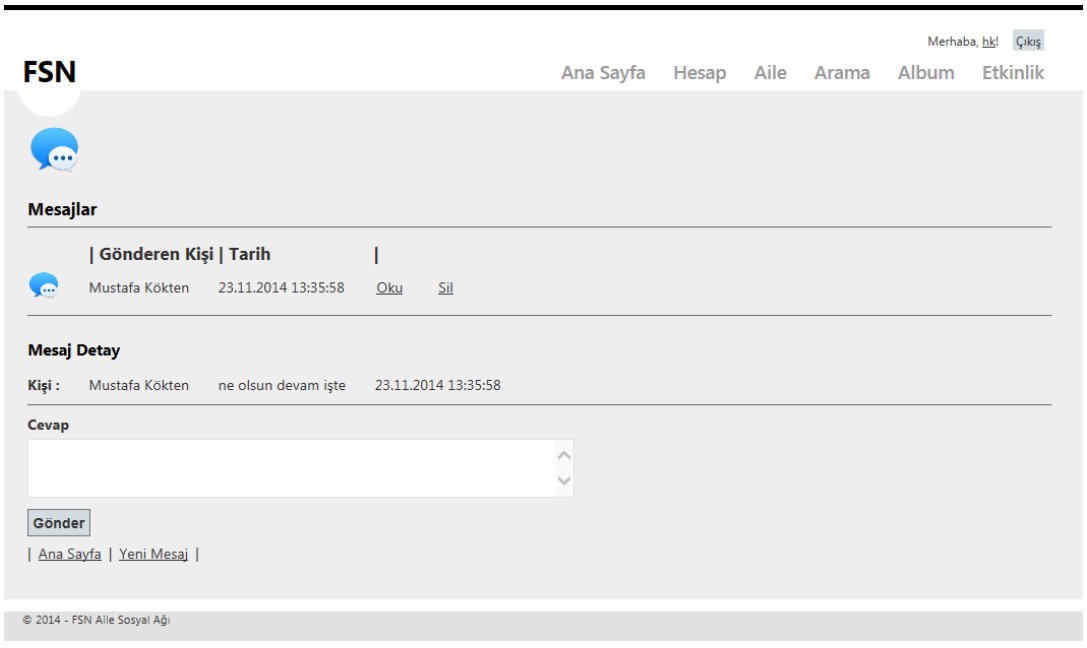
Şekil 4.38. Etkinlik detay ekranı

Mesaj yönetim ekranı Şekil 4.39'daki gibidir.

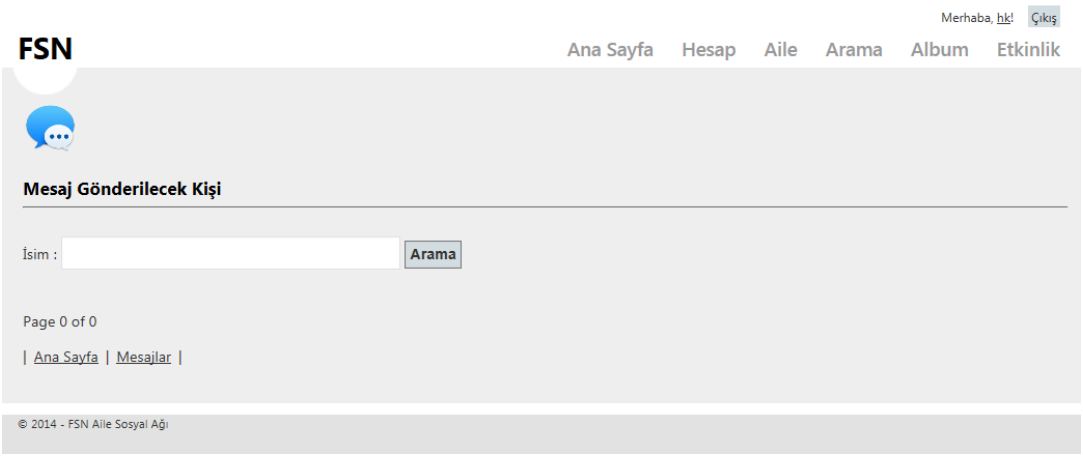
Bu ekranda kullanıcılar mesajlarını yönetebilirler, gelen mesajları okumak için listedeki "Oku" linkine tıklayarak aynı sayfada mesajlaşma detaylarını görüntüleyebilirler, cevap vermek için giriş yapıp gönderebilirler veya "Sil" linkini kullanarak eski mesajlarını silebilirler. Karşı tarafında mesajları okunma durumunu ise "Oku" linkinin durumundan anlayabilirler eğer mesajlar karşı taraftan okunmuş ile link normal fontla okunmamış ise kalın fontla görüntülenecektir. Mesaj listesinde olmayan bir kullanıcıya mesaj göndermek için "Yeni Mesaj" linkine tıklayarak işlem gerçekleştirebilirler.

Mesaj gönderim arama ekranı Şekil 4.40'daki gibidir.

Kullanıcılar bu ekranda mesaj göndermek istedikleri diğer kullanıcıları arayabilirler, kullanıcı listesi alt ekranda görünür ve mesaj göndermek istedikleri kullanıcı seçerek yeni bir mesajlaşma ekranı açabilirler.



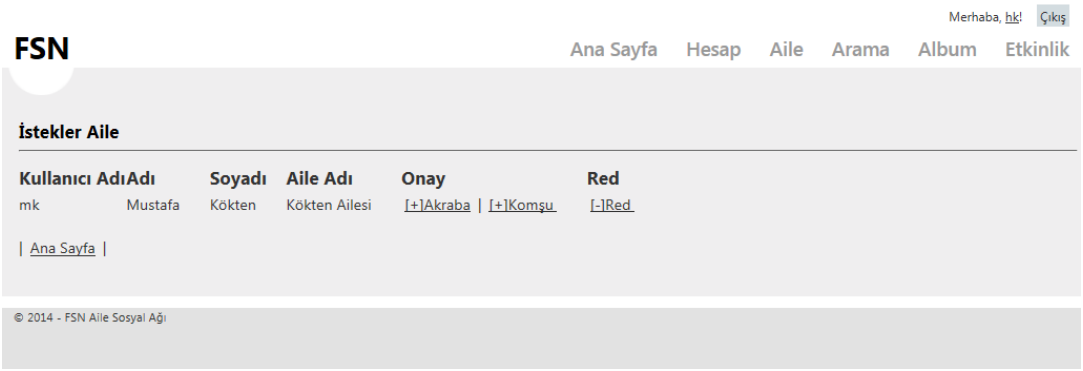
Şekil 4.39. Mesaj yönetim ekranı



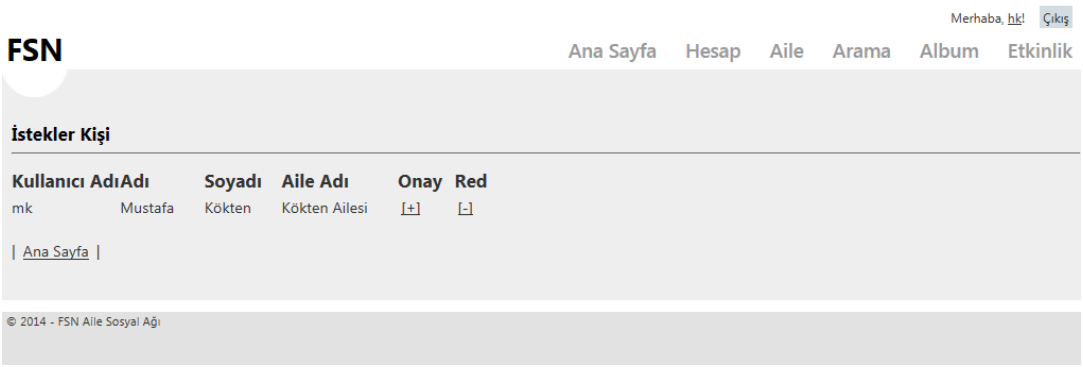
Şekil 4.40. Mesaj gönderim arama ekranı

İstek görüntüleme ve onay ekranları Şekil 4.41 ve Şekil 4.42'deki gibidir.

