TARIM ÜRÜNLERİNİN GIDA GÜVENLİĞİ BİLGİ SİSTEMLERİ İLE İZLENEBİLİRLİĞİ

TRACEABILITY OF AGRICULTURAL PRODUCTS BY FOOD SAFETY INFORMATION SYSTEM

EMRAH ORAL

Hacettepe Üniversitesi

Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliğinin

Gıda Mühendisliği Ana Bilim Dalı için Öngördüğü

YÜKSEK LİSANS TEZİ

Olarak Hazırlanmıştır.

2009

TARIM ÜRÜNLERİNİN GIDA GÜVENLİĞİ BİLGİ SİSTEMLERİ İLE İZLENEBİLİRLİĞİ

Emrah Oral

ÖZ

Bu çalışmada, ülkemiz tarımında ve ihracatında önemli bir yere sahip olan taze meyve ve sebze sektöründe gıda güvenliği yönünden mevzuata uymayan ve halk sağlığını tehlikeye düşüren ürünlerin sistem içinde yer alan tüm paydaşlar ile birlikte izlenebilmesi için bir yazılımın hazırlanması amaçlanmıştır. Taze meyve sebze sektöründe ulusal ve bölgesel düzeyde faaliyet gösteren arz/talep yönlü paydaşlar, düzenleyici ve denetleyici aktör ve örgütlenmelerin yer aldığı bir izlenebilirlik yazılımı oluşturulmuştur.

Bu kapsamda, ülkemiz taze meyve ve sebze sektörü tedarik zinciri göz önünde bulundurularak “tarladan sofraya gıda güvenliği” esasına göre üründe izlenebilirliği sağlamak ve zirai ilaç kalıntısı yönünden tespit edilen soruna anında müdahale edebilmek amacıyla elektronik ortamda yürütülecek bir bilgisayar izlenebilirlik yazılımı hazırlanmıştır.

Ülkemizdeki çiftçi profili göz önünde bulundurulduğunda çiftçilerin hazırlanan bilgisayar yazılımına entegre olmasının güçlüğü düşünülerek izlenebilirlik sistemi bilgisayar yazılımının zirai ilaç bayi, komisyoncu, toptancı, süpermarket ile gıda kontrol denetimini gerçekleştiren kamu tarafından kullanılması planlanmıştır.

Gıda güvenliği ve kontrolünü sağlamak amacıyla web tabanlı, merkezi sistem mimarisi ve merkezi veri tabanı yapısı üzerinde çalışacak bir izlenebilirlik uygulama sistemi olan TAMSİS (Taze Meyve ve Sebze İzlenebilirlik Sistemi) bilgisayar yazılımı hazırlanmıştır.

TAMSİS, ideal bir sistem olup gıda zincirinin her kademesinde tüm ilgili taraflarca yapılan her türlü işlem ve uygulamanın izlenebilirliğini sağlayan bir sistemdir. Tüm gıda zincirinde yatay ve dikey izlenebilirlik ancak komisyoncu, toptancı, süpermarket ve gıda kontrol denetimini gerçekleştiren kamunun TAMSİS’e entegre olup düzenli bir şekilde veri yüklemesiyle mümkün olacaktır.

Bu yazılımla gerçekleştirilecek etkin izlenebilirlik sistemi herhangi bir gıda güvenliği sorununda hızlı bilgi toplamayı dolayısıyla sorunun kaynağı ve nedenini mümkün olduğunca çabuk saptamayı gerçekleştirebilecek ve tedarik zincirinde gıda güvenliğinin sürdürülebilirliğini sağlayacaktır.

Anahtar Kelimeler: İzlenebilirlik, taze meyve ve sebze

Danışman: Doç. Dr. Ümran UYGUN, Hacettepe Üniversitesi, Gıda Mühendisliği Anabilim Dalı

TRACEABILITY OF AGRICULTURAL PRODUCTS BY FOOD SAFETY INFORMATION SYSTEM

Emrah Oral

ABSTRACT

The main objective of the study is to prepare a software program for tracing fresh fruits and vegetables, which are important products in the agriculture and exportation of Turkey. Monitoring the products not complying with the legislation related to food safety and endangering public health is aimed within the system by all shareholders. A traceability program has been created including the supply/demand shareholders, the regulatory and controlling actors and organizations that carry on businesses at the national and regional levels in the fresh fruits and vegetables sector.

In this context, a computerized traceability software program has been prepared in order to provide traceability of products on the basis of “food safety from farm to fork” by taking into consideration the supply chain of Turkey’s fresh fruits and vegetables sector for the purpose of interfering into determined pesticide residues issue immediately.

Traceability system computer software program is planned to be used by pesticide dealers, commissioners, wholesalers, supermarkets and the officials in charge undertaking the food inspection controls when the farmers’ profile in Turkey and the difficulties of integrating them with the prepared computer software program is considered.

TAMSİS (Traceability System for Fresh Fruits and Vegetables) computer software program that will be operating on web-based, centralized system architecture and centralized data base traceability implementation system has been prepared for the purpose of providing food safety and control.

TAMSİS is an ideal system providing for the traceability of any kind of processes and applications undertaken by the related parties at each stage of the food chain. Horizontal and vertical traceability in the whole food chain will only be possible when commissioners, wholesalers, supermarkets and the public institutions

carrying out food inspection controls are integrated with TAMSİS and upload data regularly.

The effective traceability system that will be utilized by this program will provide the sustainability of food safety in the supply chain through fast information collection, thus will enable the determination of the source and reason of the problem as soon as possible when a food safety problem emerges.

Key Words: Traceability, fresh fruit and vegetables

Advisor: Assoc. Prof. Dr. Ümran UYGUN, Hacettepe University, Faculty of Engineering, Food Engineering Department

TEŞEKKÜR

Yüksek lisans konumun belirlenmesinde ve çalışmam boyunca sağlamış olduğu imkanlardan dolayı tez danışmanım olan, sayın Doç. Dr. Ümran UYGUN' a çok teşekkür ederim. Tez çalışmam boyunca değerli katkı ve bilgilerini esirgemeyen ve bana sağladıkları destek ve yardımlardan dolayı Ali Can MOGOL ve Burçe ATAÇ MOGOL' a teşekkürü bir borç bilirim. Çalışmam boyunca katkılarını esirgemeyen Volkan ALTUN ve Yavuz AKIN' a teşekkür ederim.

İÇİNDEKİLER DİZİNİ

Sayfa

ŞEKİLLER DİZİNİ

Sayfa

SİMGELER VE KISALTMALAR DİZİNİ

AB	Avrupa Birliği
TAMSİS	Taze Meyve ve Sebze İzlenebilirlik Sistemi
WHO	Dünya Sağlık Örgütü
FAO	Gıda ve Tarım Örgütü
CAC	Codex Alimentarius Komisyonu
GS1	Global Standarts 1
TOBB	Türkiye Odalar ve Borsalar Birliği
BSE	Bovine Spongiform Encephalopathy
vCJD	Creutzfeldt-Jakob hastalığı
HACCP	Tehlike analizleri kritik kontrol noktaları
SPC	Statistical Process Control
IT	Information Technology
PCR	Polimeraz Zincir Reaksiyonu
MIR	Mid-Infrared Spektroskopi
SPME	Katı Faz Mikroekstraksiyon
GC-MS	Gaz Kromatografisi-Kütle Spektrofotometre
DNA	Deoksiribonükleik asit
RFID	Radio Frequency Identification
EPC	Elektronik Ürün Kodu
EAN-UCC	EAN International-Uniform Code Council
GTIN	Global Trade Item Number-Ticari Ürün Numarası
EUREPGAP	Avrupa perakende sektöründe iyi tarım uygulamaları standardı
GPS	Küresel Yer Belirleme Sistemi veya Küresel Konumlandırma Sistemi
GIS	Coğrafi Bilgi Sistemi
RS	Uzaktan Algılama
J2EE/JEE	Java Enterprise Edition
İVTYS	İlişkisel Veri Tabanı Yönetim Sistemi
ORM	Nesne İlişkisel Eşleştirim (Object Orient Relation Mapping)

1. GİRİŞ

Tarım, ekonominin önemli bileşeni olmasının yanı sıra, ulusal nüfusun gıda kaynağı olması, kırsal nüfusun istihdamı ve ekolojik çevrede yarattığı etki sebebiyle aynı zamanda sosyal ve kültürel öneme sahiptir. Küresel rekabetin arttığı bir dönemde, hemen her alanda değişimin gereklerini yerine getirmek kaçınılmaz olmuştur. Küresel eğilimlerin, rekabetin kurallarını belirlediği dünyada, ülkeler artık kendi başlarına hareket edememektedirler. Günümüzde tarım ürünleri ticaretinde gıda güvenliği ön planda tutulmakta ve uluslar arası standartlara uyma zorunluluğu bulunmaktadır. Son zamanlarda, bu nedenle gıda güvenliği kamuoyunun gündeminde daha fazla yer almaya başlamış ve tarım ürünlerinde ve gıda sektöründe izlenebilirlik önem kazanmaya başlamıştır.

Ülkemizde yaşanan sıkıntıların başında, geleneksel aile işletmelerinin yaygınlığı, üreticilerin eğitim düzeylerinin düşük olması izlenebilirliğin tam ve etkin olarak gerçekleştirilememesine neden olmaktadır. Hammaddede ve son üründe insan sağlığını doğrudan ve/veya dolaylı olarak olumsuz yönde etkileyen riskler tespit edildiğinde ilgili birimler arasındaki bilgi akış sisteminin ve alınan tedbirlerin koordinasyonunun etkin olarak yapılabilmesi için iyi bir izlenebilirlik sistemine gereksinim duyulmaktadır.

Gıda güvenliği yönünden taze meyve ve sebzedeki en önemli risk, üretim girdilerindeki kalıntı sorunudur. Tarımsal üretimde kullanılan ilaç, gübre ve bitki gelişmeyi düzenleyicilerin yanlış kullanımı sonucu, meyve ve sebzenin metabolizmasında yeterince parçalanmayan kimyasallar üründe kalıntıya sebep olmaktadır. Bu sorun ve bununla beraber gıdaların işleme ve pazarlama ortamlarında teknik ve hijyenik şartlar Tarım ve Köyişleri Bakanlığı'na bağlı Koruma ve Kontrol Genel Müdürlüğü tarafından belirlenmiştir. Gıda güvenliği yönünden taze meyve ve sebze sektörünü düzenleyen ana mevzuat 5179 Sayılı Kanun'dur. Bu kanun izlenebilirliğin tesis edilmesini zorunlu hale getirmiştir.

Avrupa Birliği'nin (AB) 28 Ocak 2002 tarih 178/2002/EC sayılı yönetmeliğinde izlenebilirlik kavramı "*gıda, yem ve gıda olarak üretilen hayvan veya gıda veya yeme katılmak amacıyla üretilen veya katılması beklenen maddeleri üretim, işleme ve dağıtımın tüm aşamalarında izleyebilmek ve takip edebilmek*" şeklinde tanımlanmaktadır. Söz konusu yönetmelik, ülkemiz için çok kritik bir öneme sahip

olup bu yönetmelik gereğince, 1 Ocak 2005 tarihinden sonra AB Gıda Yasası şartlarına uymayan hiçbir gıda ve tarım ürünü AB sınırları içerisine kabul edilmemektedir. Bu yönetmelik, gıda ve tarım ürünlerinin üretimine ve denetlenmesine ilişkin bütün çerçeveleri çizmekte; gıda tedarik zincirlerinde "İzlenebilirlik Sistemi"nin tesis edilmesini açık bir biçimde zorunlu kılmaktadır.

Bu çalışmada, Türkiye ve AB üyesi ülkelerdeki taze meyve ve sebze sektöründeki tarımsal örgütlenme, sektördeki paydaşlar genel düzeyde incelenerek faaliyetleri ortaya konmuş ve gıda güvenliği alanında etkin bir izleme ve denetim sistemi geliştirmek amacıyla bir bilgisayar yazılımı TAMSİS (Taze Meyve ve Sebze İzlenebilirlik Sistemi) hazırlanmıştır.

Hazırlanan TAMSİS yazılımının çiftçiye ek bir yük getirmeden zirai ilaç bayi, komisyoncu, toptancı, süpermarket ile gıda kontrol denetimini gerçekleştiren kamu tarafından kullanılması planlanmıştır. Bu yazılım, gıda kontrol denetimini gerçekleştiren kamuda görevli gıda denetçisine, taze meyve sebzede zirai ilaç kalıntısı yönünden analiz sonucunda tespit edilen tehlikeye karşı anında önlem alma ve ürünün kimler tarafından el değiştirdiği bilgisine ulaşabilme imkanını sağlayacaktır.

2. GENEL BİLGİLER

2.1. Taze Meyve Sebze Sektöründe Yerel ve Ulusal Piyasa Paydaşları, Örgütlenme ve Denetim

3. Üreticiler

Taze meyve ve sebze tedarik zincirinin ilk basamağı olan üreticiler, geleneksel veya modern üretim teknikleri ile mevsiminde veya mevsim dışında üretim gerçekleştiren şahıs ya da şirketlerden oluşmaktadırlar. Üreticiler, meyve ve sebze üretimi için gerekli tohumu, fide/fidanı, tarım ilacı ve gübre gibi üretim girdilerini bu konuda gerekli teknik ruhsatlara sahip işletmelerden tedarik ederek üretim yapmaktadırlar.

Üreticiler, faaliyet gösterdikleri ülkenin ulusal mevzuatına göre ve pazarın talep ettiği kalite ve gıda güvenliği standartlarına uygun üretim yapmakla yükümlüdürler. Üretim yapan işletmenin yapısına göre, ürün hiç işlenmeden veya belirli bir düzeye kadar işlenerek tedarik zincirinin bir sonraki basamağı olan toptancılara aktarılır. Farklı ülkelerde taze meyve ve sebze sektöründe farklı tedarik zinciri yapıları bulunmaktadır.

Özellikle kişi başına milli gelirin yüksek olduğu gelişmiş ülkelerde üretici olarak faaliyet gösteren işletmelerin tedarik zincirinin sonraki basamaklarında da faaliyet gösterdiği (hammaddeden pazara kadar) entegre işletme yapıları bulunmaktadır. Türkiye'de ise bu entegre yapıyı sağlayan çok az işletme bulunmaktadır (Yurdakul, 2002).

4. Komisyoncular

Türkiye'de meyve ve sebze ürünleri, bazı pazarlama kanalları yolu ile tüketiciye çeşitli şekillerde ulaştırılabilmektedir. Meyve ve sebze üreticilerinin bir kısmı ürünlerini yol üstü pazarlarında veya tarlada (üretim yerinde) satmaktadırlar. Bir kısım üreticiler, üretim yerinin pazara uzak olması veya nakliye masrafının ağır olması sebebiyle, ürününü üretim yerinde komisyonculara satarak pazarlayabilmektedir. Komisyoncu, çiftçilerden satın aldığı ürünleri perakendeci ve toptancı dağıtım kanallarına pazarlamakta ve ürün buradan pazar, süper market, manav ve bakkallara aktararak tüketiciye ulaştırmaktadır (Yurdakul, 2002).

Taze meyve ve sebze tedarik zincirinde dünyada farklı ülkelerde farklı zincir yapıları bulunmaktadır. Dünyadaki diğer yapıların aksine Türkiye'de daha karmaşık bir tedarik zinciri yapısı göze çarpmaktadır. Ülkemizdeki mevzuata göre, üreticilerin ürünlerini doğrudan tüketiciye veya perakendeciye pazarlama yolu kısıtlanmakta olup özellikle AB üyesi ülkelerde "Toptancılar" olarak ifade edilen halka, Türkiye'de "Komisyoncular" ve "Tüccarlar" olarak iki ayrı halkadan oluşmaktadır (Yurdakul, 2002).

5. Toptancılar

Taze meyve ve sebze tedarik zincirinde bir sonraki basamak olan yurtiçi perakendeciler ile komisyoncular arasında ara dağıtım kanalı olarak bulunurlar. Toptancı, gerçek ya da tüzel kişiler olup farklı komisyonculardan alınan malı gerekirse işleyerek gerekirse işlenmeden perakende satış yerlerine satışını gerçekleştirirler.

Toptancılar, perakendecilerin talebi doğrultusunda ürünleri alarak yine talep doğrultusunda (çeşit/boy/renk/ambalaj) işleme faaliyeti göstermektedirler. Toptancıların büyük çoğunluğu aynı zamanda paketleme/işleme tesislerine sahip olabilirler. Toptancılar, Tarım ve Köyişleri Bakanlığı tarafından 5179 sayılı Kanun ile denetlenirler. Bu nedenle teknik anlamda komisyonculardan daha fazla yükümlülüğe sahiplerdir. Söz konusu kanun çerçevesinde gıda işleyen bir tesis olarak nitelendirildikleri için Tarım ve Köyişleri Bakanlığı tarafından "Gıda İşyeri Çalışma İzni ve Sicil Numarası" almak zorundadırlar (TKİB, 2008).

6. İhracatçılar

Yurtiçinde üretilen ürünlerin yurtdışı piyasasına pazarlanmasını sağlayan ticari işletmeler olan ihracatçılar, ulusal mevzuat, uluslararası ticareti düzenleyen mevzuat ve uluslararası antlaşmalara uygun bir şekilde ürün arzını sağlarlar. Dünyada ve ülkemizdeki ihracatçı yapısı karşılaştırıldığında işleyiş düzeyinde çok büyük farklılıklar olmamakla birlikte, işletme büyüklüğü ve işletmelerdeki kurumsallaşma bakımından büyük farklılıklar bulunmaktadır (Akbay, 2005).

Taze meyve ve sebze ihracatçıları, ihracata konu ürünü kendi tesislerinde işleyip pazarlayan firmaların yanında, toptancılarla yapılan anlaşmalarla, toptancıların hazırladığı ürünü sadece ihraç etme faaliyeti gösteren firma olarak ihracat

yapabilirler. İhraç edilecek ürünü kendilerinin işlemeleri durumunda, toptancılarla yaklaşık olarak aynı mevzuat hükümlerine tabi tutulmaktadırlar (TKİB, 2008).