Bu ekranlar ana sayfadaki istekler alanında aile ve kişi talepleri linki üzerinden çağrılır, gelen talepler burada ekranlar üzerinden görüntülenir ve onay işlemleri gerçekleştirilir.



Şekil 4.41. Aile istek görüntüleme ve onay ekranı



Şekil 4.42. Kişi istek görüntüleme ve onay ekranı

5. SONUÇ

Hali hazırda kullanılmakta olan ve kaynaklar bölümünde listesi verilmiş kişisel veya aileye yönelik sosyal ağ siteleri incelenmiştir. Ailelere yönelik sosyal ağ sitelerinde genel yaklaşımın veri depolama alanı satışı için ortam oluşturmak ve buradan gelir elde etmeye yönelik olduğu gözlenmiştir. Bu sebeple de kısıtlı fonksiyonların bulunduğu, tam olarak bir paylaşım ve etkileşim ortamının bulunmadığı tespit edilmiştir. Aile için bireylerin resim, görüntü ve anlık düşünce paylaşımları dışında, aile içi etkinlik, aileler arası etkileşim, aile içi bireyler arası görevlendirme, çocuk gelişimi gibi süreçlere yönelik fonksiyonlara rastlanmamıştır. Bu sebeple aile için etkileşimi gün içinde de sanal ortamda devam ettirecek, diğer ailelerle güvenli ve kontrollü bir şekilde fikir ve veri paylaşımı yapabilecekleri, tam bir paylaşım ve etkileşim ortamı oluşturma amaçlanmaktadır.

KAYNAKLAR

Kitaplar

Dr. Ayfer ALPER(2012). *Sosyal Ağlar*. Pelikan Tıp Teknik Yayıncılık, Ankara

Necmi GÜRSAKAL(2009). *Sosyal Ağ Analizi*. Dora Basım Yayın, Bursa

İnternet

Utku Ozan Çankaya(2009). *UML Class Diyagramları* Erişim Tarihi: 15.08.2014, http://umlnedir.blogspot.com/

UML ile modelleme. Erişim Tarihi: 12.01.2014, http://univera-g.blogspot.com/2010/04/uml-ve-modelleme-bolum-9activity.html

UML. Erişim Tarihi: 13.08.2014, http://tr.wikipedia.org/wiki/Model-View-Controller

ASP.NET ve MSSQL. Erişim Tarihi: 20.08.2014, http://www.microsoft.com/

TTB(2012),*Yazılım test süreçleri* Erişim Tarihi: 21.08.2014, http://www.turkishtestingboard.org/

Keytorc(2014). *Yazılım test süreçleri* Erişim Tarihi: 21.08.2014, http://www.keytorc.com/

BA-Works(2014). *İş analizi ve analiz süreçleri* Erişim Tarihi: 21.08.2014, http://www.ba-works.com/

Tezler Erişim Tarihi: 24.08.2014, http://tez.yok.gov.tr/

Microsoft (2014). *MVC* Erişim Tarihi: 24.02.2014, http://msdn.microsoft.com/en-us/library/ff649643.aspx

Rick Anderson (2012). *ASP.NET MVC 4*, Erişim Tarihi: 24.03.2014, http://www.asp.net/mvc/overview/older-versions/getting-started-with-aspnet-mvc4/intro-to-aspnet-mvc-4

Facebook(2014). *Kişisel sosyal ağ*. Erişim Tarihi: 26.11.2014, www.facebook.com

Twitter(2014). *Mikro blog sitesi*. Erişim Tarihi: 26.11.2014, www.twitter.com

Linkedin(2014). *İş amaçlı profesyonel sosyal ağ*. Erişim Tarihi: 26.11.2014, www.linkedin.com

MyFamily(2014). *Aile sosyal ağ*. Erişim Tarihi: 26.11.2014, www.myfamily.com

EFamily(2014). *Aile sosyal ağ* Erişim Tarihi: 26.11.2014, https://efamily.com

Tezler

Yaylagül O. (2010). *Web tabanlı bilgi paylaşım platformu*, Yüksek Lisans Tezi. Haliç Üniversitesi. Fen Bilimleri Enstitüsü, İstanbul.

Önder Ö. (2010). *Mimarlara yönelik bir sosyal ağ sitesinin tasarımı ve geliştirilmesi.* Yüksek Lisans Tezi. Haliç Üniversitesi. Fen Bilimleri Enstitüsü, İstanbul.

Sarı E. (2007). *C# ile alışveriş-ticari portal tasarımı.* Yüksek Lisans Tezi. Haliç Üniversitesi. Fen Bilimleri Enstitüsü, İstanbul.

İnce Türker G. (2009). *E-ticaret amaçlı bir web sitesinin tasarlanması (Örnek bir uygulama).* Yüksek Lisans Tezi. Afyon Kocatepe Üniversitesi. Fen Bilimleri Enstitüsü, Afyon.

Çimeliler A. (2008). *Bilgi ve erişim amaçlı bir tekstil web sitesinin tasarımı ve uygulaması.* Yüksek Lisans Tezi. Çukurova Üniversitesi. Fen Bilimleri Enstitüsü, Adana.

EKLER

EK 1: Kullanıcı Karşılama Ana Ekranı Kaynak Kodları

MainController.cs

```
using System;
using System.Web.Mvc;
using FSNApp.Models;
using FSNApp.GenF;
using FSNApp.Controllers;
using WebMatrix.WebData;
using System.Linq;
using System.Collections.Generic;
using FSNApp.Models.DBSet;
using FSNApp.Models.Classes;

namespace FSNApp.Controllers
{
    [Authorize]
    public class MainController : Controller
    {
        private MainModel mm = new MainModel();
        private AkisLogContext ak = new AkisLogContext();
        private FMemberContext fm = new FMemberContext();
        private FInteractionContext fi = new FInteractionContext();
        private ResDb Al = new ResDb();
        private EventDbContext EvDb = new EventDbContext();
        private MessageDb Mdb = new MessageDb();
        Kimlik HMidKimlik1;
        string OFtIdName1;
        string FDtIdName1;
        public ActionResult Index()
        {
            mm.PicPath = GenFunc.GetUsrPic(WebSecurity.GetUserId(User.Identity.Name) , 1);
            mm.FPicPath = GenFunc.GetUsrPic(GenFunc.GetFmId(User.Identity.Name), 2);
            if (GenFunc.GetUserName(User.Identity.Name, 0).adi != null)
            {
                mm.Adi = GenFunc.GetUserName(User.Identity.Name, 0).adi + ' ' + GenFunc.GetUserName(User.Identity.Name,
0).soyadi;
            }
            else
                mm.Adi = null;
            if (GenFunc.GetFmNm(1, User.Identity.Name, 0) != null)
            {
                mm.AileAdi = GenFunc.GetFmNm(1, User.Identity.Name, 0) + '(' +
GenFunc.GetFmCount(GenFunc.GetFmId(WebSecurity.CurrentUserName), 4) + ')';
            }
            else
                mm.AileAdi = null;
            mm.Talep = GenFunc.GetFmCount(WebSecurity.GetUserId(User.Identity.Name) , 1);
            mm.Komsu = GenFunc.GetFmCount(WebSecurity.GetUserId(User.Identity.Name), 2);
            mm.KTalep = GenFunc.GetFmCount(WebSecurity.GetUserId(User.Identity.Name), 3);
            int fmid = GenFunc.GetFmId(User.Identity.Name);
            int uid = WebSecurity.CurrentUserId;
            mm.Mesajs = (from tmp in Mdb.Messages
                         where tmp.RID == uid && tmp.Status == false
                         select tmp).Count();
            #region iliskiliaileler
            List<int> fiaile = new List<int>();
            var aileler = (from tmp in fi.FInteractiondbs
                           where (tmp.MFid == fmid || tmp.SFid == fmid) && tmp.level == 2
                           select tmp).ToList();
            var aileler1 = (from tmp in fm.FMembers
                            where tmp.UserId == WebSecurity.CurrentUserId
                            select tmp).ToList();
            if (aileler.Count > 0 && fmid>0) /*Aile nosu olan kullanıcı için ilişkili aileler*/
            {
                foreach (var item in aileler)
                {
                    if (item.MFid == fmid)
                        fiaile.Add(item.SFid);
                    else
                        fiaile.Add(item.MFid);
                }
            }

            if (fiaile.Count == 0 && fmid == 0) /*Aile nosu olmayan kullanıcı için ilişkili aileler*/
            {
                foreach (var item in aileler1)
                {
                    fiaile.Add(item.FID);
                }
            }
```

```
            if (aileler.Count==0 && aileler1.Count==0)
                fiaile.Add(0);
            #endregion
            #region Loglar
            var AkisLogTemp = (from temp in ak.AkisLogDbs
                               where (temp.FDtId == fmid || fiaile.Contains(temp.FDtId)) || (temp.HMIdt == 3 && (temp.HMid
== fmid || fiaile.Contains(temp.HMid)))
                               select temp).Take(10);
            var LogV = new List<LogViewModel>();
            if (AkisLogTemp.Count() > 0)
            {
                foreach (var item in AkisLogTemp)
                {
                    if (item.HId == 5)/*Albüm izin kontrolü*/
                    {
                        var kont = (from temp in Al.Albums
                                    where temp.ID == item.HId
                                    select temp).ToList();
                        if (kont.Count>0)
                        if ((kont[0].SecLev == 1) && (kont[0].AUserID != WebSecurity.CurrentUserId))
                        {
                            continue;
                        }
                    }
                    if (item.HMIdt == 1)
                    {
                        FMember fmb = fm.FMembers.Find(item.HMid);
                        if (fmb != null)
                        {
                            OFtIdName1 = fmb.Name + ' ' + fmb.LastName;
                        }
                    }
                    else
                        OFtIdName1 = null;
                    if (item.HMIdt == 2)
                    {
                        HMidKimlik1 = GenFunc.GetUserName(null, item.HMid);
                        OFtIdName1 = HMidKimlik1.adi + ' ' + HMidKimlik1.soyadi;
                        HMidKimlik1 = null;
                    }
                    if (item.HMIdt == 3)
                        OFtIdName1 = GenFunc.GetFmNm(2, null, item.HMid);
                    FDtIdName1 = GenFunc.GetFmNm(2, null, item.FDtId);
                    LogV.Add(new LogViewModel
                    {
                        FDtId = item.FDtId,
                        FDtIdName = FDtIdName1,
                        Ack = item.Ack,
                        HDate = item.HDate,
                        HId = item.HId,
                        HMid = item.HMid,
                        HMIdt = item.HMIdt,
                        OFtIdName = OFtIdName1,
                        HMidKimlik = HMidKimlik1
                    });
                }
                    }
                    else
                    {
                    LogV.Add(new LogViewModel
                        {
                        FDtId = 0,
                        FDtIdName = "Şimdilik temiz bir sayfa ;)",
                        Ack = "",
                        HId = 0,
                        HMid = 0,
                        HMIdt = 0,
                        OFtIdName = "",
                        HMidKimlik = null
                        });
                    }
            ViewData["LogKayit"] = LogV;
            #endregion
            #region DogumGunleri
            var LogD = new List<DogumGModel>();
            var DogumGl = (from tmp in fm.FMembers
                           where tmp.FID== fmid
                           select tmp).ToList();
            if (DogumGl.Count > 0)
            {
                foreach (var item in DogumGl)
                {
                    if (DateTime.Now.Month == item.Bdate.Month)
                        if ((DateTime.Now.Day - item.Bdate.Day) >= -5)
                            if ((DateTime.Now.Day - item.Bdate.Day) <= 0)
                            {
                                LogD.Add(new DogumGModel
                                        {
                                            adi = item.Name,
                                            Soyadi = item.LastName,
                                            tarih = item.Bdate
                                        });
                            }
                }
            }
            if (LogD.Count==0)
            {
                LogD.Add(new DogumGModel
                {
```

```
                adi = "Kayıt yok",
                Soyadi = null
            });
        }
        ViewData["DGKayit"] = LogD;
        #endregion
        #region Event
        mm.Etkinlik = 0;
        /*Etkinlikler listesi*/
        var EvtTmp = (from temp in EvDb.Events
                      where fiaile.Contains(temp.EFID) && temp.Status == 1 && temp.EETime > DateTime.Now && temp.SecLev
>= 2
                      select temp).ToList();

        var EvtTmpB = (from temp in EvDb.Events
                       where temp.EFID == fmid && temp.Status == 1 && temp.EETime > DateTime.Now && temp.SecLev >= 2
                       select temp).ToList();
        /*Katılım kontrolü*/
        var LogEv = new List<Event>();
        var LogEvn = new List<Event>();

        foreach (var item in EvtTmp)
        {
            if ((from tm in EvDb.EventMembers
                 where tm.EventID == item.ID && tm.EMFID == fmid
                 select tm).Count() > 0)
            {
                LogEv.Add(new Event
                {
                    EAck = item.EAck,
                    EETime = item.EETime,
                    EFID = item.EFID,
                    EName = item.EName,
                    ESTime = item.ESTime,
                    EUID = item.EUID,
                    EventMembers = null,
                    ID = item.ID,
                    PicPath = null,
                    SecLev = item.SecLev,
                    Status = item.Status
                });
            }
            else
            {
                LogEvn.Add(new Event
                {
                    EAck = item.EAck,
                    EETime = item.EETime,
                    EFID = item.EFID,
                    EName = item.EName,
                    ESTime = item.ESTime,
                    EUID = item.EUID,
                    EventMembers = null,
                    ID = item.ID,
                    PicPath = null,
                    SecLev = item.SecLev,
                    Status = item.Status
                });
            }
        };
        #region Benim aile etkinliklerim
        foreach (var item in EvtTmpB)
        {
            LogEv.Add(new Event
            {
                EAck = item.EAck,
                EETime = item.EETime,
                EFID = item.EFID,
                EName = item.EName,
                ESTime = item.ESTime,
                EUID = item.EUID,
                EventMembers = null,
                ID = item.ID,
                PicPath = null,
                SecLev = item.SecLev,
                Status = item.Status
            });
        }
        #endregion
        mm.Etkinlik = LogEvn.Count();
        if (LogEv.Count == 0)
        {
            LogEv.Add(new Event
                {
                    EAck = null,
                    EETime = System.DateTime.Now,
                    EFID = 0,
                    EName = "Kayıt yok",
                    ESTime = System.DateTime.Now,
                    EUID = 0,
                    EventMembers = null,
                    ID = 0,
                    PicPath = null,
                    SecLev = 0,
                    Status = 0
                });
        }
        ViewData["EVKayit"] = LogEv;
        #endregion
```

```
            return View(mm);
        }
        public ActionResult ERequest()
        {
            int fmid = GenFunc.GetFmId(User.Identity.Name);
            List<int> fiaile = new List<int>();
            var aileler = (from tmp in fi.FInteractiondbs
                           where (tmp.MFid == fmid || tmp.SFid == fmid) && tmp.level == 2
                           select tmp).ToList();
            if (aileler.Count > 0)
            {
                foreach (var item in aileler)
                {
                    if (item.MFid == fmid)
                        fiaile.Add(item.SFid);
                    else
                        fiaile.Add(item.MFid);
                }
            }
            else
                fiaile.Add(0);
            var EvtTmp = (from temp in EvDb.Events
                          where fiaile.Contains(temp.EFID) && temp.Status == 1 && temp.EETime > DateTime.Now && temp.SecLev
>= 2
                          select temp).ToList();
            /*Katılım kontrolü*/
            var LogEv = new List<EventV>();
            var LogEvn = new List<EventV>();
            Event Evnn = new Event();
            Kimlik Km;
            #region Başka ailelerin etkinlikleri
            foreach (var item in EvtTmp)
            {
                Evnn.EAck = item.EAck;
                Evnn.EETime = item.EETime;
                Evnn.EFID = item.EFID;
                Evnn.EName = item.EName;
                Evnn.ESTime = item.ESTime;
                Evnn.EUID = item.EUID;
                Evnn.EventMembers = null;
                Evnn.ID = item.ID;
                Evnn.PicPath = null;
                Evnn.SecLev = item.SecLev;
                Evnn.Status = item.Status;

                if ((from tm in EvDb.EventMembers
                     where tm.EventID == item.ID && tm.EMFID == fmid
                     select tm).Count() > 0)
                {
                    Km = GenF.GenFunc.GetUserName(null, item.EUID);
                    LogEv.Add(new EventV
                    {
                        Ev = Evnn,
                        Aile = GenF.GenFunc.GetFmNm(2,null,item.EFID),
                        Kisi = Km.adi + ' ' + Km.soyadi
                    });
                }
                else
                {
                    Km = GenF.GenFunc.GetUserName(null,item.EUID);
                    LogEvn.Add(new EventV
                    {
                        Ev = Evnn,
                        Aile = GenF.GenFunc.GetFmNm(2,null,item.EFID),
                        Kisi = Km.adi+' '+Km.soyadi
                    });
                }
            };
            #endregion
            return View(LogEvn);
        }
        public ActionResult EAccept(int Id, bool Dr)
        {
            EventMember Evn = new EventMember();
            Evn.status = Dr;
            Evn.EventID = Id;
            Evn.EMFID = GenFunc.GetFmId(WebSecurity.CurrentUserName);
            EvDb.EventMembers.Add(Evn);
            EvDb.SaveChanges();
            return RedirectToAction("ERequest");
        }
        public ActionResult EDetail(int Id)
        {
            EventV Evnv = new EventV();
            Kimlik Km = new Kimlik();
            Evnv.Ev = EvDb.Events.Find(Id);

            if (Evnv != null)
            {
                Km = GenFunc.GetUserName(null, Evnv.Ev.EUID);
                Evnv.Kisi = Km.adi + " " + Km.soyadi;
                Evnv.Aile = GenFunc.GetFmNm(2, null, Evnv.Ev.EFID);
                return View(Evnv);
            }
            return View(Evnv);
        }
        public ActionResult AView(int? Id)
        {
            ResDb _db = new ResDb();
```

```
            if (Id.HasValue)
            {
                return View((from temp in _db.AlbumDtys
                             where temp.AlbumId == Id.Value
                             select temp).ToList());
            }
            else
                return View();
        }
        public ActionResult HesapYon(string Duzenle , FormCollection Fc)
        {
            if (Duzenle == "Parola")
            {
                return RedirectToAction("Manage", "Account");
            }
            else if (Duzenle == "Aile")
            {
                return RedirectToAction("Details", "FamilyD");
            }
            else if (Duzenle == "Kullanici")
            {
                return RedirectToAction("UserDetail", "Account");
            }
            return View();
        }
        protected override void Dispose(bool disposing)
        {
            if (ak != null)
                ak.Dispose();
            if (fm != null)
                fm.Dispose();
            if (fi != null)
                fi.Dispose();
            if (Al != null)
                Al.Dispose();
            if (Mdb != null)
                Mdb.Dispose();
            if (EvDb != null)
                EvDb.Dispose();
        }
    }
}
```