7. Perakendeciler

Taze meyve ve sebze tedarik zincirinde ürünün tüketiciye ulaşmasını sağlayan son basamak "Perakendeciler"dir. Perakendeciler, tüketime hazırlanmış ürünlerin tüketiciye sunulduğu satış noktalarıdır. Türkiye'deki gıda perakendeciliği sektörü oldukça parçalı bir yapı sergilemektedir. Geleneksel aile işletmeciliğine dayanan bakkallar, açık ve kapalı pazarlar halâ tüm ülkede oldukça yaygındır. Bu geleneksel perakendecilik şekilleri modern perakendecilik şekillerinin henüz boy göstermediği kırsal bölgeler ve küçük kasabalarda oldukça önemli bir yer tutmaktadır (TKİB, 2008).

8. Düzenleyici ve denetleyici paydaşlar

Tarımsal üretim ve ticaretinde mevzuat oluşturma ve denetleme büyük oranda kamu kuruluşları tarafından yapılmaktadır. Özellikle üretim ve işleme kısımları Tarım ve Köyişleri Bakanlığı; ticaret Sanayi ve Ticaret Bakanlığı; ithalat ve ihracat ise Başbakanlık Dış Ticaret Müsteşarlığı tarafından denetlenmektedir.

Tarım Bakanlığı'nın taşra teşkilatları olan İl Tarım Müdürlükleri, Bakanlık Koruma ve Kontrol Genel Müdürlüğü'nün koordinasyonunda piyasaya arz edilen ürünlerin uygun şekilde üretildiği, işlendiği ve saklandığını garanti altına alan denetimler yapmaktadır (TKİB, 2008).

9. Küresel paydaşlar

10. Codex alimentarius komisyonu (CAC)

Codex Alimentarius, Dünya Sağlık Örgütü (WHO) ve Gıda ve Tarım Örgütü'nün (FAO) 1963 yılında gıda standartlarının, kılavuzlarının ve uygulama kuralları gibi, ilgili yazılı materyallerin geliştirilmesi amacıyla FAO/WHO Ortak Gıda Standartları Yazılımı çerçevesinde kurulmuş bir komisyondur. Bu yazılımın amacı tüketicilerin sağlığının korunması, gıda ticaretinde adil ticaretin sağlanması ve uluslararası devlet kurumları veya özerk kuruluşların gerçekleştirdiği gıda standartları çalışmalarının koordinasyonunun geliştirilmesini sağlamaktır.

Codex Alimentarius'un taze meyve ve sebzedeki zirai ilaç kalıntıları için belirlediği limitler, AB ve Türkiye kodeksine referans oluşturmaktadır. Codex tarafından 2006 yılında yayınlanan "İzlenebilirlik İçin Prensipler (CAC/GL 60-2006)" izlenebilirlik için genel çerçeveyi oluşturmaktadır (Codex, 2008).

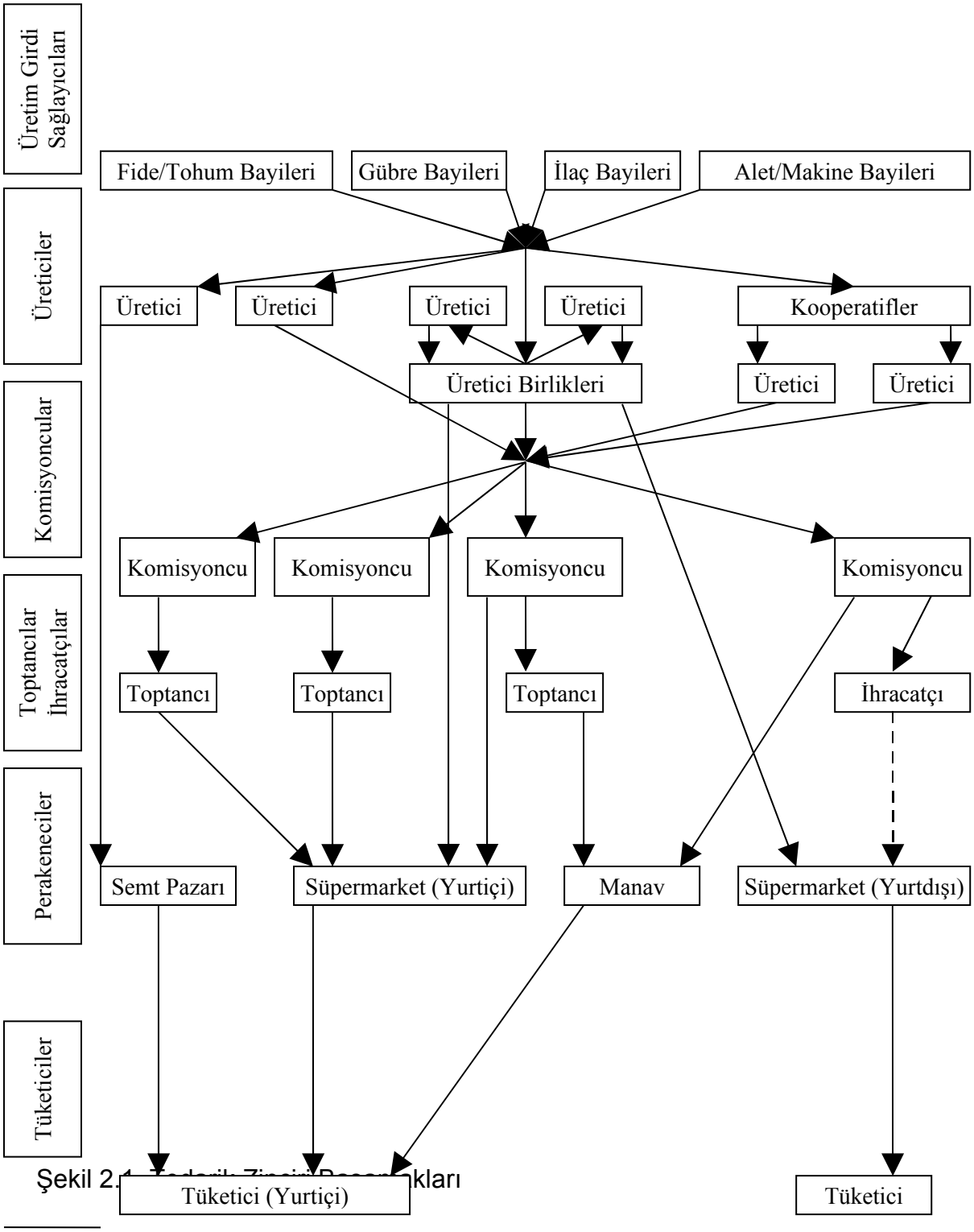
11. Global standarts 1 (GS1)

GS1, bütün iş sektörlerinde küresel standartların ve arz ve talep zincirlerinin etkinliği ve görünürlüğünün geliştirilmesine yönelik çözümlerin tasarımı ve kurulmasını sağlamayı amaçlayan küresel bir organizasyondur. GS1, Türkiye'de Türkiye Odalar ve Borsalar Birliği (TOBB) bünyesinde faaliyet göstermektedir (GS1, 2008).

12. Taze Meyve ve Sebze Tedarik Zinciri Yapısı

Meyve ve sebze ürünleri, bazı pazarlama kanalları yolu ile tüketiciye çeşitli şekillerde ulaştırılabilmektedir. Şekil 2.1.'de taze meyve ve sebze paydaşlarının üretim girdi sağlayıcılarından tüketime kadar basamakları şekiller ile ifade edilmiştir (Özkan, 2006). Şekilde de görüldüğü gibi, Türkiye'de karmaşık bir taze meyve ve sebze tedarik zinciri ortaya çıkmaktadır. Küçük ve dağınık üretim yapısının da sonuçları göz önünde bulundurulduğunda bu zincirin işleyişinde önemli sorunlar oluşmaktadır (TKİB, 2004).

Üreticiler, komisyonculara, tarım kooperatiflerine veya ihracatçı firmalara doğrudan ürünlerini pazarlayabilmektedir. Taze meyve ve sebze pazarlama kanallarının en uzun olanı "üretici – toplayıcı – komisyoncu (üretim yerinde) – toptancı – komisyoncu (tüketim yerinde) – perakendeci – tüketici" şeklindedir (Yurdakul, 2002). Bu sistemde ürünün çok fazla el değiştiriyor olması izlenebilirliğini güçleştirmekte ve kalitesinin de azalmasına neden olmaktadır.



Şekil 2.1 Tedarik Zinciri Bileşenleri

12.1. İzlenebilirlik

İzlenebilirlik, bir ürünün üretim aşamasından itibaren tanımlanarak, paketlenmesi, depolanması, nakliyesi ve nihai satış noktasına ulaşması ile ilgili tüm bilgilerin kolay ulaşılabilecek şekilde kayıt altına alınması, böylece geriye dönük takibinin yapılabilmesi sürecidir (Opara, 2003).

Dünyada son yıllarda, çoğunluğu hayvansal kökenli olan ve insan sağlığı için ciddi tehditler, tehlikeler oluşturmuş ve hatta binlerce ölümle sonuçlanmış gıda kaynaklı sorunlar yaşanmıştır. İlk olarak 1986'da İngiltere'de görülen *Bovine Spongiform Encephalopathy* (BSE) (ya da "deli dana" hastalığı) ve bunun insanlardaki varyantı olan Creutzfeldt-Jakob hastalığı (vCJD) dünya kamuoyunda kapsam itibariyle en geniş şekilde tartışılmış gıda güvenliği sorunlarının başında gelmektedir. Buna, birçok ülkede zehirlenme, hastalık ve ölüm vakalarıyla sonuçlanmış birçok sorunu ekleyebiliriz. Belçika'da yem nakil paletlerindeki zehirli ve öldürücü dioksin kalıntısıyla oluşan kontaminasyon krizi, tavuk yeminde kanalizasyon atıkları, tavuk etinde büyümeyi arttırıcı hormon kullanımı ve tedavi amaçlı kullanılan antibiyotik kalıntıları örnek verilebilir. Ayrıca birçok ilaca karşı direnç kazanmış *Salmonella* (*Salmonella enteritidis* ve *Salmonella typhimurium*), peynirde patojenik bir bakteri olan *Listeria*, kırmızı et ile meyve ve sebzede *E.coli* kaynaklı sorunlar da insan sağlığı üzerinde ciddi tehlike/tehdit yaratanlardan bazılarıdır. Dahası *Campylobacter* birçok ülkede tavuk etiyle ilişkili görülen gıda zehirlenmelerinin başında gelmektedir. En son "kuş gribi" olarak bilinen "H5N1 viral enfeksiyonu" ve buna bağlı olarak Uzakdoğu Ülkeleri ve ülkemizde görülen ölümler en yoğun tartışılan konulardan bir başkasıdır (Furness and Osman, 2006).

Gıdalardan kaynaklanan sağlık sorunları, ölüm vakaları ve potansiyel riskler tüketicilerde gıdalara karşı büyük güvensizlik yaratmıştır. Buna bağlı olarak tüketicilerin, özellikle gelişmiş ülkelerde, gıda güvenliği ve kalitesi konusundaki duyarlılıkları artmış, bu yönde etkili yöntemlerin uygulanmasını isteyen tepkiler göstermeye başlamışlardır. Tüketiciler, güvenli ve kaliteli gıda istemleri dışında ayrıca çevresel, ekonomik ve sosyal açıdan sürdürebilir, hayvan hakları, sağlığı ve refahına da saygılı tarımsal üretim isteyen yurttaş konumuna geçmişlerdir. Sonuçta tüm bu beklenti ve istekler, devlet yönetimlerini gıda güvenliği ile

sürdürülebilir tarım ve kırsal kalkınma için birtakım tedbirler almaya ve yasal düzenlemeye gitmeye zorlamıştır (Cebeci, 2006).

5179 sayılı Kanun'da gıda güvenliği için izlenebilirlik, sorumluluklar ve denetimler de dahil birçok uygulama ve işlemin yönetmeliklerle düzenleneceğine dair hükümler yer almaktadır. 16. maddesinde "*Gıda işletmecileri; gıda, gıdanın elde edildiği hayvan, bitki ya da gıda maddelerine karıştırılması tasarlanan herhangi bir maddeyi, kimden aldıklarını belirleyebilecek sisteme sahip olmak zorundadır. Gerektiğinde denetim sonucu oluşan bilgiler ilgili mercilere verilir.*" denildiğinden ve 3. maddede ise "*İthal ettikleri, ürettikleri, işledikleri, imal ettikleri veya dağıtımını yaptıkları gıda maddelerinin gıda mevzuatı şartlarına uygunluğundan sorumlu olan gerçek veya tüzel kişiler*" gıda işletmecisi olarak tanımlandığından tarım işletmeleri de izlenebilirlik sistemi tesis etmekle yükümlü kılınmıştır (Cebeci, 2006).

12.1.1. İzlenebilirliğin yararları

1. Sadece uygun kalitedeki ham maddelerin, katkı ve ingrediyenlerin son ürüne girdiğini güvenceye almak,
2. Birbirine benzer ürünlerde karışmayı önleyecek açık belirteçlerin kullanılmasını sağlamak,
3. Minimum maliyetle, hataların nedenlerini ortaya koymak ve gerekli önlemleri almak,
4. Envanter kontrolü ve planlamayı (örneğin ilk giren malzemenin ilk kullanılması esası bozulma riskini azaltır) sağlamak,
5. Ürünlerin insan sağlığı için bir tehdit ve/veya tehlike oluşturması halinde problemin kaynağını, nedenini ve sorumlularını saptamak ve gerekli önlemleri almak üzere geriye doğru izlenmesini sağlamak,
6. Tehlike ve/veya tehdit oluşturan ürünleri geri toplamak üzere ileriye doğru izlenmesini sağlamak,
7. Tehlike analizleri kritik kontrol noktaları (HACCP) planlarının realize edilmesi ve sürdürülebilirliğini sağlamaktır.

Bunun yanında tüketici tarafından talep edilmemekle birlikte işletmede kalite yönetimi ve etkinliği açısından aşağıdaki avantajları da sağlamaktadır:

1. Üretim ile ilgili veri ve bilgileri kayıt altına alarak işletmelerde istatistiksel süreç kontrolü (Statistical Process Control, SPC) analizlerine olanak sağlamak; böylece üretim maliyetini ve müşteri memnuniyetini dikkate alan kalite yönetim sistemlerinin geliştirilmesini kolaylaştırmak,

2. İşletme riskini azaltmak, gerekli olduğunda ise geri toplama maliyetini düşürmek,

3. Sessiz geri toplamayı gerçekleştirerek marka imajının korunmasını sağlamak,

4. Sahtecilik/taklitçilik ile mücadeleyi kolaylaştırmak,

5. Tüketicide markaya güven yaratarak rekabet avantajı oluşturmak,

6. Yasalarla yükümlü kılınan belge ve bilgilerin kolayca üretilerek yetkili kuruluşlara ve ticaret ortaklarına ulaştırılmasını sağlamak ve böylece işletme yönetimini etkinleştirmektir (Verdenius, 2006).

12.1.2. İzlenebilirlik türleri

İzlenebilirlik aşağıdaki gösterilen şekilde farklı kategoriler ve amaçlarla gerçekleştirilebilir:

1. Ürün izlenebilirliği, lojistik, geri toplama ve tüketiciye ve diğer taraflara bilgi sağlamayı kolaylaştırma amacıyla bir ürünün tedarik zincirindeki fiziksel konumunu saptama işlemidir.

2. Süreç izlenebilirliği, ürünün üretim, depolama, işleme vb aşamalarında geçirmiş olduğu uygulama ve işlemlerin türü ve zamanını belirleme amacı taşır. Bu bir tür “nerede, ne zaman, ne oldu/yapıldı” sorularına yanıt arayan bir izlenebilirlik biçimidir. Böylece, üretim sırasındaki fiziksel/mekanik, kimyasal, çevresel ve atmosferik faktörler dikkate alınarak tehdit veya risk oluşturan unsurları önleyici eylemlerin gerçekleştirilmesini garanti altına almayı hedefler.

3. Girdi izlenebilirliği üretimde kullanılan tohum, gübre, kimyasal ilaçlar, sulama suyu, toprak yapısı, hayvan, hayvan yemi, katkı maddeleri gibi her türlü girdinin sağlandığı yer ve özellikleri gibi bilgilerin izlenebilirliğini sağlamak amacındadır.

4. Genetik izlenebilirlik bir ürünün genetik yapısını saptama amacındadır. Genetik izlenebilirlik ürünün genetik türü, çeşidi, kaynağında genetik olarak modifiye edilmiş organizma veya madde veya girdi/bileşen (tohum, fide, sperm, embryo vb) kullanılıp kullanılmadığını ortaya koyar.

5. Hastalık ve kalıntı izlenebilirliği gıdaya bulaşabilme ihtimali olan patojenik bakteri, virüs, mantar vb izlemeyi hedefler.

6. Ölçü/Ölçme izlenebilirliği, ürünlerin belli bileşenler ve risk etkenleri bakımından analiz edilmesi yanında ölçü ve test elemanlarının standartlara uygunluğu ve kalibrasyonunun yeterliliğini izlemeyi de amaçlayan bir izlenebilirlik yöntemidir (Opara, 2003).

12.1.3. İzlenebilirlik sistemleri ve kapsamı

İzlenebilirlik, kapsam itibariyle:

a. Bir kademe geriye ve bir kademe ileriye izlenebilirlik,

b. Tam izlenebilirlik şeklinde olabilir.