Index.cshtml

```
@model FSNApp.Models.MainModel
@{
    IEnumerable<FSNApp.Models.LogViewModel> sidebarLog = ViewData["LogKayit"] as IEnumerable<FSNApp.Models.LogViewModel>;
    IEnumerable<FSNApp.Models.DogumGModel> sidebardg = ViewData["DGKayit"] as IEnumerable<FSNApp.Models.DogumGModel>;
    IEnumerable<FSNApp.Models.Event> sidebarEv = ViewData["EVKayit"] as IEnumerable<FSNApp.Models.Event>;
}
@{
    ViewBag.Title = "Kullanıcı Ana Sayfası";
    Layout = "~/Views/Shared/BaseLayout.cshtml";
}
@Styles.Render("~/Content/css")
@Scripts.Render("~/bundles/modernizr")
<table >
<tr>
<td >
<table >
 <tr>
     <td style="text-align:left" bgcolor="#FFFFFF">
         @if (Model.PicPath != null && Model.PicPath != "")
         {
             <div>
                 <img src="/Pictures/MemberPics/@Model.PicPath" width="100" height="120" />
             </div>
         }

         <div class="display-field">
             @if (Model.Adi == null)
     {
     <b>Sayfa sahibi detayı yok.</b>
         } else
         {
     <text>@Html.DisplayFor(model => model.Adi) </text>
         }
         </div>
         <hr />
         <div>
             @if (Model.AileAdi == null)
     {
     <b>Aile Bilgisi detayı yok.</b>
         }
         else
         {
     <text>@Html.DisplayFor(model =>model.AileAdi)</text>
         }
         </div>
         @if (Model.FPicPath != null && Model.FPicPath != "")
         {
             <div>
                 <img src="/Pictures/FamilyPics/@Model.FPicPath" width="100" height="120" />
             </div>
         }
         <hr />
         <div class="display-field">
             @if (Model.Komsu > 0)
```

```
            {<b>@Html.ActionLink("Komşular(" + @Html.DisplayFor(model => model.Komsu) + ")", "NIndex", "FSearch")</b>}
            else
            {<text>Komşular(0)</text>}
        </div>
        <hr />
        <b>İstekler</b>
        <hr />
        <div class="display-field">
            @if (Model.Talep > 0)
            {@Html.ActionLink("Aile => (" + @Html.DisplayFor(model => model.Talep) + ")", "Index", "Request")}
            else
            {<div>Aile=> 0</div>}
        </div>
        <div class="display-field">
            @if (Model.KTalep > 0)
            {@Html.ActionLink("Kişi => (" + @Html.DisplayFor(model => model.KTalep) + ")", "Index", "KRequest")}
            else
            {<div>Kişi=> 0</div>}
        </div>
        <div class="display-field">
            @if (Model.Etkinlik > 0)
            {@Html.ActionLink("Etkinlik => (" + @Html.DisplayFor(model => model.Etkinlik) + ")", "ERequest")}
            else
            {<div>Etkinlik=> 0</div>}
        </div>
        <div>
            @Html.Partial("DogumGPartial", sidebardg)
        </div>
        <div>
            @Html.Partial("EventPartial", sidebarEv)
        </div>
        <div>
            <hr />
            @if(Model.Mesajs==0)
            {@Html.ActionLink("Mesajlar (" + @Model.Mesajs + ")", "Index","Messages" )}
            else
            {<b>@Html.ActionLink("Mesajlar (" + @Model.Mesajs + ")", "Index" , "Messages" )</b> }
            <hr />
        </div>
</tr>
</table>
</td>
<td>
<table >
    <tr >
        <td bgcolor="#FFFFFF">
            @Html.Partial("AkisPartial", sidebarLog)
        </td>
    </tr>
</table>
</td>
</tr>
</table>
```