Mevcut yasal düzenlemeler, bir kademe ileri ve geriye izlenebilirliği zorunlu kılmaktadır. Bir başka deyişle, yasalara göre bir işletme satın aldığı ve sattığı mal ve hizmetlere ilişkin kayıtlardan sorumludur. Tam izlenebilirlik ise ideal bir sistem olup gıda zincirinin her kademesinde tüm ilgili taraflarca yapılan her türlü işlem ve uygulamanın izlenebilirliğini amaçlayan bir sistemdir. Tüm gıda zincirinde yatay ve dikey izlenebilirlik bilgi sistemleri ve uygun teknolojilerin kullanılmasıyla gerçekleştirilebilecek kadar kapsamlıdır.

Böyle bir sistem:

1. İzlenebilirlik için gerekli verilerin etkin, düşük maliyetli ve modern yöntem ve tekniklerle toplanması ve kayıt altına alınmasını (mümkün olduğunca elektronik olanaklarla otomatik olarak) sağlar.

2. Verilerin birbirinden bağımsız uygulama ve çözümler yerine birbiriyle tümleşik ve mümkün olduğunca gerçek zamanlı işlenmesi ve analizini mümkün kılar.

Gıda zincirinde yer alan işletmelerde, girdilere ait veriler, üretim/işlem verileri, satış ve dağıtım verileri söz konusudur. İzlenebilirlik sistemlerinin tesisinde, gerekli/zorunlu verilerin/bilgiler ile bunların tedarik zincirinde akışlarının ve nihayetinde değişimde bulunulacak veri yapısının belirlenmesi sistem tasarımı için önemli bir başlangıç adımıdır. Çünkü ilgili veritabanının tasarımı, ara yüzler, kullanıcılar, roller, raporlar, sorgulamalar, izleme denetimleri vb. tüm unsurları kapsayacak şekilde bir bilgi sisteminin genel yapılandırması ve özelde sistem yazılım araçları ve ara yüzlerinin tasarım ve programlanması için temel işler arasında yer almaktadır (Cebeci, 2006).

İzlenebilirlik sistemlerinde tutulacak kayıtları oluşturan veriler, temel veya zorunlu veriler ile ikincil veya destek verileri olmak üzere iki bölümde incelenebilir. Temel veya zorunlu veriler, izlenebilirlik sisteminin amaçlanan risk ve kriz yönetimini mümkün kılmak, geri toplama mekanizmasını mümkün olan en kısa sürede gerçekleştirmek için gerekli verileri kapsamaktadır. Bu veriler bir başka ifadeyle yasal sorumluluk ve zorunlulukları karşılamak için tutulması gereken verilerdir. İkincil veriler ise, izlenebilirlik sisteminin oluşturulmasında yapısal olarak önemli olmamakla birlikte işletme içi izlenebilirlik (internal traceability) ve kalite yönetiminde iyileştirme sağlamak; müşteri/tüketicilere ayrıntılı bilgi vererek rekabet avantajı sağlamak ve ileride çıkması muhtemel düzenlemelere hazırlık açısından tutulan verileri kapsamaktadır (Cebeci, 2006).

Son yıllarda izlenebilirliğin kontrolü için kayıt ve IT (Information Technology) teknolojisi yanında özellikle Avrupa Birliği projeleri tarafından desteklenen laboratuvar analizleri ile ürünlerin izlenebilirliğinin takip edilmesi yöntemi kullanılmaya başlanmıştır. Örneğin tahıllarda, tavuklarda ve zeytinlerde element izotop oranlarının analiz edilerek ürünlerin hangi bölgeden olduğunun belirlenmesi, balın polimeraz zincir reaksiyonu (PCR) gibi moleküler metotlarla karakterize edilmesi, mid-infrared (MIR) spektroskopi ile zeytinlerin coğrafik orijinlerinin izlenebilmesi, katı faz mikroekstraksiyon (SPME) GC-MS ile zeytinlerde orijinin takibi ve ballarda fenolik ve flavanoidlerin analiz edilerek takibinin yapılabilmesi bunlara bazı örneklerdir (Cebeci, 2006).

12.1.4. İzlenebilirlik uygulaması

12.1.4.1. Ürün tanımlamaları

Gıda izlenebilirliği tedarik zincirinde yer alan ürün ve/veya proseslerin etkin şekilde tanımlanması ve zincirin her aşamasında tanınabilmesi ile ilişkili ve uygulanabilir bir sistemdir. Birincil tanımlama gıdanın biyolojik belirteçler ve spesifik özelliklerine bağlı olan anatomik, fizyolojik, biyokimyasal ve DNA (Deoksiribonükleik asit) analizi dahil moleküler birtakım biyolojik işlemlerle belirlenen kimliğidir. İkincil veya etiket (veri taşıyıcıları) tabanlı tanımlama ise ürünün tanımlanması amacıyla bir dizi alfa-sayısal karakter dizilimi kullanan tekniklerden oluşur. Tanımlama veya kimlik bilgisi, izlenebilirlik veya süreç desteği amacıyla diğer veri/bilgilerle kombine edilebilir. İkincil tanımlayıcı özellikle birincil tanımlayıcının bir veri şablonu veya veritabanı olarak tutulduğu yerlerde birincil tanımlayıcıya bağlanabilir. İşlenmekte veya üretilmekte olan unsurların kaynağını otomatik tanımaya yardımcı olmak ve veri türlerini ayırt etmek üzere üst veriler kullanılabilir (Klafft et al., 2006).

12.1.4.2. Etiketleme

İzlenebilirlik ve süreç destek sistemlerinin geliştirilmesi ve yapılandırılmasında optik-manyetik okuyucu destekli çok sayıda tanımlama sistemi ve teknolojisi kullanılabilir durumda olmakla birlikte barkod ve Radyo Frekansıyla Tanımlama (Radio Frequency Identification veya kısaca RFID) teknolojileri en yaygın olanlarıdır.

Barkod

Doğrusal (lineer) barkodlar perakende ve tedarik zinciri lojistiğinde uzun yıllardan beri otomatik okunabilir tanımlama ve veri aktarımında yaygın olarak kullanılmaktadır. Farklı amaç ve gereksinimleri karşılamak için geliştirilmiş birçok doğrusal barkod gösterimi söz konusu olup bir dizi sayısal veya alfa sayısal karakter dizilimi barkodlar üzerine yazılabilmektedir. Doğrusal barkodlar ayrıca üst veri tanımlayıcıları ile son kullanma tarihi ve ağırlık gibi birtakım veriyi de sınırlı miktarda taşıyabilmektedir. Barkodlar iki boyutlu, çok sıralı ve matriks türünde de olabilirler. Bu tür barkodlar doğrusal barkodlardan daha fazla bilgi taşıyabilmekte, 2000 byte kadar ulaşan hacmiyle bir anlamda küçük bir veri dosyası büyüklüğünde olabilmektedir (Furness and Osman, 2006).

Radyo Frekansıyla Tanımlama (Radio Frequency Identification veya kısaca RFID)

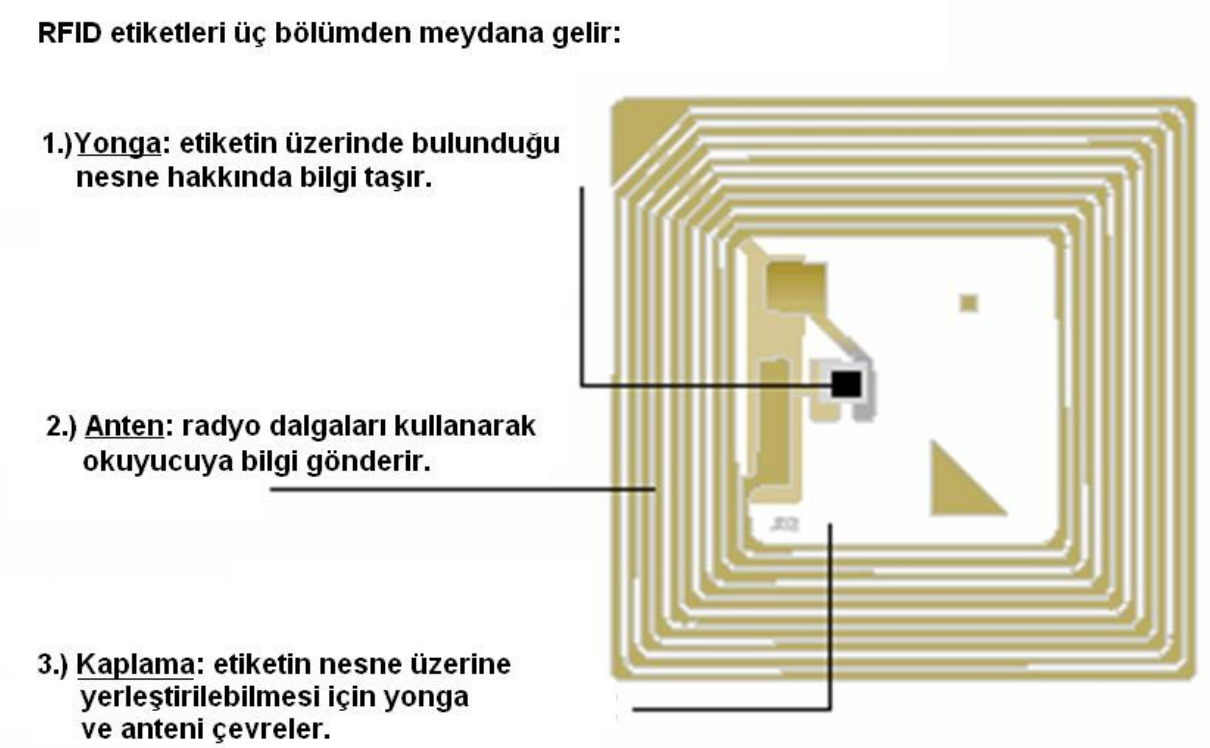
Radyo Frekansı ile Tanımlama (Radio Frequency Identification veya kısaca RFID) teknolojisi, radyo frekansı kullanarak nesneleri tekil ve otomatik olarak tanımlama yöntemidir. RFID, temel olarak bir etiket ve okuyucudan meydana gelir. RFID etiketleri Elektronik Ürün Kodu (EPC) gibi nesne bilgilerini almak, saklamak ve göndermek için programlanabilirler. Ürün üzerine yerleştirilen etiketlerin okuyucu tarafından okunmasıyla tedarik zinciri yönetimi ile ilgili bilgiler otomatik olarak kaydedilebilir veya değiştirilebilir. Haberleşme uydularının belli frekanstan yayın yapan modülleri izlemesi ile lojistik başta olmak üzere otomatik tanımlama sistemlerinde kullanılan ve sürekli gelişme içinde olan son teknolojilerden biridir (Cebeci, 2006).

RFID aynı anda tanımlama, sağlam, kabul edilebilir çalışma mesafesi ile görüş açısı gereksinimine ihtiyacı olmaması gibi özelliklere sahiptir. Radyo frekans dalgalarının nakli ile canlı ve cansız varlıkların otomatik tanımlanması ve daha çok bilgi depolanması mümkün olmaktadır (Furness and Osman, 2006).

Barkodların aksine RFID bir ürün paketi üzerindeki verinin herhangi bir kontak ve ışık gereksinimi olmadan otomatik olarak bir okuyucu tarafından okunmasını sağlamakta; mikro dalga ve uzun dalga gibi değişik dalga boylarında elektromanyetik dalga tekniğine dayanmaktadır. RFID, canlı hayvanları ve taşımacılıkta kullanılan konteynırları tanımada ve otomatikleşmiş üretim süreçlerinde kullanılmaktadır. Tedarik zincirinde izleme ve denetlemede büyük bir etkinlik sağlama potansiyeli olan RFID teknolojisi stok düzeylerindeki hareketleri, işlemlerin gerçek zamanlı optimizasyonunu, havaalanı ve limanlarda taşıma sistemlerinin düzenlenmesini, nakliyenin izlenmesini, nakliye sırasında ürünler hakkında mekanik ve iklimsel etkilerin gözlenmesini olanaklı kılan çok ileri düzeyde izleme ve yönetim sağlama potansiyeli taşıyan bir teknolojidir.

RFID etiketi (Şekil 2.2.), radyo frekansı ile yapılan sorguları almaya ve cevaplamaya olanak tanıyan bir silikon yonga, anten ve kaplamadan meydana gelir. Yonga, etiketin üzerinde bulunduğu nesne ile ilgili bilgileri saklar. Anten, radyo frekansı kullanarak nesne bilgilerini okuyucuya iletir. Kaplama ise etiketin bir

nesne üzerine yerleştirilebilmesi için yonga ve anteni çevreler (Furness and Osman, 2006).



Şekil 2.2. RFID Etiketi

RFID ve barkod teknolojileri farklı teknolojilerdir ve bazen üst üste gelen farklı uygulama alanları vardır. En önemli farkları barkodun görüş alanı teknolojisi olmasıdır. Yani, bir barkod okuyucusu, barkodu okuyabilmek için "görmelidir". Bu nedenle barkodların elle tek tek okutulması gerekmektedir. RFID ise görüş alanı gerektirmez. RFID etiketleri okuyucunun okuma alanı içinde oldukları sürece okunabilirler. Bir başka önemli farklılık da standart barkodların sadece üretici ve ürün çeşidini tanımlamasıdır. Öte yandan, RFID etiketlerinde bir ürünü dünyada tek olarak tanımlayan EPC (Electronic Product Code) numarası saklanmaktadır. EPC, barkod numaralandırmasından farklı olarak nesneleri tekil ve özgün olarak tanımlamaktadır. Örneğin, A marka bir meyve suyu kutusunun üzerindeki barkod tüm aynı marka ve çeşitteki meyve suyu kutularında aynıdır. Bu durum, bir raftaki hangi meyve suyu kutusunun son tüketim tarihinin daha önce geçeceğinin anlaşılmasını engeller. Oysa üzerinde RFID etiketi taşıyan bir meyve suyu kutusunun EPC numarası ile aynı raftaki, aynı marka ve cins meyve suyu kutusunun üzerindeki EPC numarası farklıdır.

12.1.4.3. Ticari ürünlerin numaralandırılması

Ticari ürünlerin tanımları, numaraları ve barkodları, ticari ürünün hareket ettiği tedarik zinciri içinde ve dağıtım kanalları boyunca gerçekleştirilen, satın alma, envanter yönetimi, sipariş verme, satış ve satış noktası operasyonlarında otomasyon yapılmasını sağlamakta, böylece işlemlerin elektronik ortamda gerçekleştirilmesine (elektronik ticaret) olanak vermektedir (TOBB, 2007).

12.1.4.4. EAN-UCC (EAN International-Uniform Code Council) sistemi

Dünyada farklı İzlenebilirlik Sistemleri bulunmakla birlikte, küresel geçerliliği olan, uluslararası ölçekte bilgi standartları koyan ve Birleşmiş Milletler tarafından da tavsiye edilen tek İzlenebilirlik Standardı EAN International tarafından oluşturulmuş olan EAN.UCC sistemidir. EAN.UCC Sistemi çerçevesinde İzlenebilirliğin nasıl tesis edileceği konusundaki çalışmalar, Global Standartlar Merkezi TOBB-GS1 TÜRKİYE tarafından yürütülmektedir (TOBB, 2007).

Kısaca bir bilgi standardı tanımlama ve uygulama sistemi olarak anılabilecek EAN-UCC Sistemi, merkezi Brüksel'de bulunan EAN International tarafından geliştirilmekte ve tüm dünya çapında yönetilmektedir. EAN International'a bağlı yerel EAN Numaralama Organizasyonları, tanımlama ve numaralandırma standartlarının uygulayıcılarıdır. Türkiye'deki EAN Numaralama Organizasyonu Türkiye Odalar ve Borsalar Birliği'nin bünyesindeki Milli Mal Numaralandırma Merkezi (TOBB-MMNM)'dir (TOBB, 2007).

EAN Numaralama Organizasyonları tarafından verilen numaralar ve bu numaraları içeren barkodlar, dünyanın her yerinde geçerli olup; uluslararası tüm ticari işlemler ve tedarik zinciri uygulamalarında herhangi bir değişikliğe gerek kalmaksızın kullanılabilirler (TOBB, 2006).

EAN-UCC Sistemi ile her bir değişik ticari ürüne bu ürünü dünya üzerinde tek olarak tanımlayacak bir numara verilir; bu numara GTIN (Global Trade Item Number-Ticari Ürün Numarası) olarak anılır. GTIN, ürünü kimliklendiren bir numaradır ve ürünün özelliklerine ilişkin hiçbir bilgi içermez. Ürünün tanımında değişiklik olmadığı sürece ürünün GTIN'i de aynı kalır (TOBB, 2006).

EAN-UCC Sisteminde ticari ürünlere verilen GTIN, 14 basamaklı ve tümü rakamlardan oluşan (sayısal) bir numaradır (Şekil 2.2.). Bu numara, ticari ürünün üzerinde gözle görülür biçimde ve barkod sembolü olarak yer alır, aynı zamanda da elektronik ortamda gerçekleştirilen elektronik veri değişimi uygulamalarındaki standart mesaj kayıtlarında kullanılır. EAN-UCC Sisteminde, 14 basamaklı numaradan başka EAN/UCC-8, UCC-12, EAN/UCC-13 numaralı barkodlar da bulunmaktadır (TOBB, 2006).

Söz konusu numaraların önemli özellikleri şu şekilde sıralanabilir:

1. Tektir: Bir ürüne tahsis edilmiş olan barkod numarasının, dünyada başka bir ürüne tahsis edilmiş olması mümkün değildir.