EK 2: Aile İşlemleri Ekranı Kaynak Kodları

FMemberController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using FSNApp.Models;
using WebMatrix.WebData;
using FSNApp.GenF;
using System.Data;
using System.Data.Entity;

namespace FSNApp.Controllers
{
    [Authorize]
    public class FMemberController : Controller
    {
        private FMemberContext fmdb = new FMemberContext();
        private FMemberContext fmdb1 = new FMemberContext();
        private FamilyDetailsContext fdb = new FamilyDetailsContext();
        private FamilyRoleModel frdb = new FamilyRoleModel();
        private FInteractionContext Fidb = new FInteractionContext();
        private AkisLogContext AklDb = new AkisLogContext();
        private int UID = new int();
        private int FID = new int();
        private int FMk = new int();
        int fmid = GenFunc.GetFmId(WebSecurity.CurrentUserName);
        // GET: /FMember/
        public ActionResult Index()
        {
            var model = fmdb.FMembers.ToList();
            var fddb = fdb.FamilyDetaildbs.ToList();
            ViewBag.FFMID = fmid;
            UID = WebSecurity.CurrentUserId;
            fddb = (from tempid in fdb.FamilyDetaildbs
                    where tempid.UserId == UID
                    select tempid).ToList();
            if (fddb.Count>0)
                FID = fddb[0].FDtId;
            model = (from temp in fmdb.FMembers
                         where temp.FID == FID
                         orderby temp.FMemberId
                         select temp).ToList();
            List<FTreeModel> FTM = new List<FTreeModel>();
            if (model.Count>0)
            {
                ViewBag.Kon = true;
                foreach (var item in model)
                {
                    FTM.Add(new FTreeModel
                    {
                        FMemberId=@item.FMemberId,
                        Name=@item.Name,
                        LastName=@item.LastName,
                        Type = @item.Type,
                        FMemberLv =(from temp in frdb.FamilyRoledbs
                                    where temp.FRid==@item.FMemberId
                                    select temp.TreeId).FirstOrDefault()
                    });
                }
                Kimlik Km = GenF.GenFunc.GetUserName(null, UID);
                if (Km.cins == "E")
                    FMk = 1;
                else
                    FMk = 2;
                FTM.Add(new FTreeModel
                {
                    FMemberId = FMk,
                    Name = Km.adi,
                    LastName = Km.soyadi,
                    Type = Km.cins,
                    FMemberLv = 1
                });
                ViewData["AileKayit"] = FTM.OrderBy(x => x.FMemberLv).ToList();
                return View(model);
            }
            ViewBag.Kon = false;
            return View();
        }
        // POST: /FMember/Create
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create(FMember fmemberdb, string Cins, int Yakinlik)
        {
                if (ModelState.IsValid)
                {
                    fmemberdb.FID = GenFunc.GetFmId(User.Identity.Name);
                    fmemberdb.UserFID = 0;
                    fmemberdb.FamilyM = false;
                    fmemberdb.FOk = true;
                    fmemberdb.FOkUserId = WebSecurity.CurrentUserId;
                    fmemberdb.FokDate = DateTime.Now;
                    fmemberdb.FMemberId = 0;
                    fmemberdb.UserId = 0;
```

```
            fmemberdb.Type = Cins;
            fmemberdb.FMemberId = Yakinlik;
            #region Resim Ekleme
            if (Request.Files.Count > 0)
            {
                string DosyaAdi = Guid.NewGuid().ToString().Replace("-", "");
                string uzanti = System.IO.Path.GetExtension(Request.Files[0].FileName);
                string TamYolYeri = "~/Pictures/FamilyPics/" + DosyaAdi + uzanti;
                Request.Files[0].SaveAs(Server.MapPath(TamYolYeri));
                if (uzanti != "")
                    fmemberdb.PicPath = DosyaAdi + uzanti;
            }
            #endregion
            fmdb.FMembers.Add(fmemberdb);
            fmdb.SaveChanges();
            BatchF.LogKayit(fmemberdb ,null, 1 , null);
            return RedirectToAction("Index");
        }
        return View();
    }
// GET: /FMember/Edit/5
public ActionResult Edit(int id)
{
    FMember fmemberdb = fmdb.FMembers.Find(id);
    if (fmemberdb == null)
    {
        return RedirectToAction("Index");
    }
    return View(fmemberdb);
}
// GET: /FMember/Edit/5
public ActionResult Create()
{
    return View();
}
// POST: /FMember/Edit/5
[HttpPost]
public ActionResult Edit(FMember fmemberdb, string Cins, int Yakinlik)
{
    fmemberdb.Type = Cins;
    fmemberdb.FMemberId = Yakinlik;
    #region Resim Ekleme
    if (Request.Files.Count > 0)
    {
        string DosyaAdi = Guid.NewGuid().ToString().Replace("-", "");
        string uzanti = System.IO.Path.GetExtension(Request.Files[0].FileName);
        string TamYolYeri = "~/Pictures/FamilyPics/" + DosyaAdi + uzanti;
        Request.Files[0].SaveAs(Server.MapPath(TamYolYeri));
        if (uzanti != "")
            fmemberdb.PicPath = DosyaAdi + uzanti;
    }
    #endregion
    try
    {
        if (ModelState.IsValid)
        {
            fmdb.Entry(fmemberdb).State = EntityState.Modified;
            fmdb.SaveChanges();
        }
        return RedirectToAction("Index");
    }
    catch
    {
        return View();
    }
}
public ActionResult Delete(int id)
{
    FMember fmemberdb = fmdb.FMembers.Find(id);
    int Usr = WebSecurity.CurrentUserId;
    if (fmemberdb != null)
    {
        var rrno = (from s in Fidb.FInteractiondbs
                    where s.MUserId == WebSecurity.CurrentUserId && s.FUserId == fmemberdb.UserId
                    && s.level == 2 && s.IntLev == 3
                    select s.FIID).ToList();
        if (rrno.Count > 0)
        {
            FInteractiondb fd = Fidb.FInteractiondbs.Find(rrno[0]);
            Fidb.Entry(fd).State = EntityState.Deleted;
            Fidb.SaveChanges();
        }
        BatchF.LogKayit(fmemberdb, null, 2,null);
        fmdb.Entry(fmemberdb).State = EntityState.Deleted;
        fmdb.SaveChanges();
    }
    return RedirectToAction("Index");
}
protected override void Dispose(bool disposing)
{
    if (fmdb != null)
        fmdb.Dispose();
    if (fmdb1 != null)
        fmdb1.Dispose();
    if (fdb != null)
        fdb.Dispose();
    if (frdb != null)
        frdb.Dispose();
    if (Fidb != null)
```

```
                Fidb.Dispose();
            if (AklDb != null)
                AklDb.Dispose();
            base.Dispose(disposing);
        }
    }
}
```

Index.cshtml

```
@model IEnumerable<FSNApp.Models.FMember>
@{
    IEnumerable<FSNApp.Models.FTreeModel> Treebar = ViewData["AileKayit"] as IEnumerable<FSNApp.Models.FTreeModel>;
}
@{
    ViewBag.Title = "Aile Listesi";
    Layout = "~/Views/Shared/BaseLayout.cshtml";
}
<img src="/Pictures/icons/home.png" width="60" height="60" />
<h3>Aile Üyeleri</h3><hr />
@if (ViewBag.Kon)
{
<table class="table">
     <tr>
        <th>
            @Html.DisplayNameFor(model => model.Name) |
        </th>
        <th>
            @Html.DisplayNameFor(model => model.LastName) |
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Bdate) |
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Type) |
        </th>
        <th>
            @Html.DisplayNameFor(model => model.FMemberId)
        </th>
    </tr>
@foreach (var item in Model)
{
     <tr>
        <td>
        @Html.DisplayFor(model => item.Name)
        </td>
        <td>
        @Html.DisplayFor(model => item.LastName)
        </td>
        <td>
        @Html.DisplayFor(model => item.Bdate)
        </td>
        <td>
        @if (item.Type=="E")
        {<text>Erkek</text>}
        @if (item.Type=="K")
        {<text>Kadın</text>}
        </td>
        <td>
        @switch (item.FMemberId)
        {
        case 1 : <text>Baba</text>
            break;
        case 2: <text>Anne</text>
            break;
        case 3: <text>Kız Evlat</text>
            break;
        case 4: <text>Erkek Evlat</text>
            break;
        case 5: <text>Amca</text>
            break;
        case 6: <text>Dayı</text>
            break;
        case 7: <text>Teyze</text>
            break;
        case 8: <text>Hala</text>
            break;
        case 9: <text>Kuzen</text>
            break;
        case 10: <text>Büyük Anne</text>
            break;
        case 11: <text>Büyük Baba</text>
            break;
        case 12: <text>Dede</text>
            break;
        case 13: <text>Nine</text>
            break;
        case 14: <text>Torun</text>
            break;
        }
        </td>
        <td>
        @if (item.PicPath != null && item.PicPath!="")
        {
        <img src="/Pictures/FamilyPics/@item.PicPath" width="50" height="60" />
        }
        </td>
        <td>
        @if (item.UserId == 0 )
```