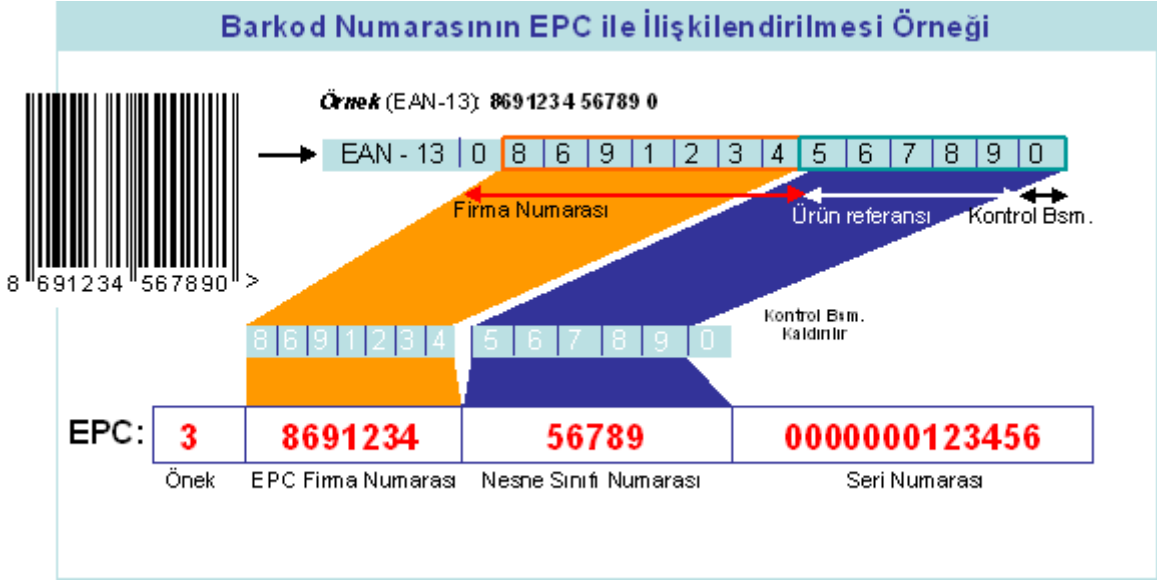
2. Uluslararasıdır: Bir ürüne tahsis edilmiş olan barkod numarası, dünyanın her yerinde tanınır.

3. Güvenlidir: Barkod numaraları içerdiği kontrol hanesi ile doğru veri aktarımını sağlarlar.

Birbirleri ile ticaret yapan işletmeler, EAN.UCC barkodları ile el ve yer değiştiren malı kolayca tanımlarlar. Ayrıca bu malın gönderme, teslim alma, depolama hareketlerini oluşturan verileri paylaşabilirler.

Üründe gıda güvenliğine ilişkin bir sorunla karşılaşıldığında barkod bilgisiyle üretim kaynağına ulaşılması sağlanmaktadır. Ayrıca söz konusu ürünün tedarik zincirinde hangi halkalarda bulunduğu tespit edilebilmekte böylece etkin bir şekilde geri toplanması mümkün olmaktadır.

Tarım ürünleri tedarik zincirinin bütün akışlarının izlenebilmesi yoluyla, ihracatta ve iç tüketimde takip işlemlerinin hız ve doğruluğu arttırılacak böylece gıda güvenliği ile ilgili karşılaşabilecek olası riskler sınırlanabilmektedir (GS1, 2008).



Şekil 2.3. Barkod Numarası Örneği

12.1.4.5. DNA barkodlama

Bir yumurta ikizi olan canlılar hariç, her canlı (bitki, hayvan, insan ve diğer) kendine özgü bir DNA yapısına sahiptir. Bundan dolayı gıda olarak kullanılan çiftlik hayvanları, meyve, sebze, baharatlar ve diğer otlar kendine özgü bir DNA yapısı ile tanımlanmaktadır. Bu ürünler pişmiş, çiğ veya 1-2 yıl muhafaza edilmiş, dondurulmuş olsalar bile DNA molekülleri değişmediğinden izole edilerek, veri bankasındaki DNA çipleri ile karşılaştırılması sonucu; ürünün orijini, çeşidi, ve kalitesi anlaşılmaktadır (Lenstra, 2006).

Türkiye'de gıda olarak tüketilen tüm bitkisel ve hayvansal gıda kaynaklarının DNA yapılarının tanımlanması ve bununla ilgili bir veri bankasının oluşturulması zarureti vardır. Bu şekilde hazırlanmış bir veri bankası sayesinde Türk gıda ürünleri dünyanın neresinde olursa olsun, birkaç saatlik bir test sonucunda anlaşılacaktır.

12.1.4.6. Coğrafi (GIS) / mekânsal (GPS) teknolojiler

Tarımsal üretimde EUREPGAP (Avrupa perakende sektöründe iyi tarım uygulamaları standardı), organik tarım ve diğer sertifikasyonlar; bahçe pasaportu, tarla pasaportu, toprak ve ürün analizleri, iklim verilerinin, izlenebilir olması için coğrafik ve mekânsal bilgi sistemi tabanına oturması gerekli olmaktadır. Diğer taraftan, konteyner takibi, kargo takibi, aynı şekilde GPS ile yapılabilmektedir.

Coğrafi Bilgi Sistemlerinin (GIS) entegrasyonu, Uzaktan Algılama (RS) ve Küresel Konumlama Sistemleri (GPS); çiftlikte ve takip eden işleme faaliyetlerinde, tarımsal ürünlerle ilgili verilerin elde edilmesi ve yere özgü tarım için önemli bir fırsat sunmaktadır. Bu teknolojiler hayvanlar ve bitkilerle ilgili verilerin uzaktan toplanmasına olanak vermektedir. İzlenebilirlikle ilgili olarak, bu teknolojilerin önemli bir özelliği, verim, ürün kalitesi, hayvan hareketi ve hastalık epidemiyolojisi gibi seçilmiş özelliklerin toprak/yüzey değişkenliğini planlayabilmesi ve haritasını yapabilmesidir (Kaplan and Hegarty, 2005).

13. MATERYAL METOT

13.1. Sistemin Tanımı

Taze Meyve ve Sebze İzlenebilirlik Sistemi (TAMSİS), gıda güvenliği ve kontrolünü sağlamak amacıyla gerçekleştirilen web tabanlı, merkezi sistem mimarisi ve merkezi veri tabanı yapısı üzerinde çalışacak olan bir izlenebilirlik uygulama sistemidir.

TAMSİS, iş süreçlerini, yazılım mimarisi ve diğer sistemlerle olan entegrasyonlar gibi temel tasarım konularını kapsamaktadır.

13.2. Temel Tasarım

Bu bölümde, TAMSİS'in temel sistem tasarım esasları ve bileşenleri ele alınmaktadır. Bu kapsamda, TAMSİS'in üzerinde çalışacağı yazılım mimarisi, raporlama ve dış sistemlerle entegrasyon gibi konuların nasıl kurgulandığı tariflenmektedir.

13.2.1. Yazılım mimarisi

Bu kısımda, geliştirilen TAMSİS uygulamasının yazılım bileşenlerine yönelik gereksinimler ile bu gereksinimlerden yola çıkılarak hazırlanmış yazılım mimarisi tasarımı yer almaktadır.

TAMSİS, web tabanlı, merkezi sunucu-istemci mimarisi ve merkezi veri tabanı yapısı ile çalışmaktadır. Sistem, tüm istemcilere merkezi uygulama sunucuları ve ilişkisel veri tabanı sunucuları ile hizmet verecektir. Yazılımın tüm işlevleri tamamen internet tarayıcısı arayüzü üzerinde çalışmaktadır. Kullanıcılar, uygulamayı web gezgini (browser) yazılımları aracılığı ile kullanabileceklerdir.

Uygulama yazılımı, web tabanlı teknolojiler kullanılacağı için performans ve ölçeklenebilirliğin arttırılması amacı ile n-katmanlı mimariye sahiptir. TAMSİS, J2EE/JEE (Java Enterprise Edition) standartlarına uygun olacak şekilde Java Teknolojileri kullanılarak kodlanmıştır.

Uygulama yazılımı genel amaçlı bir ilişkisel veri tabanı yönetim sistemi üzerinde çalışmaktadır. Endüstri standardı olan tüm ilişkisel veri tabanı sistemleri, uygulama

yazılımı tarafından desteklenmektedir ve gerektiğinde veri tabanı yönetim sisteminin değiştirilmesine imkan vermektedir.

TAMSİS, Türkçe karakter standardını desteklemekte ve Unicode karakter standardı kullanılarak geliştirilmiştir. Sistem ön tanımlı olarak Türkçe dil desteği ile kurulmuştur. Yazılımı oluşturan açık kodlar Ek'te yer almaktadır.

13.2.1.1. Yazılım katmanları

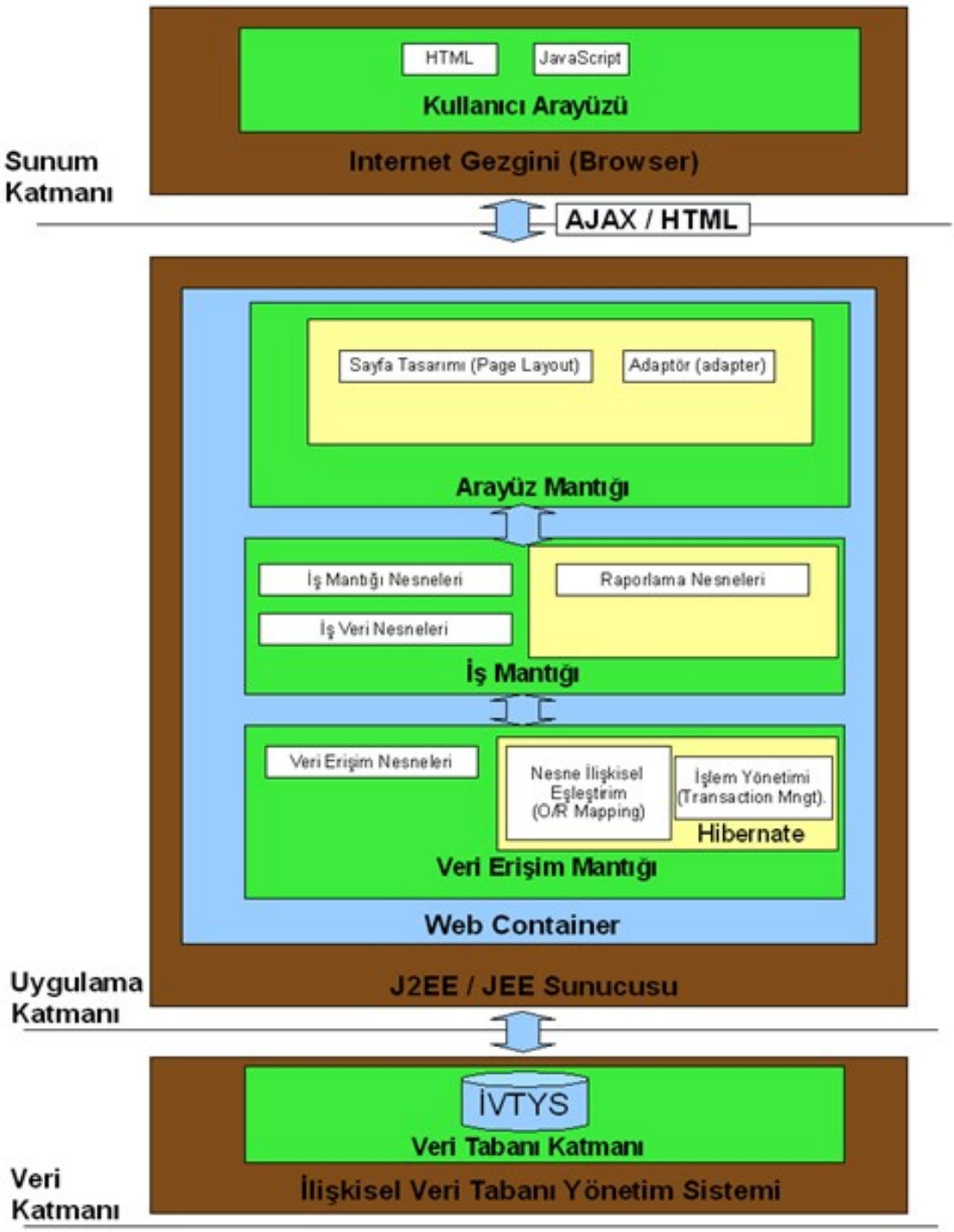
TAMSİS fiziksel olarak 3 katmanlı, mantıksal olarak ise n-katmanlı bir mimariye sahiptir (Şekil 3.1.).

Fiziksel Katmanlar

Fiziksel katmanlar, ölçeklenebilirliğin sağlanmasını mümkün kılan katmanlardır. Bu katmanlarda yeni kaynakların (uygulama sunucuları, vb.) talebe yönelik olarak çoğaltılması yolu ile performans yönetilebilecektir. TAMSİS fiziksel katmanları şunlardır: Sunum Katmanı, Uygulama Katmanı, Veri Katmanı.

Mantıksal Katmanlar

Mantıksal katmanlar, fiziksel katmanlar içerisindeki uygulama yazılımı işlevlerinin ayrıştırılmasına yönelik katmanlardır. Bu katmanlar sayesinde, uygulama yazılımı geliştirme sürecinde meydana gelebilecek değişikliklerin ya da daha ileride yeni işlevlerin eklenmesini kolaylaştıracak durumların yönetilmesi planlanmaktadır. Mantıksal açıdan TAMSİS, n-katmanlı mimariye uygun olarak gerçekleştirilecektir. TAMSİS mantıksal katmanları şunlardır: Kullanıcı Arayüzü, Arayüz Mantığı, İş Mantığı, Veri Erişim Mantığı ve Veri Tabanı Katmanı.



Şekil 3.1. TAMSİS Mantıksal Yazılım Katmanları Diyagramı

13.2.1.1.1. Sunum katmanı

Bu bölümde, TAMSİS yazılım mimarisinde bulunan sunum katmanındaki uygulama çatıları ve arayüz modelleri ile ilgili bilgiler yer almaktadır. Aşağıdaki kısımlarda, son kullanıcı ile yüksek etkileşimli iletişim kurulması, verimlilik ve

güvenilirliğin sağlanması için kullanılan yapılar ile temel kullanıcı arayüz bileşenleri tanımlanmaktadır.

Sunum Katmanı Geliştirme Platformu

Web üzerinde geliştirilen uygulamalarda kullanılan sunum katmanı, kullanıcı etkileşiminin gerçekleştirilmesini sağlar. Kullanıcıdan alınan girdilerin, sunucu katmanında yazılmış olan iş mantığına göre işlenmesi sonucunda oluşan çıktıların, internet üzerinden istemciye iletilmesiyle kullanıcı yaptığı işlemlerin sonucunu görmüş olur. Günümüzde sunum katmanında kullanılan, etkinliği ve kaliteyi arttıran uygulama çatıları da bulunmaktadır.

Temel Kullanıcı Arayüz Bileşenleri

TAMSİS sunum katmanında kullanılacak olan kullanıcı arayüz tasarımı, tarayıcı üzerinde kullanıcının tüm işlemlerini mümkün olduğunca tek pencere üzerinde gerçekleştirmesine olanak sağlayacak çoklu çerçeveli yapıyı temel almaktadır.

13.2.1.2. Uygulama katmanı

Bu bölümde, TAMSİS yazılım mimarisinde bulunan uygulama katmanındaki temel yapılar ile ilgili bilgiler yer almaktadır. Uygulama katmanı, TAMSİS uygulama sunucuları üzerinde çalışacak fiziksel bir katmandır. Yeni uygulama sunucuları eklenmesi ile ölçeklendirilebilecek bir yapıyı temsil etmektedir. Uygulama katmanı, kullanıcı arayüz bileşenlerinin sunucu tarafından kontrol edilmesi ve yönetilmesinden, kullanıcılar tarafından girilen bilgilerin doğrulanmasından, izlenebilirlik ile ilgili iş süreçlerinin yönetilmesinden ve veri katmanı ile bağlantı kurarak verilerin İlişkisel Veri Tabanı Yönetim Sistemi'ne (İVTYS) aktarılmasından sorumludur.

İş Mantığı Katmanı

TAMSİS kapsamında yer alan tüm iş süreçleri bu katman üzerinde yönetilir. İş Mantığı Katmanı, Arayüz Mantığı'ndan gelen kullanıcı işlemlerini modellenmiş iş süreçleri ile işleyip gerekli olan işlemleri yapmaktan, kullanıcılar tarafından girilen bilgilerin iş süreçlerinin gerektirdiği kurallar doğrultusunda doğrulamaktan ve raporlama işlevlerini gerçekleştirmekten sorumludur.

Veri Erişim Mantığı Katmanı

Veri Katmanı ile bağlantı kurup verilerin İVTYS'ne yazılması ve okunması ile ilgili tüm işlemlerden sorumlu olan katmandır. Bu katman üzerinde, Nesne İlişkisel Eşleştirim (Object Orient Relation Mapping (ORM)) teknikleri kullanılarak Hibernate yapısı üzerinden Veri Katmanı ile haberleşilecektir.

Hibernate, Java platformunda yazılmış bir ORM aracıdır. ORM, nesne odaklı (object oriented) dillerdeki nesnelerin, ilişkisel veritabanlarındaki (relational databases) kayıtlara nasıl karşılık geldiğini yürüten bir teknolojidir. Hibernate gibi ORM araçlarıyla, bir nesneyi veritabanına kaydetmek, yeni halini güncellemek ve sorgulama yapmak düz SQL bağlantılarına göre çok kolaydır. Hibernate gibi ORM araçlarının en önemli faydası, kod yazımını kısaltmak veya kolaylaştırmaktan öte, yazılım bakımını kolaylaştırmasıdır. Veritabanı temelli uygulamalarda, kodun 1/3´ü veritabanı erişimine yöneliktir. Hibernate kullanılan yazılımlarda, veritabanındaki değişikliklerde yapılması gereken sadece nesnelerle tabloların birbirine nasıl eşleştirildiğinin (mapping) gözden geçirilmesidir.