```
        {
        @Html.ActionLink("Düzenle", "Edit", new { id = @item.FMid })
        }
        </td>
        <td>
        @Html.ActionLink("Sil", "Delete", new { id = @item.FMid })
        </td>
    </tr>
}
</table>
}
else
{
<b>Aile Üye Bilgisi yok.</b>
}
<br />
<div>
    |
    @if(ViewBag.FFMID>0)
    {@Html.ActionLink("Ekle", "Create", "FMember") }|
    @Html.ActionLink("Ana Sayfa", "Index", "Main") |
</div>
<center>
    @if (ViewBag.Kon)
    {
        <h2>Aile Ağacı</h2>
        <div>
            @Html.Partial("FamilyTree",Treebar)
        </div>
    }
</center>
```

EK 3: Arama ve Bağlantı Ekranı Kaynak Kodları

FSearchController.cs

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Web.Mvc;
using PagedList;
using FSNApp.Models;
using FSNApp.GenF;
using WebMatrix.WebData;

namespace FSNApp.Controllers
{
    [Authorize]
    public class FSearchController : Controller
    {
        // GET: /FSearch/
        private FamilyDetailsContext fddb = new FamilyDetailsContext();
        private UsersContext usrdb = new UsersContext();
        private FInteractionContext fidb = new FInteractionContext();
        private Kimlik km = new Kimlik();
        private FSCard Fsc = new FSCard();
        public ActionResult Index(string currentFilter, string searchString,
        string Secim ,int? page, int? isl, int? fid ,int? uid ,int? did )
        {
            ViewBag.Secim = Secim;
            if (isl.HasValue)
            {
            if (isl.Value == 1)  /*Aile ilişki talebi*/
                interaction(fid,uid, null);
            if (isl.Value == 2) /* İlişki Red/Silme*/
                intDel(did, 1);
            if (isl.Value == 3)  /*Aile ilişki talebi*/
                interaction(fid, uid, 3);
            }
            var FSearhFs = new List<FSearchF>();
            int pageSize = 10;
            int pageNumber = (page ?? 1);
            int Susr = GenFunc.GetFmId(User.Identity.Name);
            if ((searchString != null) || (Secim =="Komşular"))
            {
                page = 1;
            }
            else
            {
                searchString = currentFilter;
            }
            ViewBag.CurrentFilter = searchString;
            #region Aile
            if (Secim == "Aile")
            {
                if (!String.IsNullOrEmpty(searchString))
                {
                    var Familys = (from s in fddb.FamilyDetaildbs
                                   where s.FamilyName.ToUpper().Contains(searchString.ToUpper())
                                   select s).ToList();
                    for (int i = 0; i <= Familys.Count()-1; i++)
                    {
                        int gusr = Familys[i].UserId;
                        int FDt = Familys[i].FDtId;
                        int ILevel;
                        int Rno;
                        int IIntLev;
                        if (gusr != WebSecurity.CurrentUserId)
                        {
                        var gelen = (from d in usrdb.UserProfiles
                                     where d.UserId == gusr
                                     select new { d.Adi, d.Soyadi, d.UserName }).ToList();
                        // Aileler arası önceden girilmiş bir ilişki var mı?
                        var Ailekon = (from f in fidb.FInteractiondbs
                                       where (f.SFid == Susr && f.MFid == FDt) ||
                                       (f.SFid == FDt && f.MFid == Susr)
                                       select f).ToList();
                        if (Ailekon.Count > 0)
                        {
                            ILevel = 0;
                            if (Ailekon[0].IntLev == 3) /*Kişisel*/
                            {
                                switch (Ailekon[0].level)
                                {
                                    case 1:
                                        ILevel = 3;
                                        break;
                                    case 2:
                                        ILevel = 4;
                                        break;
                                }
                            }
                            if (Ailekon[0].level == 1 && Ailekon[0].IntLev == 0) /*Teklif*/
                                ILevel = 1;
                            if (Ailekon[0].level == 2 && (Ailekon[0].IntLev == 1 || Ailekon[0].IntLev == 2)) /*Teklif*/
                                ILevel = 2;
                                Rno = Ailekon[0].FIID;
```

```
                IIntLev = Ailekon[0].IntLev;
            }
            else
            {
                ILevel = 0;
                Rno = 0;
                IIntLev = 0;
            }
            if (gelen.Count > 0)
            {
                FSearhFs.Add(new FSearchF
                {
                    Tid = Rno,
                    Adi = gelen[0].Adi,
                    FamilyName = Familys[i].FamilyName,
                    FDtId = Familys[i].FDtId,
                    Soyadi = gelen[0].Soyadi,
                    UserId = gusr,
                    UserName = gelen[0].UserName,
                    IntLev = IIntLev,
                    akkon = ILevel
                });
            }
            }
        }

        return View(FSearhFs.ToPagedList(pageNumber, pageSize));
        }
    }
    #endregion
    #region kisi
    if (Secim == "Kişi")
    {
          if (!String.IsNullOrEmpty(searchString))
        {
        var Users = (from s in usrdb.UserProfiles
                     where (s.Adi+" "+s.Soyadi).ToUpper().Contains(searchString.ToUpper())
                     select s).ToList();
        for (int i = 0; i <= Users.Count()-1; i++)
        {
            int uss = Users[i].UserId;

            var  gelen = (from a in fddb.FamilyDetaildbs
                     where a.UserId == uss
                     select new {a.FDtId,a.FamilyName}).ToList();
            if (gelen.Count > 0)
            {
                int gell = gelen[0].FDtId;
                int ILevel;
                int Rno;
                int IIntLev;
                // Aileler arası önceden girilmiş bir ilişki var mı?
                var Ailekon = (from f in fidb.FInteractiondbs
                               where (f.SFid == Susr && f.MFid == gell) ||
                               (f.SFid == gell && f.MFid == Susr)
                               select f).ToList();
                if (Ailekon.Count > 0)
                {
                    ILevel = 0;
                    if (Ailekon[0].IntLev == 3)
                    {
                        switch (Ailekon[0].level)
                        {
                            case 1:
                                ILevel = 3;
                                break;
                            case 2:
                                ILevel = 4;
                                break;
                        }
                    }
                    if (Ailekon[0].level == 1 && Ailekon[0].IntLev == 0) /*Teklif*/
                        ILevel = 1;
                    if (Ailekon[0].level == 2 && (Ailekon[0].IntLev == 1 || Ailekon[0].IntLev == 2)) /*Teklif*/
                        ILevel = 2;
                    Rno = Ailekon[0].FIID;
                    IIntLev = Ailekon[0].IntLev;
                }
                else
                {
                    ILevel = 0;
                    Rno = 0;
                    IIntLev = 0;
                }

                FSearhFs.Add(new FSearchF
                {
                    Tid = Rno,
                    Adi = Users[i].Adi,
                    FamilyName = gelen[0].FamilyName,
                    FDtId = gelen[0].FDtId,
                    Soyadi = Users[i].Soyadi,
                    UserId = Users[i].UserId,
                    UserName = Users[i].UserName,
                    IntLev = IIntLev,
                    akkon = ILevel
                });
            }
            else
```

```
                {
                    FSearhFs.Add(new FSearchF
                    { Adi = Users[i].Adi,
                        FamilyName = null,
                        FDtId = 0,
                        Soyadi = Users[i].Soyadi,
                        UserId = Users[i].UserId,
                        UserName = Users[i].UserName,
                        IntLev=0,
                        akkon =-1
                    });
                }
            }
            }
            return View(FSearhFs.ToPagedList(pageNumber, pageSize));
        }
        #endregion
        #region Komşular
        if (Secim == "Komşular")
        {
               var komsum = (from k in fidb.FInteractiondbs
                         where ((k.MFid == Susr) || (k.SFid == Susr)) && k.IntLev!=3
                        select k).ToList();
            if (komsum.Count > 0)
            {
                for (int i = 0; i <= komsum.Count - 1; i++)
                {
                    bool konkon;
                    konkon = false;
                    if (searchString != null)
                        konkon = GenFunc.GetFmNm(2, null, komsum[i].MFid).ToUpper().Contains(searchString.ToUpper());
                    else
                        konkon = true;
                    if (konkon)
                    if (komsum[i].SFid == Susr )
                    {
                        km = GenFunc.GetUserName(null,komsum[i].MUserId);
                        FSearhFs.Add(new FSearchF
                    {
                        Tid = komsum[i].FIID,
                        Adi = km.adi,
                        Soyadi = km.soyadi,
                        FamilyName = GenFunc.GetFmNm(2, null, komsum[i].MFid) ,
                        FDtId = komsum[i].MFid,
                        UserId = komsum[i].MUserId,
                        UserName = GenFunc.GetNcName(komsum[i].MUserId),
                        IntLev = komsum[i].IntLev,
                        akkon = komsum[i].level
                    });
                    }
                    else
                    {
                        km = GenFunc.GetUserName(null, komsum[i].FUserId);
                    FSearhFs.Add(new FSearchF
                    {
                        Tid = komsum[i].FIID,
                        Adi = km.adi,
                        Soyadi = km.soyadi,
                        FamilyName = GenFunc.GetFmNm(2, null, komsum[i].SFid),
                        FDtId = komsum[i].SFid,
                        UserId = komsum[i].FUserId,
                        UserName = GenFunc.GetNcName(komsum[i].FUserId),
                        IntLev = komsum[i].IntLev,
                        akkon = komsum[i].level
                    });
                    }
                }
            }
            return View(FSearhFs.ToPagedList(pageNumber, pageSize));
        }
        #endregion
        return View(FSearhFs.ToPagedList(pageNumber, pageSize));
    }
    public ActionResult interaction(int? fid , int? uid , int? IntLev)
    {
        FInteractiondb FI = new FInteractiondb();
        if (fid > 0)
        {
            FI.MFid = Convert.ToInt32(fid.Value);
            FI.ADate = DateTime.Now;
            FI.MUserId = Convert.ToInt32(uid.Value);
            FI.FUserId = WebSecurity.GetUserId(User.Identity.Name);
            FI.SFid = GenFunc.GetFmId(User.Identity.Name);
            if (IntLev.HasValue)
            {
                FI.IntLev = IntLev.Value;
            }
            else
            {
                FI.IntLev = 0;
            }
            FI.level = 1;
            FI.BDate = new DateTime(1900,01,01,00,00,01);
            if (ModelState.IsValid)
            {
                fidb.FInteractiondbs.Add(FI);
                fidb.SaveChanges();
            }
        }
```

```
            return RedirectToAction("FSearch");
        }
        public ActionResult intDel(int? id , int gelen)
        {
            FInteractiondb Fidbn = fidb.FInteractiondbs.Find(Convert.ToInt32(id));
            if (Fidbn != null)
            {
                fidb.Entry(Fidbn).State = EntityState.Deleted;
                fidb.SaveChanges();
                GenF.BatchF.LogKayit(null, Fidbn, 2, null);
            }
            if (gelen == 2)
                return RedirectToAction("NIndex");
            return RedirectToAction("FSearch");
        }
        public ActionResult NIndex(int? page=1)
        {
                var FSearhFs = new List<FSearchF>();
                int pageSize = 10;
                int pageNumber = (page ?? 1);
                int Susr = WebSecurity.CurrentUserId;
                var komsum = (from k in fidb.FInteractiondbs
                              where ((k.MUserId == Susr) || (k.FUserId == Susr)) && k.level == 2 && k.IntLev != 3
                             select k).ToList();
                if (komsum.Count > 0)
                {
                    for (int i = 0; i <= komsum.Count - 1; i++)
                    {
                        if (komsum[i].MUserId != Susr )
                        {
                            km = GenFunc.GetUserName(null,komsum[i].MUserId);
                            FSearhFs.Add(new FSearchF
                        {
                            Tid = komsum[i].FIID,
                            Adi = km.adi,
                            Soyadi = km.soyadi,
                            FamilyName = GenFunc.GetFmNm(2, null, komsum[i].MFid) ,
                            FDtId = komsum[i].MFid,
                            UserId = komsum[i].MUserId,
                            UserName = GenFunc.GetNcName(komsum[i].MUserId),
                            IntLev = komsum[i].IntLev,
                            akkon = komsum[i].level
                        });
                        }
                        else
                        {
                            km = GenFunc.GetUserName(null, komsum[i].FUserId);
                        FSearhFs.Add(new FSearchF
                        {
                            Tid = komsum[i].FIID,
                            Adi = km.adi,
                            Soyadi = km.soyadi,
                            FamilyName = GenFunc.GetFmNm(2, null, komsum[i].SFid),
                            FDtId = komsum[i].SFid,
                            UserId = komsum[i].FUserId,
                            UserName = GenFunc.GetNcName(komsum[i].FUserId),
                            IntLev = komsum[i].IntLev,
                            akkon = komsum[i].level
                        });
                          }
                    }
                 }
                return View(FSearhFs.ToPagedList(pageNumber, pageSize));
        }
        public ActionResult Details(int? uuid, int? ffid)
        {
            if (uuid.HasValue)
            {
                UserProfile Usr = usrdb.UserProfiles.Find(uuid.Value);
                if (Usr != null)
                    Fsc.UserProfiles = Usr;
            }
            if (ffid.HasValue)
            {
                FamilyDetaildb Fd = fddb.FamilyDetaildbs.Find(ffid.Value);
                if (Fd != null)
        Fsc.FamilyDetaildbs = Fd;
                Fsc.Fmc = GenFunc.GetFmCount(ffid.Value, 4);
                Fsc.Fkc = GenFunc.GetFmCount(uuid.Value, 2);
            }
            return View(Fsc);
        }
        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                fddb.Dispose();
                usrdb.Dispose();
                fidb.Dispose();
            }
            base.Dispose(disposing);
        }
    }
}
```