13.2.1.2.1. Veri katmanı

Bu bölümde, TAMSİS yazılım mimarisinde bulunan veri katmanındaki temel yapılar ile ilgili bilgiler yer almaktadır. Veri Katmanı, İVTYS'ni içeren fiziksel bir katmandır. Gerektiği durumlarda yeni veri tabanı sunucuları eklenmesi ile ölçeklendirilebilecek bir yapıyı temsil etmektedir.

Veri Tabanı Bileşenleri

Sunucu tarafında veri tabanı sunucusu yazılımı olacak ve bu yazılım aynı zamanda veri tabanı yönetim işlemlerini gerçekleştirme yeteneğine sahip araçları da içerecektir. Uygulama sunucusu ile veri tabanı arasındaki haberleşmeyi sağlayacak gerekli kütüphaneler uygulama sunucusu üzerine kurulacaktır.

13.2.2. Uygulama güvenliği

Bu kısımda, TAMSİS'de, uygulama düzeyinde alınması tasarlanan güvenlik tedbirleri, kimlik denetimi ve yetkilendirme tariflenmektedir.

Kimlik Denetimi (Authentication)

Sisteme giriş yaparken kimlik doğrulama, kullanıcıdan, kullanıcı adı ve paralo gibi kimlik bilgilerini alıp bu bilgilerin geçerliliğini denetleme ve doğrulama işlemidir. Girilen kimlik bilgileri geçerliyse, ilgili kullanıcı bundan sonraki işlemlerde doğrulanmış bir kimliğe sahip kullanıcı olarak oturum açması / oturuma devam etmesine izin verilir. Bir kimlik doğrulandıktan sonra, yetkilendirme sorgulaması vasıtasıyla ilgili kimliğin hangi kaynaklara erişim izni olup olmadığı saptanır.

Yetkilendirme (Authorization)

Sistemde, kullanıcıların hangi ekran ve işlemleri kullanmaya yetkili olduğunu belirleme ve uygulamada kullanıcının kullanabileceği fonksiyonları buna göre düzenleme işlemidir.

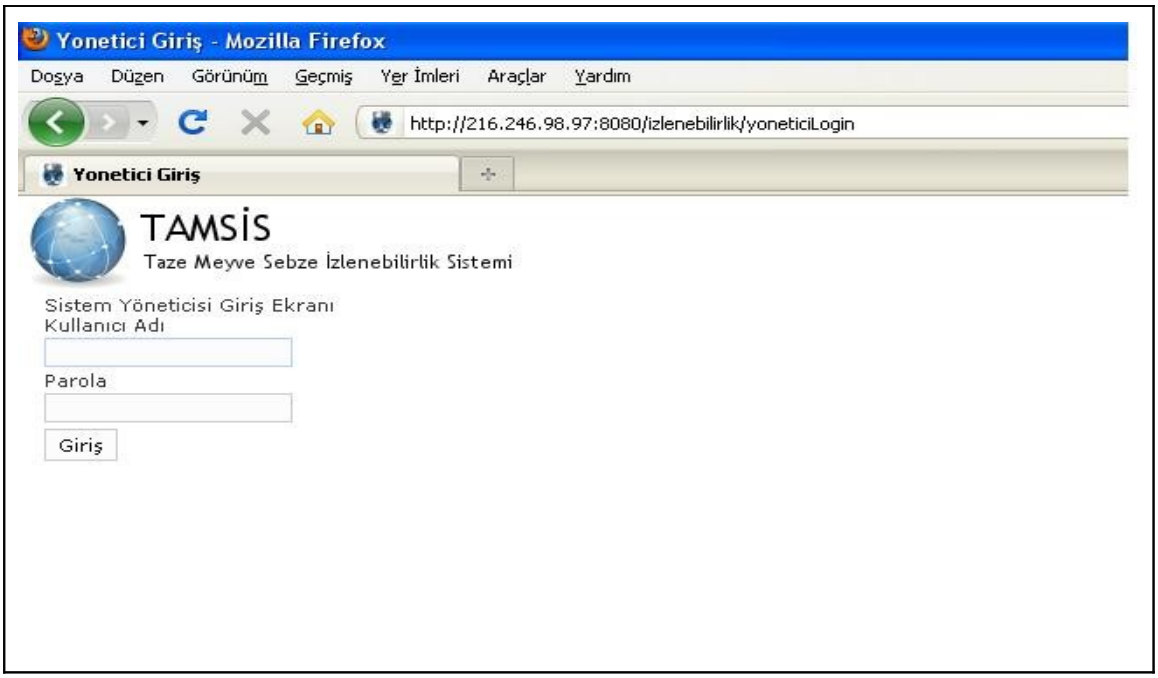
13.3. Sistemin Mevcut Ekranları

TAMSİS Ana Sayfasında (Şekil 3.2.) her bir kullanıcının sisteme giriş yapabilmesi amacıyla kendine ait bağlantıya girmesi gerekmektedir.

Şekil 3.2. TAMSİS Ana Sayfa

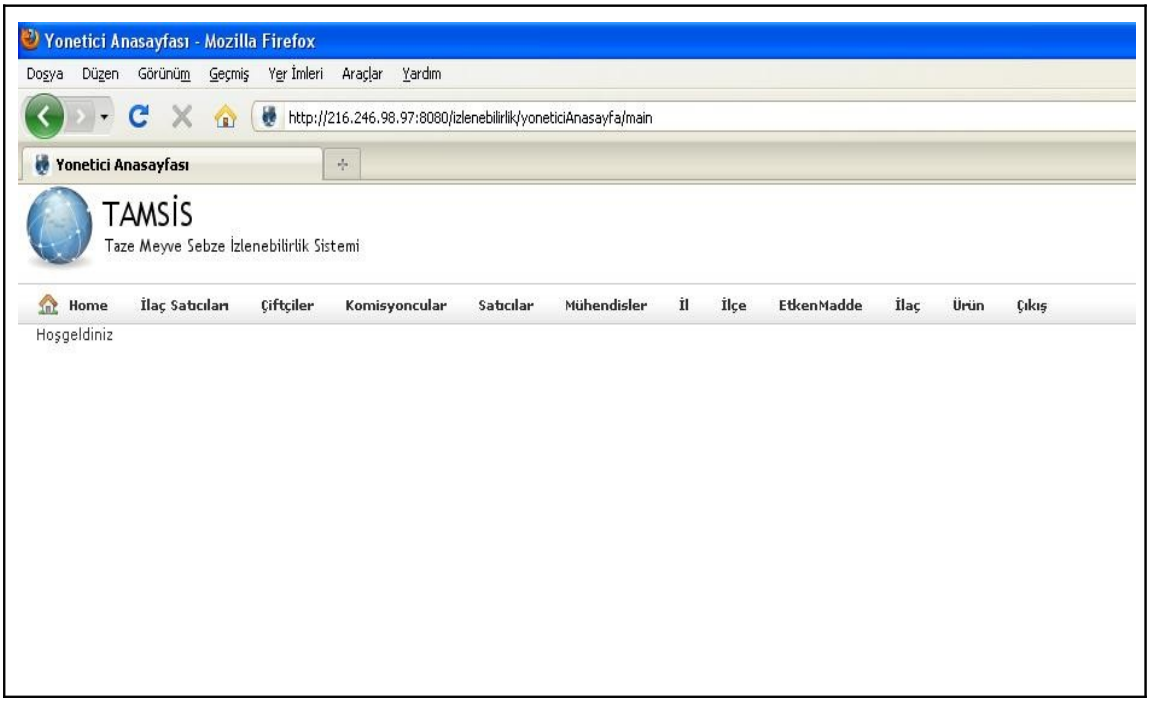
3.3.1. Sistem yöneticisi

Şekil 3.2.'de verilen TAMSİS ana sayfasındaki "Sistem Yöneticisi" bağlantısına tıklandığında sistem yöneticisi giriş ekranı (Şekil 3.3.) gelmektedir. Ekranda kullanıcı adı ve parola kutuları doldurularak sisteme giriş yapılmaktadır.



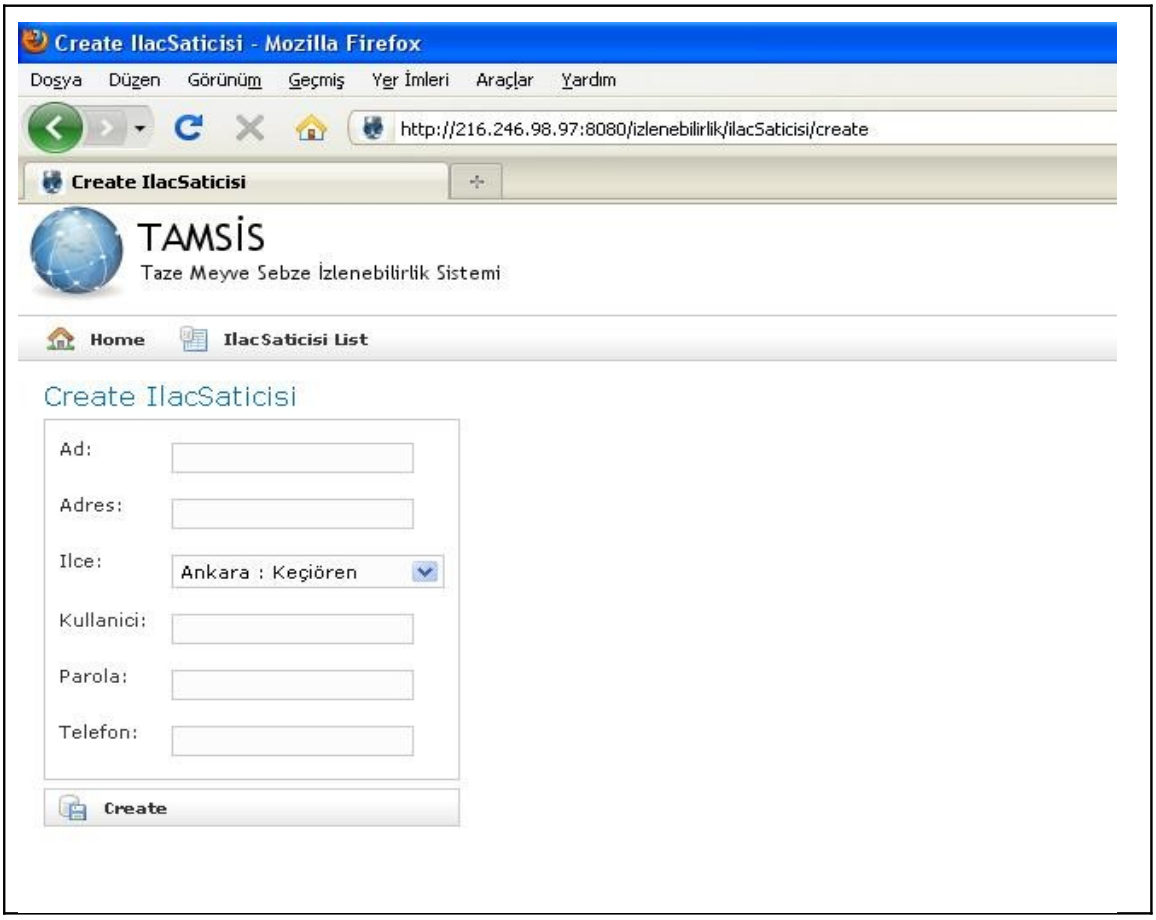
Şekil 3.3. Sistem Yöneticisi Giriş Ekranı

Bu adımda Şekil 3.4.'deki "Sistem Yöneticisi Ana Ekranı"na ulaşılmaktadır. Bu sayfa sadece sistem yöneticisi tarafından kullanılabilmektedir. Sistem yöneticisi; ilaç satıcıları, çiftçiler, komisyoncular, satıcılar ve mühendisler ile ilgili tüm bilgileri bu sayfadan yönetmekte; ayrıca il, ilçe, ilaç ve etken maddeler gibi temel verileri izlemektedir.



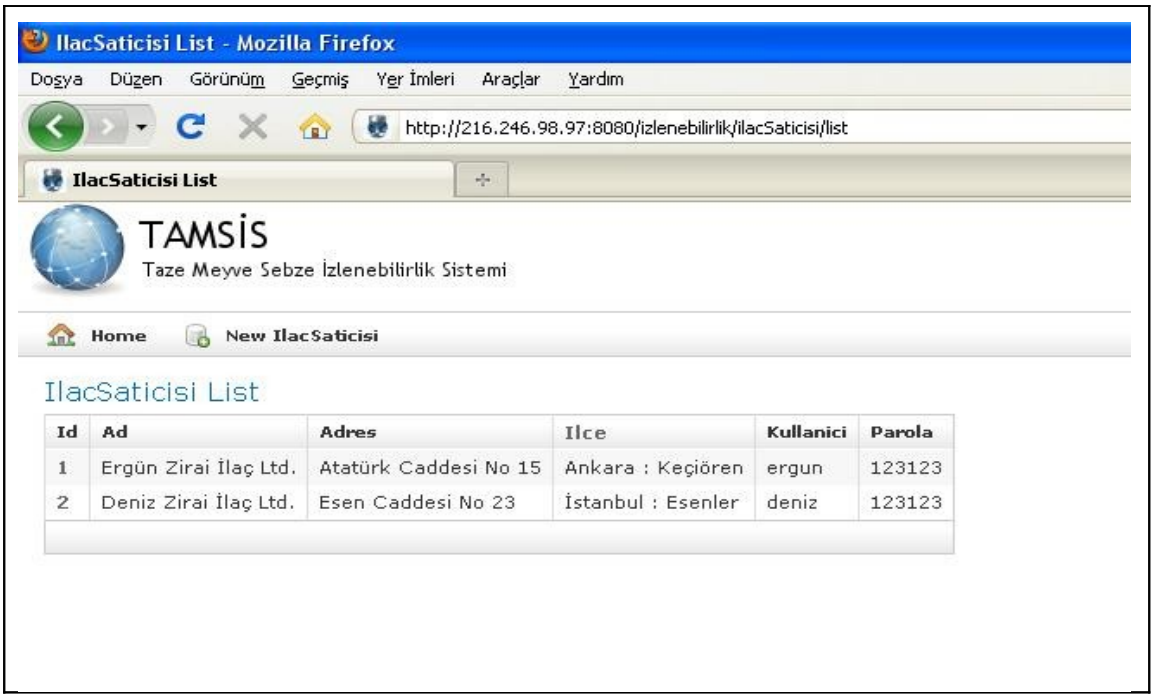
Şekil 3.4. Sistem Yöneticisi Ana Ekranı

Sistem Yöneticisi ana ekranındaki “İlaç Satıcıları” sekmesine tıkladığında yeni bir ilaç satıcısının bilgilerinin girileceği ekran (Şekil 3.5.) gelmektedir. Bu ekranda ilaç satıcısına ait ad, adres, kullanıcı adı, parola ve telefon bilgileri sistem yöneticisi tarafından sisteme girilmektedir.



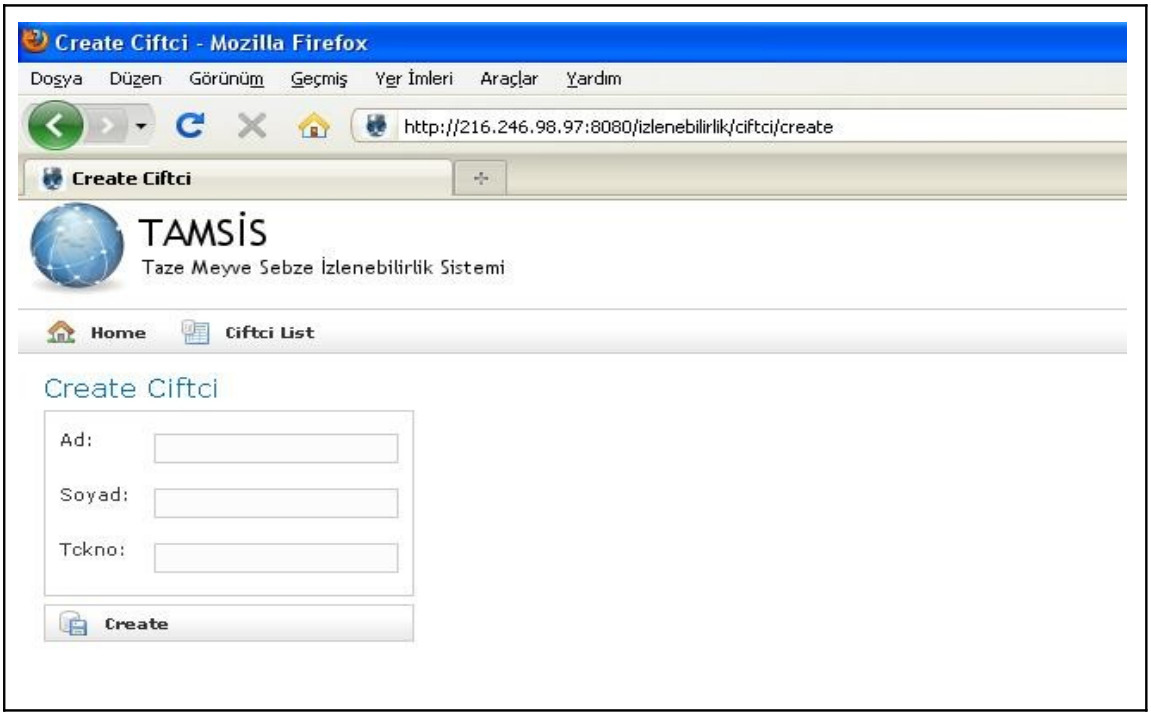
Şekil 3.5. Sistem Yöneticisi Yeni İlaç Satıcı Listesi Giriş Ekranı

İlaç satıcısı ekranında yeni ilaç satıcısı bilgisi eklendikten sonra İlaç Satıcıları Listesine tıklandığında o ana kadar sistem yöneticisi tarafından sisteme girilmiş olan ilaç satıcıları listeye yeni eklenen satıcıyla birlikte tüm liste halinde ekrana (Şekil 3.6.) gelmektedir. Sistem yöneticisinin sekmelere ulaşabilmesi için Şekil 3.4.’de verilen “Sistem Yöneticisi Ana Ekranı”na dönüş yapması gerekmektedir.



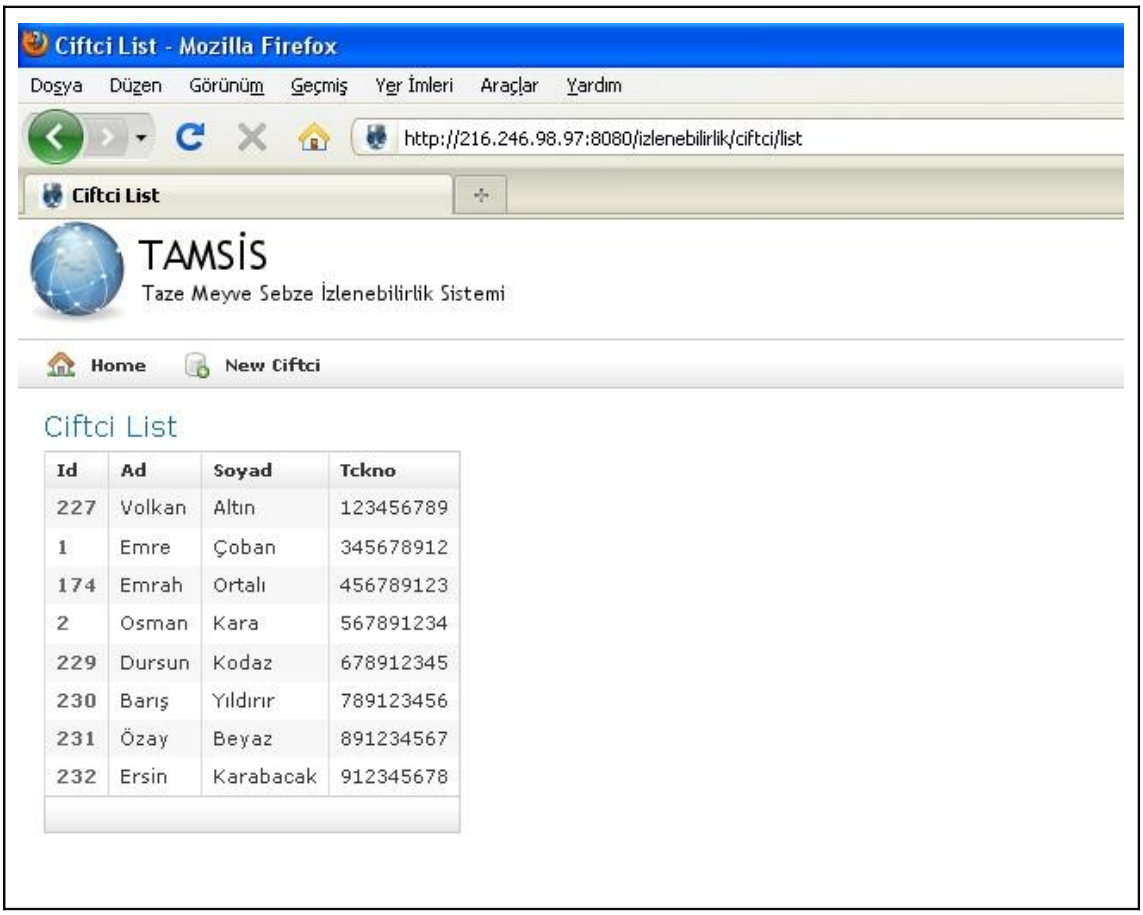
Şekil 3.6. Sistem Yöneticisi İlaç Satıcıları Liste Ekranı

Sistem Yöneticisi ana ekranındaki “Çiftçiler” sekmesine tıklandığında yeni bir çiftinin girileceği ekran (Şekil 3.7.) gelmektedir. Bu ekranda çiftçiye ait ad, soyad ve Türkiye Cumhuriyeti vatandaşlık numarası bilgileri sistem yöneticisi tarafından sisteme girilmektedir.



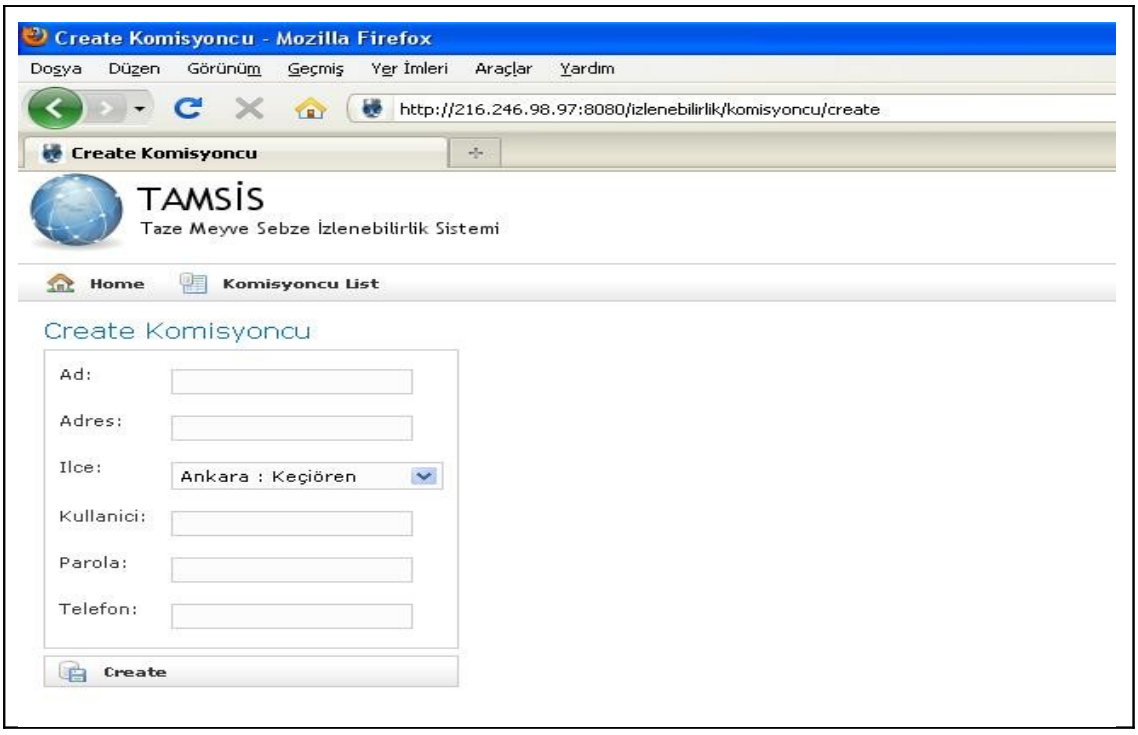
Şekil 3.7. Sistem Yöneticisi Yeni Çiftçi Kayıt Ekranı

Çiftçiler ekranında yeni çiftçi bilgisi eklendikten sonra Çiftçiler Listesine tıklandığında o ana kadar sistem yöneticisi tarafından sisteme girilmiş olan çiftçiler listesi yeni eklenen çiftçiyle birlikte tüm liste halinde ekrana (Şekil 3.8.) gelmektedir. Sistem yöneticisinin sekmelere ulaşabilmesi için Şekil 3.4.'de verilen "Sistem Yöneticisi Ana Ekranı"na dönüş yapması gerekmektedir.



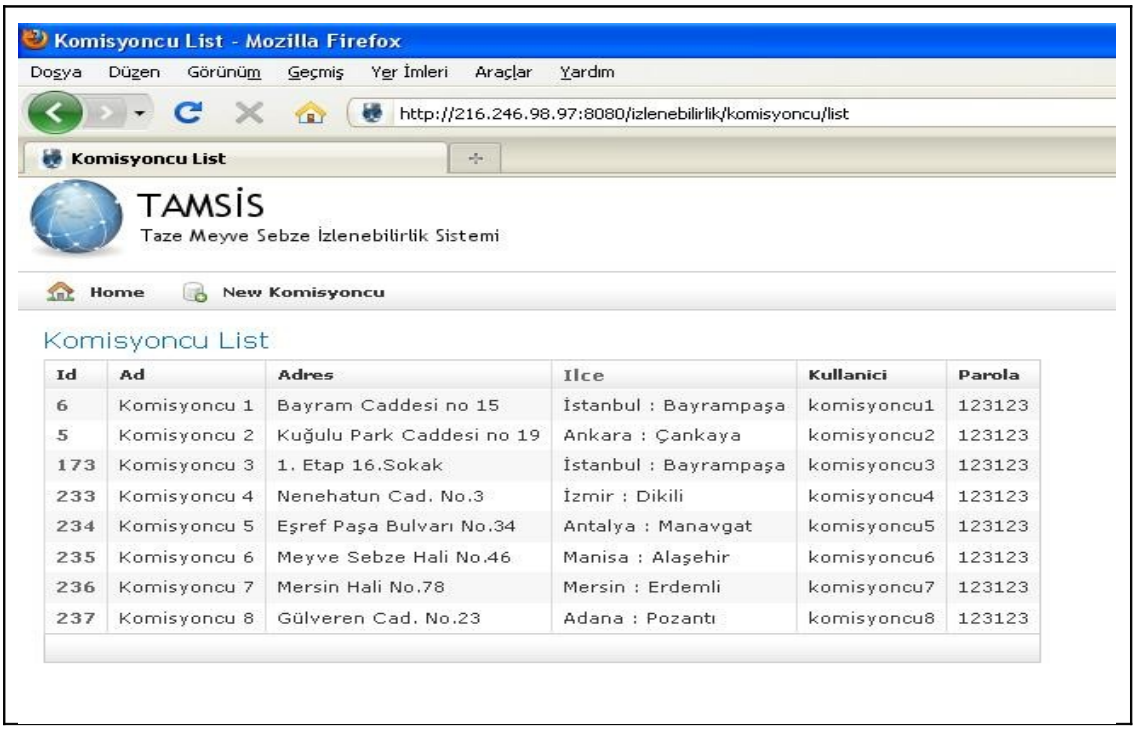
Şekil 3.8. Sistem Yöneticisi Çiftçi Listesi Ekranı

Sistem Yöneticisi ana ekranındaki "Komisyoncular" sekmesine tıklandığında yeni bir komisyoncunun girileceği ekran (Şekil 3.9.) gelmektedir. Bu ekranda komisyoncuya ait ad, adres, kullanıcı adı, parola ve telefon bilgileri sistem yöneticisi tarafından sisteme girilmektedir.



Şekil 3.9. Sistem Yöneticisi Yeni Komisyoncu Kayıt Ekranı

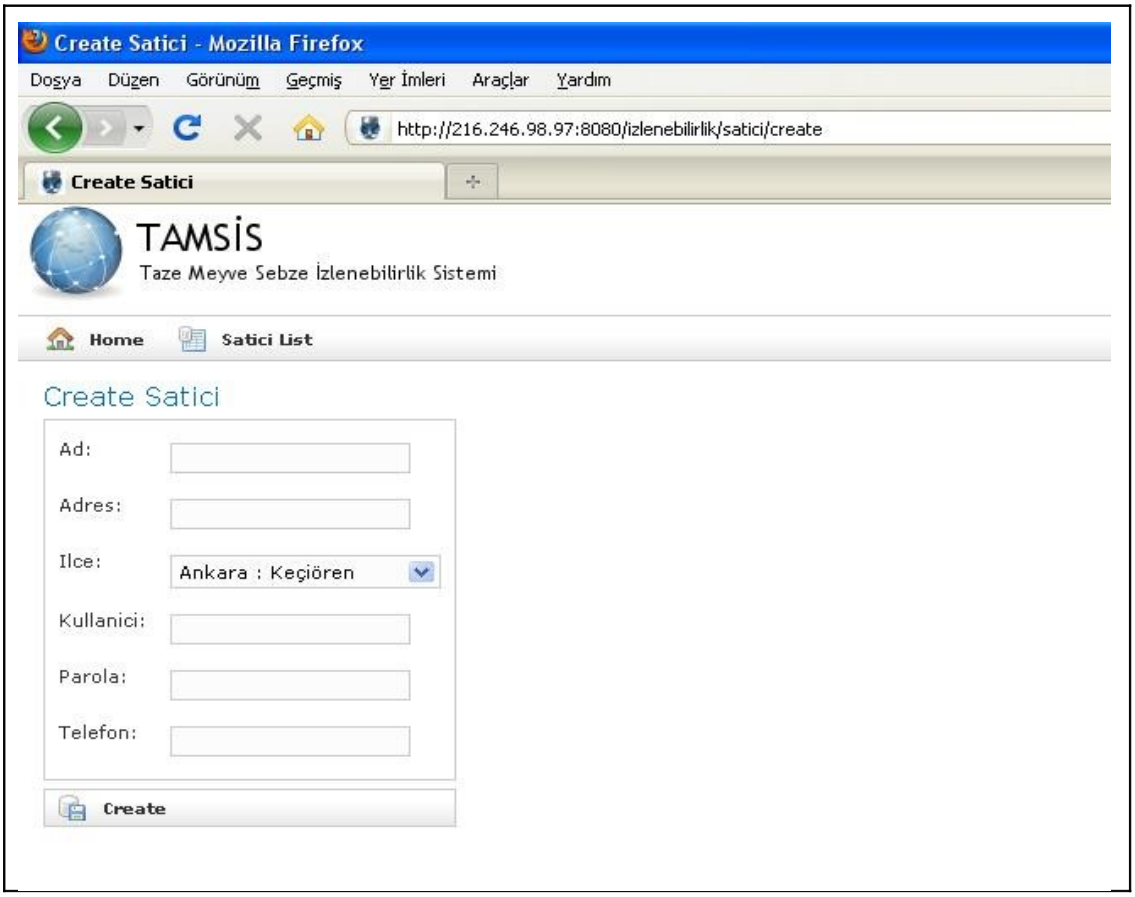
Komisyoncu ekranında yeni komisyoncular eklendikten sonra Komisyoncular Listesine tıklandığında o ana kadar sistem yöneticisi tarafından sisteme girilmiş olan komisyoncuların bilgileri ekrana (Şekil 3.10.) gelmektedir.



Id	Ad	Adres	Ilce	Kullanici	Parola
6	Komisyoncu 1	Bayram Caddesi no 15	İstanbul : Bayrampaşa	komisyoncu1	123123
5	Komisyoncu 2	Kuğulu Park Caddesi no 19	Ankara : Çankaya	komisyoncu2	123123
173	Komisyoncu 3	1. Etap 16.Sokak	İstanbul : Bayrampaşa	komisyoncu3	123123
233	Komisyoncu 4	Nenehatun Cad. No.3	İzmir : Dikili	komisyoncu4	123123
234	Komisyoncu 5	Eşref Paşa Bulvarı No.34	Antalya : Manavgat	komisyoncu5	123123
235	Komisyoncu 6	Meyve Sebze Hali No.46	Manisa : Alaşehir	komisyoncu6	123123
236	Komisyoncu 7	Mersin Hali No.78	Mersin : Erdemli	komisyoncu7	123123
237	Komisyoncu 8	Gülveren Cad. No.23	Adana : Pozantı	komisyoncu8	123123

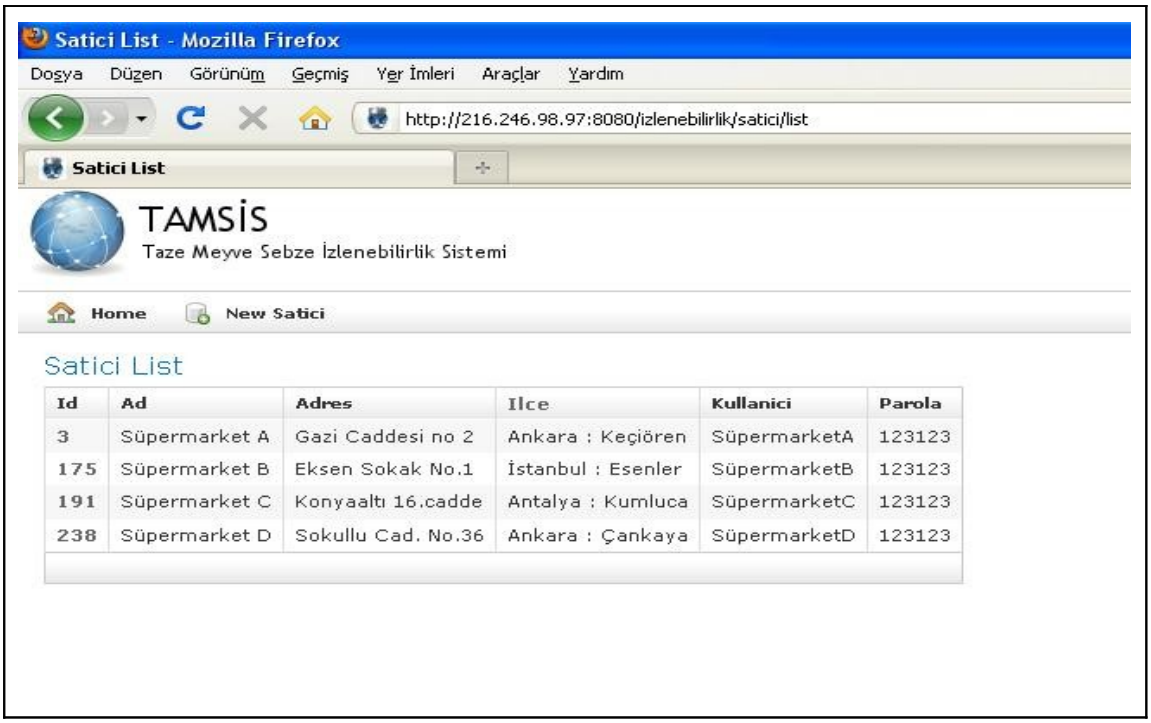
Şekil 3.10. Sistem Yöneticisi Komisyoncu Listesi Ekranı

Sistem Yöneticisi ana ekranındaki Satıcılar sekmesine tıklandığında yeni bir satıcının girileceği ekran (Şekil 3.11.) gelmektedir. Bu ekranda satıcıya ait ad, adres, kullanıcı adı, parola ve telefon bilgileri sistem yöneticisi tarafından sisteme girilmektedir.