Index.cshtml

```
@model PagedList.IPagedList<FSNApp.Models.FSearchF>
@using PagedList.Mvc;
@{
    ViewBag.Title = "Arama";
    Layout = "~/Views/Shared/BaseLayout.cshtml";
}
<img src="/Pictures/icons/Search.png" width="60" height="60" />
<h3>Arama ve Bağlantı</h3>
<hr />
@using (Html.BeginForm("Index" , "FSearch", FormMethod.Post, new { id = "submitfm" }))
{
    <fieldset>
    <div>
    <p>
        İsim : @Html.TextBox("SearchString", ViewBag.CurrentFilter as string)
        <input type="submit" value="Arama" />
     </p>
     </div>
    <div>
    <p>
        Kişi <input type="radio" id="Secim1" name="Secim" value="Kişi" @if(ViewBag.Secim=="Kişi")
{<text>checked="checked"</text>} />
        Aile <input type="radio" id="Secim2" name="Secim" value="Aile" @if(ViewBag.Secim=="Aile")
{<text>checked="checked"</text>} />
        Komşular <input type="radio" id="Secim3" name="Secim" value="Komşular" @if(ViewBag.Secim=="Komşular")
{<text>checked="checked"</text>}/>
    </p>
    </div>
@if (Model.Count>0)
{
<table class="table">
    <tr>
        <th>
            Kullanıcı Adı
        </th>
        <th>
            Adı
        </th>
        <th>
            Soyadı
        </th>
        <th>
            Aile Adı
        </th>
        <th>
            Seviye
        </th>
        <th>
            Durum
        </th>
    </tr>
    @foreach (var item in Model)
    {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.UserName)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Adi)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Soyadi)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.FamilyName)
            </td>
            <td>
                @if ((item.UserId != WebSecurity.CurrentUserId) && item.FDtId > 0 && item.akkon ==0)
                {
                @Html.ActionLink("[+] Aile ", "Index",
                new { fid = item.FDtId, uid = item.UserId, searchString = ViewBag.CurrentFilter as string, isl = 1 , Secim =
ViewBag.Secim as string} )

                @Html.ActionLink("[+] Birey", "Index",
                new { fid = item.FDtId, uid = item.UserId, searchString = ViewBag.CurrentFilter as string, isl = 3 , Secim =
ViewBag.Secim as string} )
                }
                @switch (item.akkon)
                {
                        case 1:
                        <p> Aile Talep var </p>
                        break;
                        case 2:
                        <p> Aile Bağlantı var</p>
                        break;
                        case 3:
                        <p> Birey Talep Var</p>
                        break;
                        case 4:
                        <p> Birey Aile Üyesi</p>
                        break;
                }
                @if (item.akkon<0)
                {
                    <p>Aile yok.</p>
                }
            </td>
            <td>
                @switch (item.IntLev)
```

```
                {
                        case 1:
                        <p> Akraba </p>
                        break;
                        case 2:
                        <p> Komşu</p>
                        break;
                }
            </td>
            <td>
                @if (item.akkon == 1 || item.akkon == 3)
                {
                        @Html.ActionLink("[-]", "Index",
                        new { did = item.Tid , searchString = ViewBag.CurrentFilter as string, isl = 2 , Secim =
ViewBag.Secim as string})
                }

            </td>
            <td>
            @Html.ActionLink("Detay", "Details", new { uuid = item.UserId, ffid = item.FDtId })
            </td>
            <td>
                @Html.HiddenFor(modelItem => item.Tid)
                @Html.HiddenFor(modelItem => item.FDtId)
                @Html.HiddenFor(modelItem => item.UserId)

            </td>
        </tr>
    }
</table>
}
    </fieldset>
}
<p>Not : Sadece ailelerle bağlantı kurabilirsiniz.</p>
<br />

Page @(Model.PageCount < Model.PageNumber ? 0 : Model.PageNumber) of @Model.PageCount

@Html.PagedListPager(Model, page => Url.Action("Index",
    new { page, sortOrder = ViewBag.CurrentSort, currentFilter = ViewBag.CurrentFilter }))

<div>| @Html.ActionLink("Ana Sayfa", "Index", "Main") |</div>
```

ÖZGEÇMİŞ

1974 yılında doğdu. İlköğrenimi Ordu, orta ve lise öğrenimini İstanbul'da tamamladı. End. Meslek Lisesi Elektronik bölümünden 1992'de mezun oldu. 2009 yılında Anadolu Üniversitesi, İşletme Fakültesi İşletme Bölümünü bitirdi. 1992- 1997 yıllarında Bil-Tek Bilgisayarda çalıştı. 1999- 2004 yıllarında Lonca Organizasyon Bilgi İşlem biriminde görev aldı. 2004'de İGDAŞ(İstanbul Gaz Dağıtım A.Ş) Bilgi Sistemleri Müdürlüğünde Bilgisayar programcısı olarak göreve başladı. Yine aynı işletmede 2009 yılında itibaren Proje yöneticisi olarak görev yapmaya devam etmektedir.