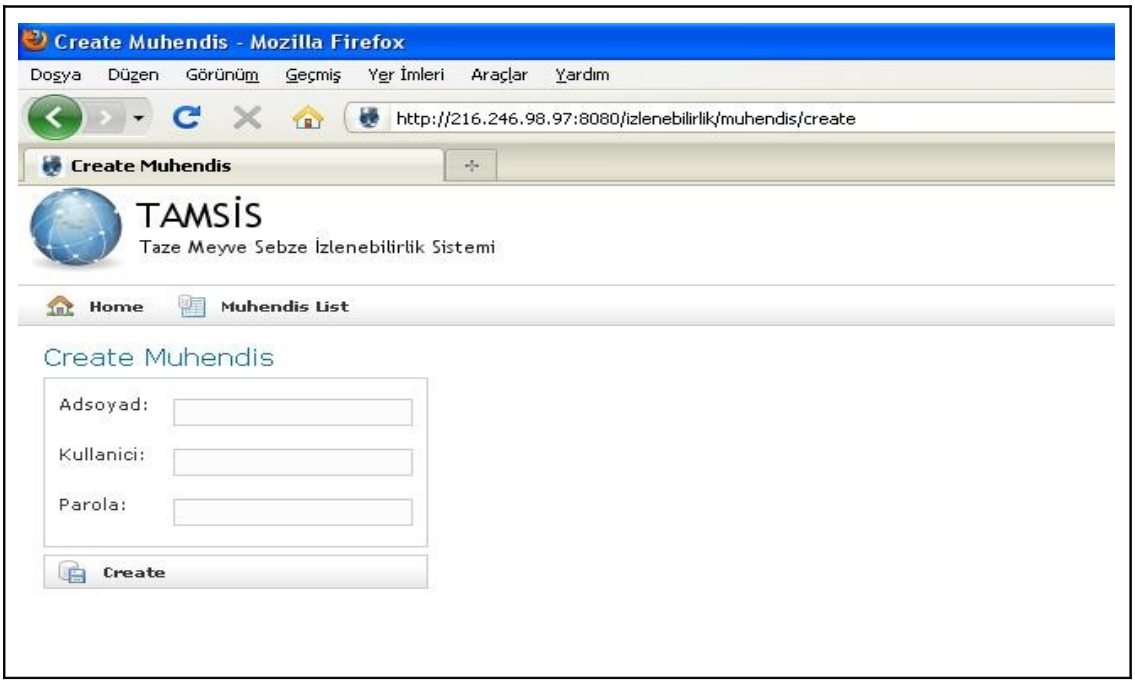
Şekil 3.11. Sistem Yöneticisi Yeni Satıcı Kayıt Ekranı

Satıcı ekranında yeni satıcılar eklendikten sonra Satıcılar Listesine tıklandığında o ana kadar sistem yöneticisi tarafından sisteme girilmiş olan satıcılar (Şekil 3.12.) yer almaktadır.



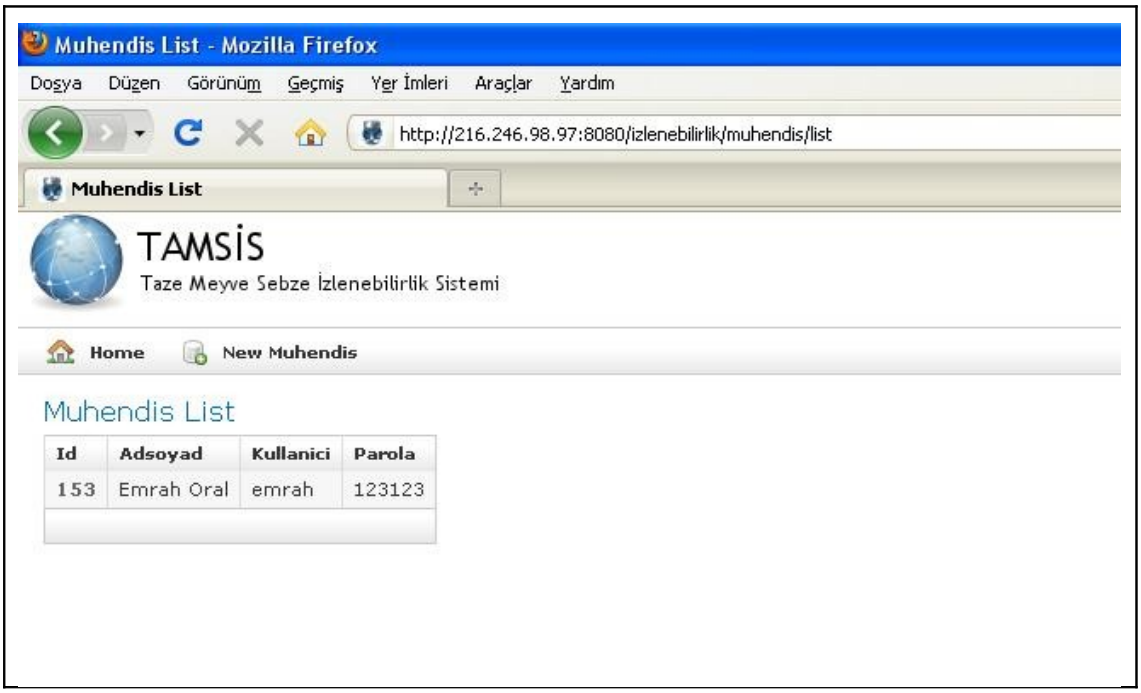
Şekil 3.12. Sistem Yöneticisi Satıcı Listesi Ekranı

Sistem Yöneticisi Ana Ekranındaki Mühendisler sekmesine tıklandığında yeni bir mühendisin girileceği ekran (Şekil 3.13.) gelmektedir. Bu ekranda sistemden rapor alacak mühendise ait ad, soyad ve parola bilgileri sistem yöneticisi tarafından sisteme girilmektedir.



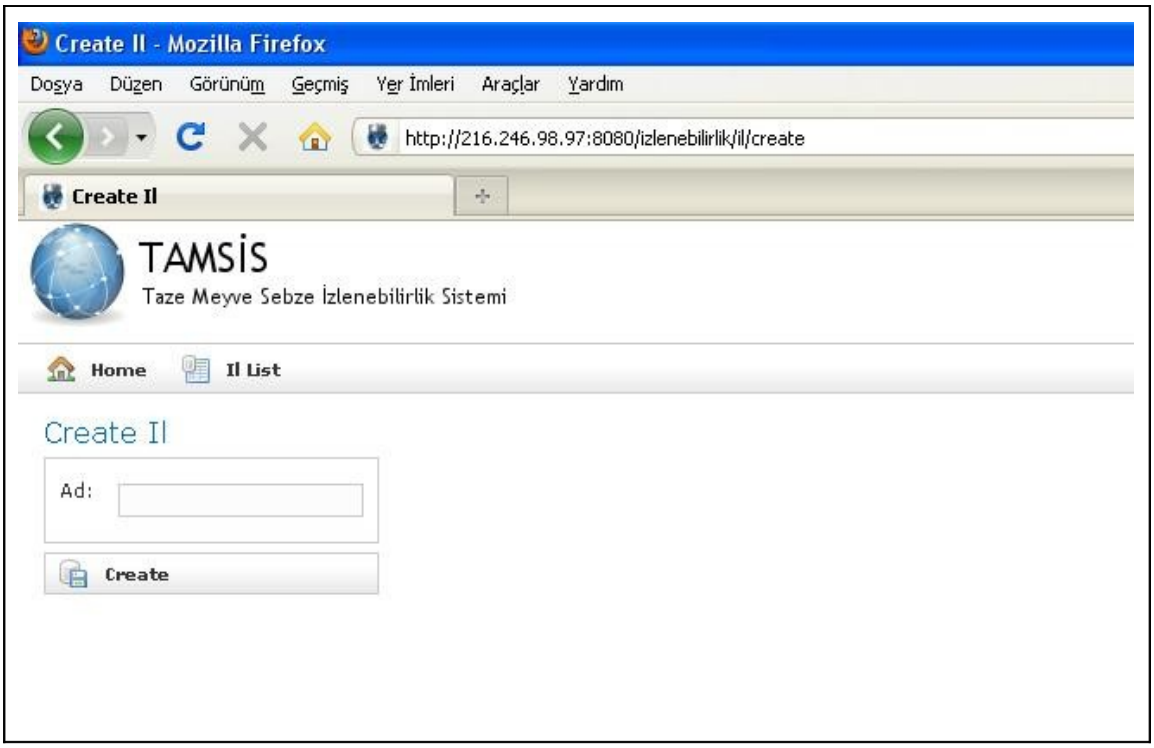
Şekil 3.13. Sistem Yöneticisi Mühendis Yeni Kayıt Ekranı

Mühendis ekranında sistemi kullanacak yeni mühendisler eklendikten sonra Mühendisler Listesine tıklandığında o ana kadar sistem yöneticisi tarafından sisteme girilmiş olan mühendisler (Şekil 3.14.) yer almaktadır.

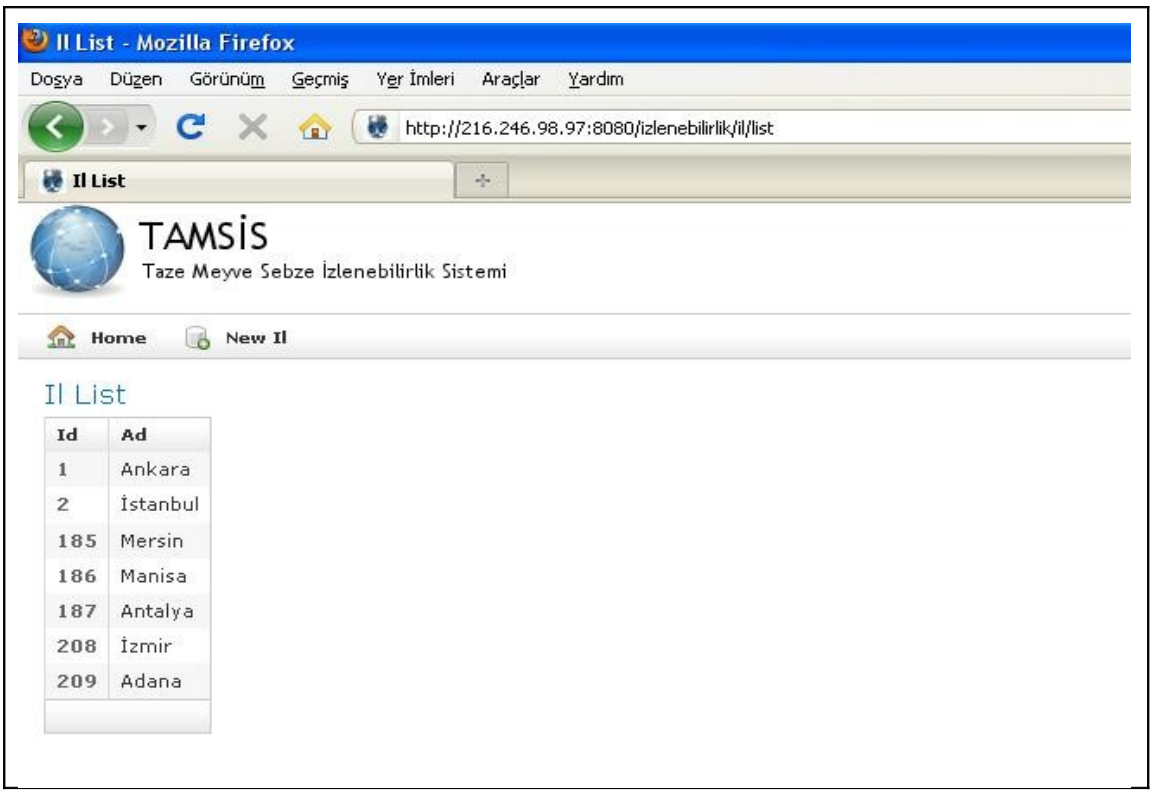


Şekil 3.14. Sistem Yöneticisi Mühendis Listesi Ekranı

Sistem Yöneticisi Ana Ekranındaki İl sekmesine tıklandığında yeni bir ilin girileceği ekran (Şekil 3.15.) gelmektedir. Bu ekran temel veri ekranı olup sistem yöneticisi tarafından sisteme girilmektedir. İl listesine tıklandığında Şekil 3.16. ekrana gelmektedir.

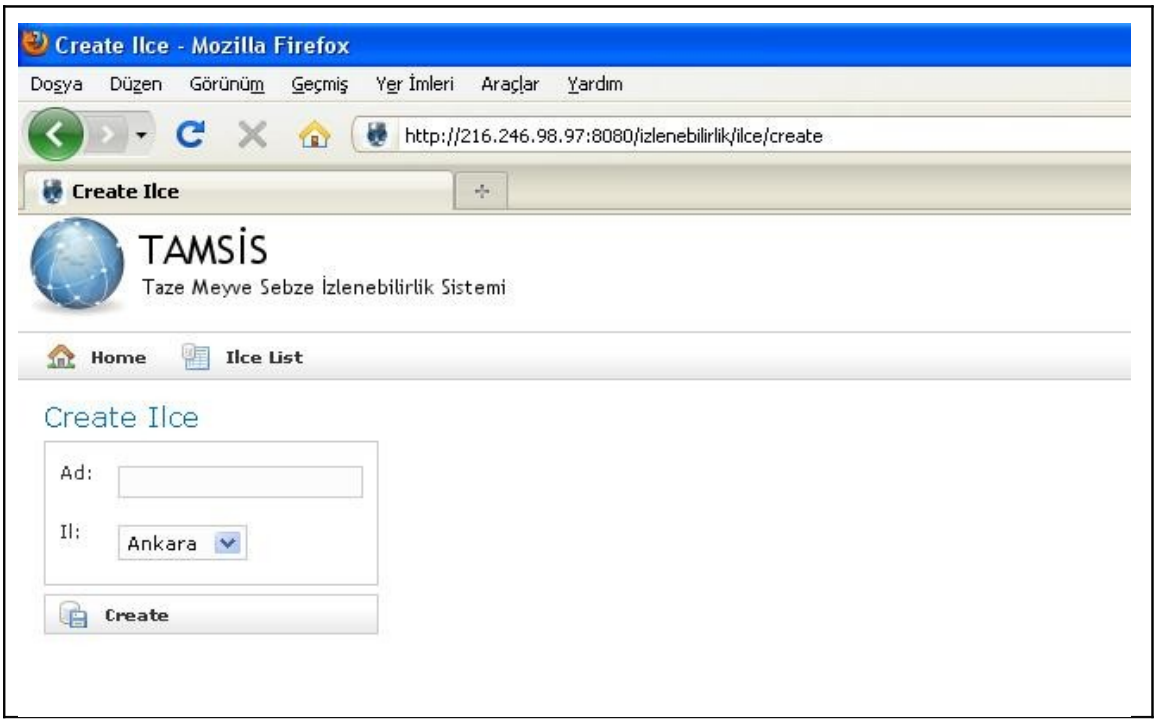


Şekil 3.15. Sistem Yöneticisi İl Giriş Ekranı

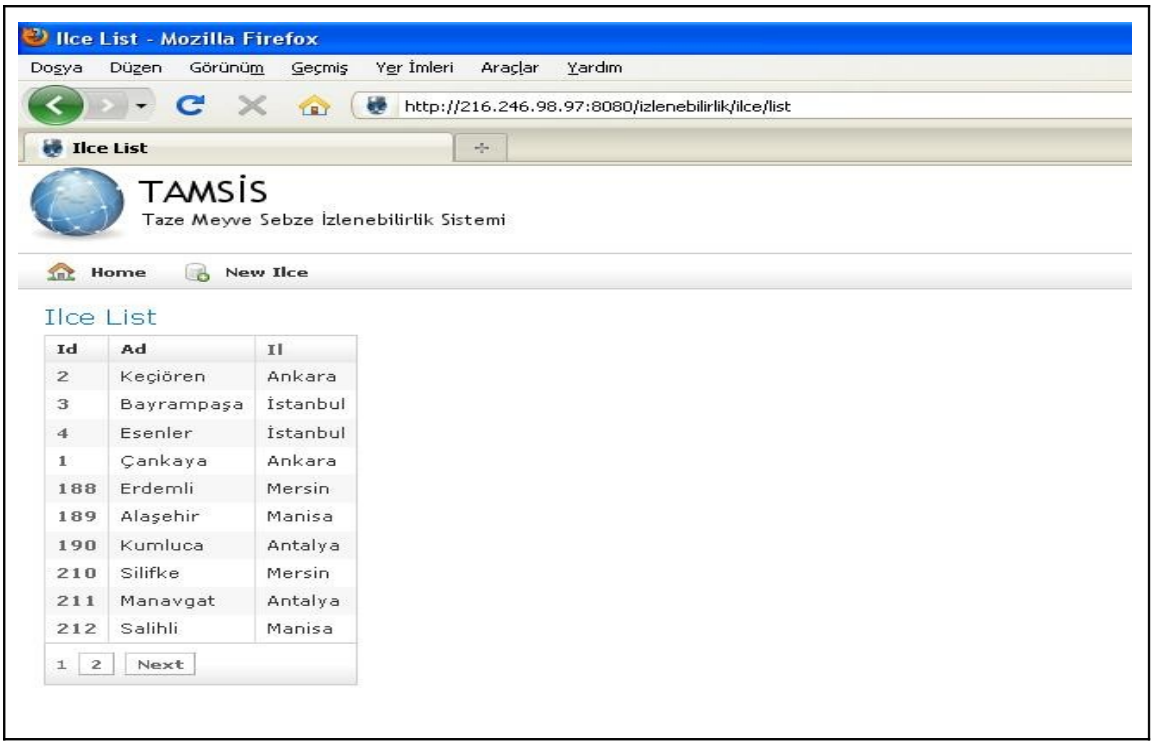


Şekil 3.16. Sistem Yöneticisi İl Listesi Ekranı

Sistem Yöneticisi Ana Ekranındaki İlçe sekmesine tıklandığında yeni bir ilçenin girileceği ekran (Şekil 3.17.) gelmektedir. Bu ekran temel veri ekranı olup sistem yöneticisi tarafından sisteme girilmektedir.

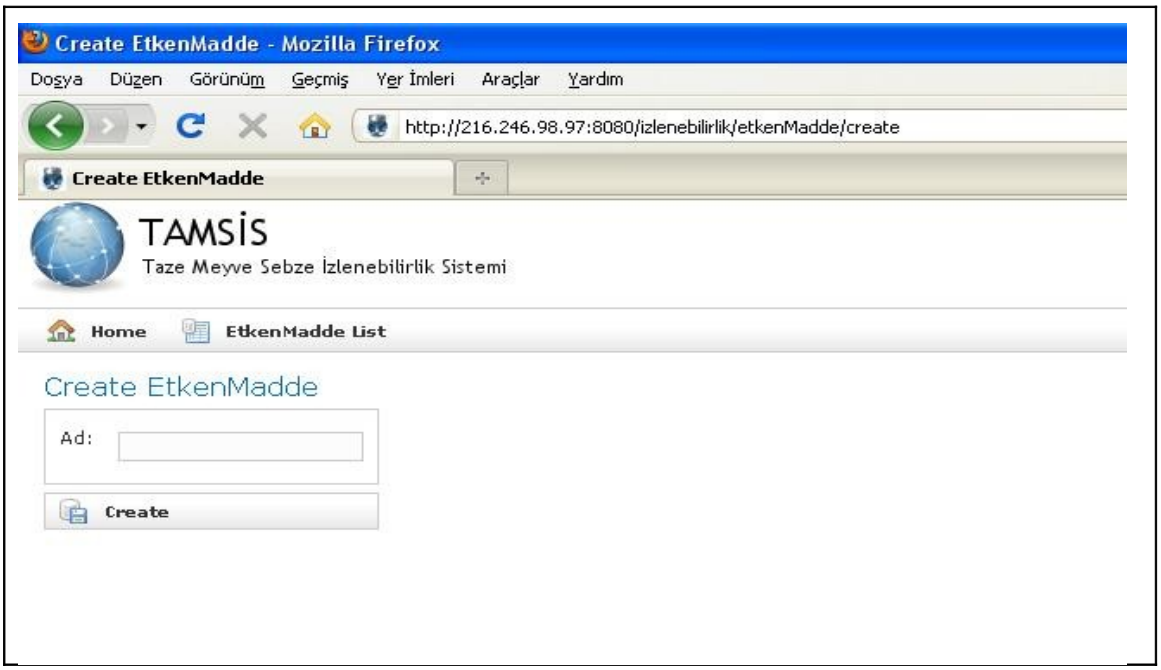


Şekil 3.17. Sistem Yöneticisi İlçe Giriş Ekranı

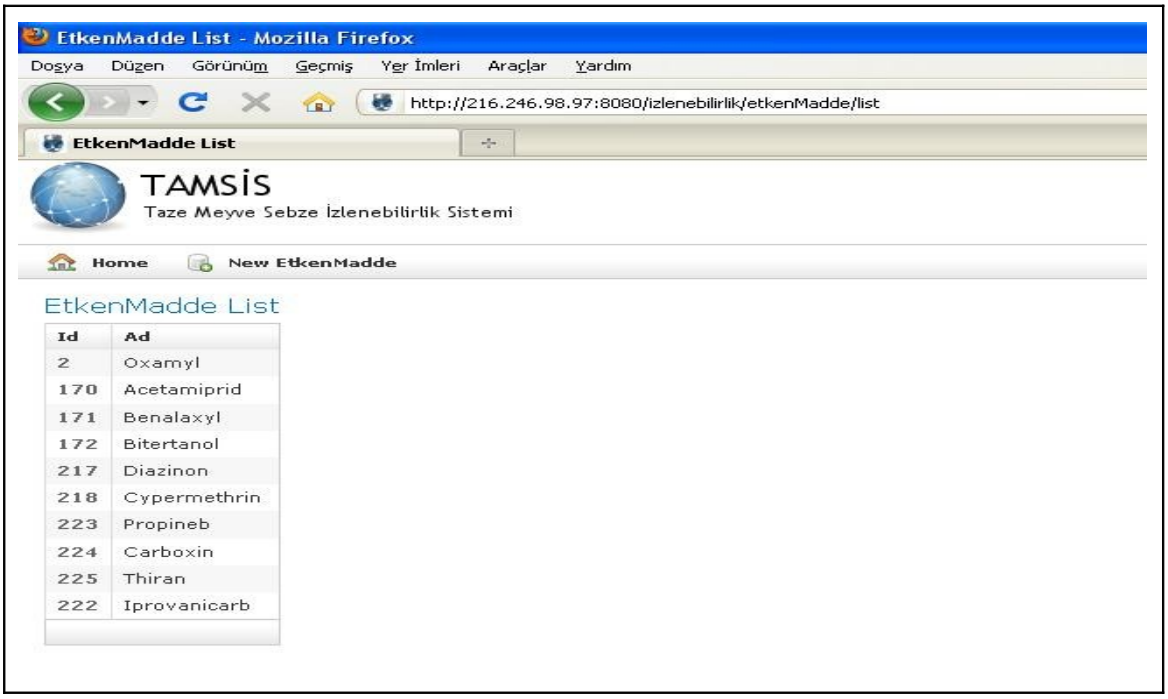


Şekil 3.18. Sistem Yöneticisi İlçe Listesi Ekranı

Sistem Yöneticisi Ana Ekranındaki Etken Madde sekmesine tıklandığında yeni bir etken maddenin girileceği ekran (Şekil 3.19.) gelmektedir. Bu ekran temel veri ekranı olup sistem yöneticisi tarafından sisteme girilmektedir. Etken madde listesi (Şekil 3.20.) tümüyle görülmekte ve etken maddeye tıklandığında hangi ruhsatlı ilaçlarda bulundukları ilaç sekmesine gidilmeden görülebilmektedir.

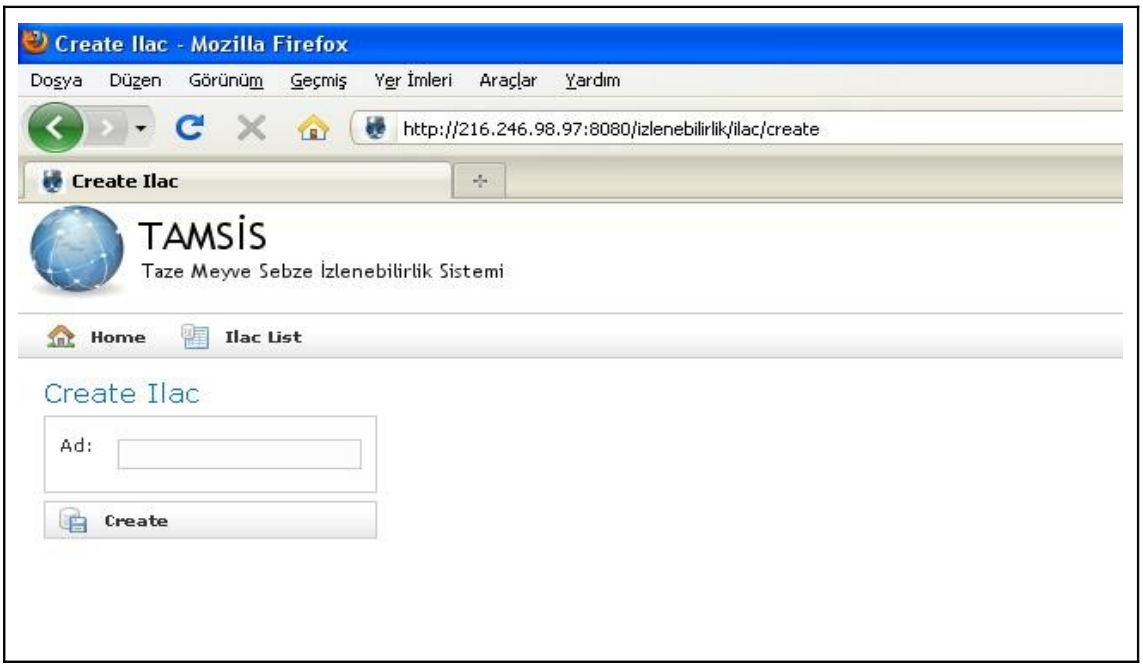


Şekil 3.19. Sistem Yöneticisi Etken Madde Giriş Ekranı

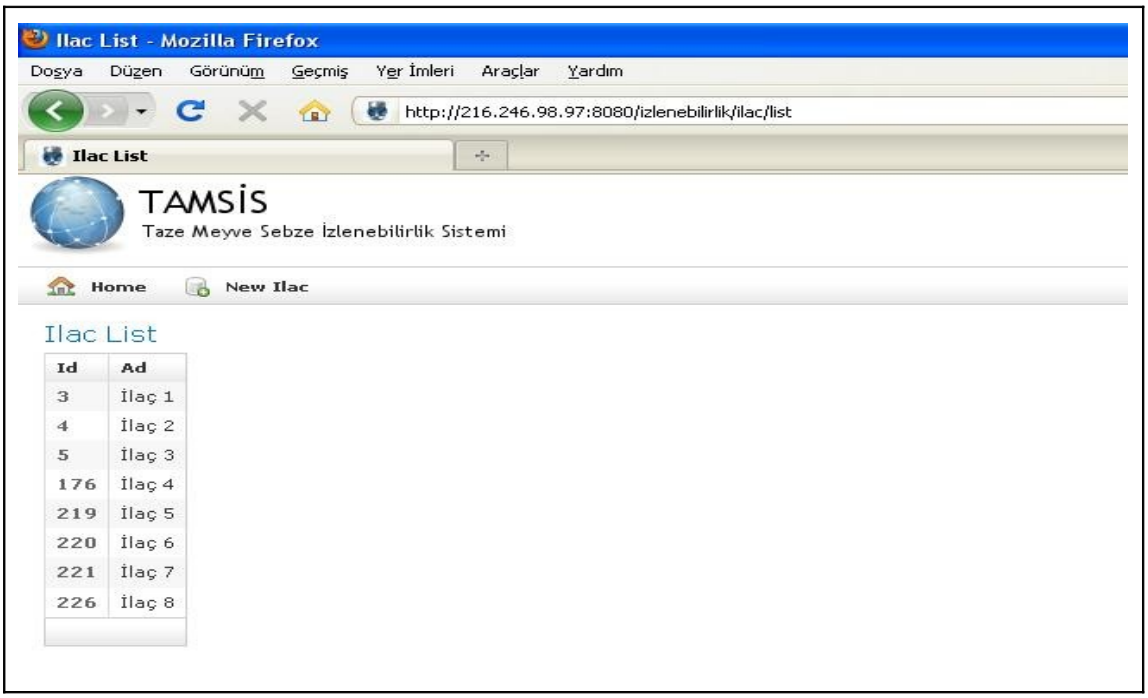


Şekil 3.20. Sistem Yöneticisi Etken Madde Listesi Ekranı

Sistem Yöneticisi Ana Ekranındaki İlaç sekmesine tıklandığında yeni bir ruhsatlı ilacın girileceği ekran (Şekil 3.21.) gelmektedir. Bu ekran temel veri ekranı olup sistem yöneticisi tarafından sisteme girilmektedir. Ruhsatlı ilaç listesi (Şekil 3.22.) tümüyle görülmekte ve ruhsatlı ilaçlar sisteme girilirken hangi etken maddeyi içerdikleri girilmektedir.

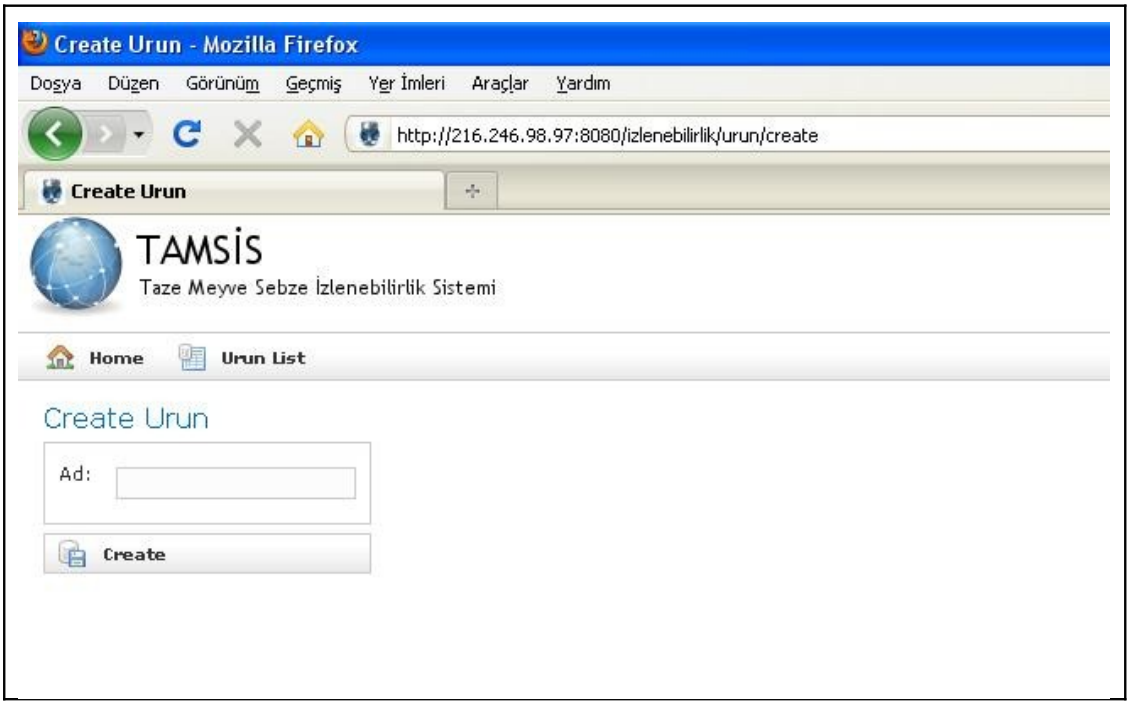


Şekil 3.21. Sistem Yöneticisi İlaç Giriş Ekranı

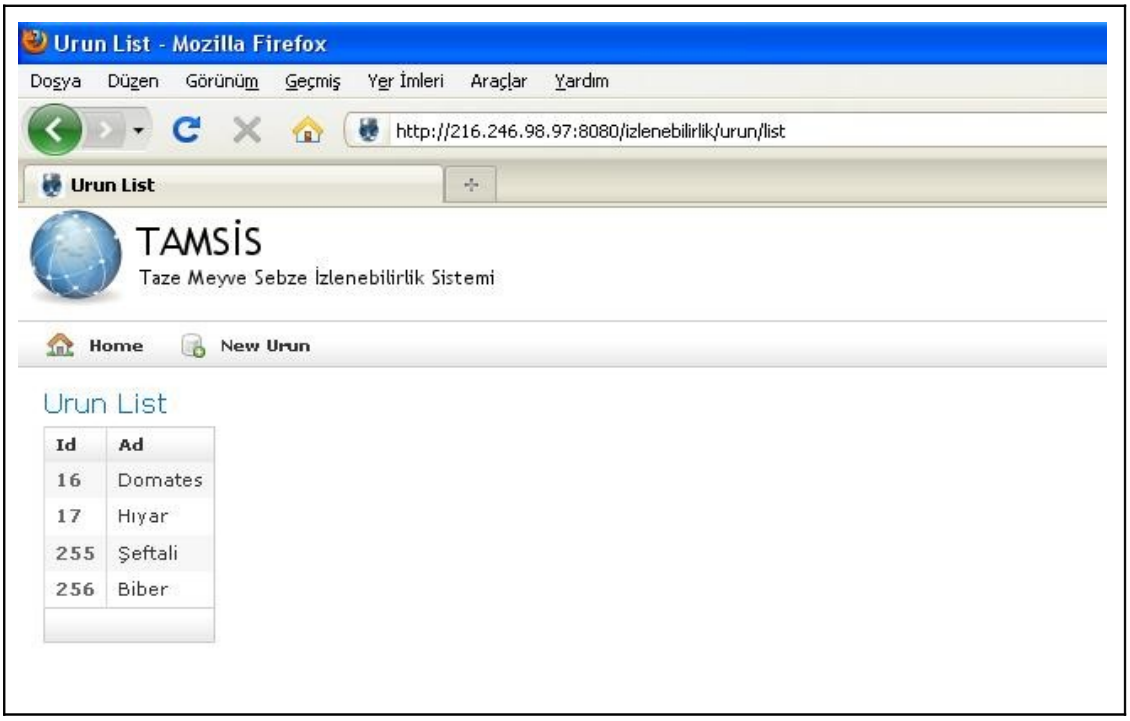


Şekil 3.22. Sistem Yöneticisi İlaç Listesi Ekranı

Sistem Yöneticisi Ana Ekranındaki Ürün sekmesine tıklandığında yeni bir ürünün girileceği ekran (Şekil 3.23.) gelmektedir. Bu ekran temel veri ekranı olup sistem yöneticisi tarafından sisteme girilmekte olup ürünlerin tümü ürün listesine (Şekil 3.24.) tıklanarak görülmektedir.



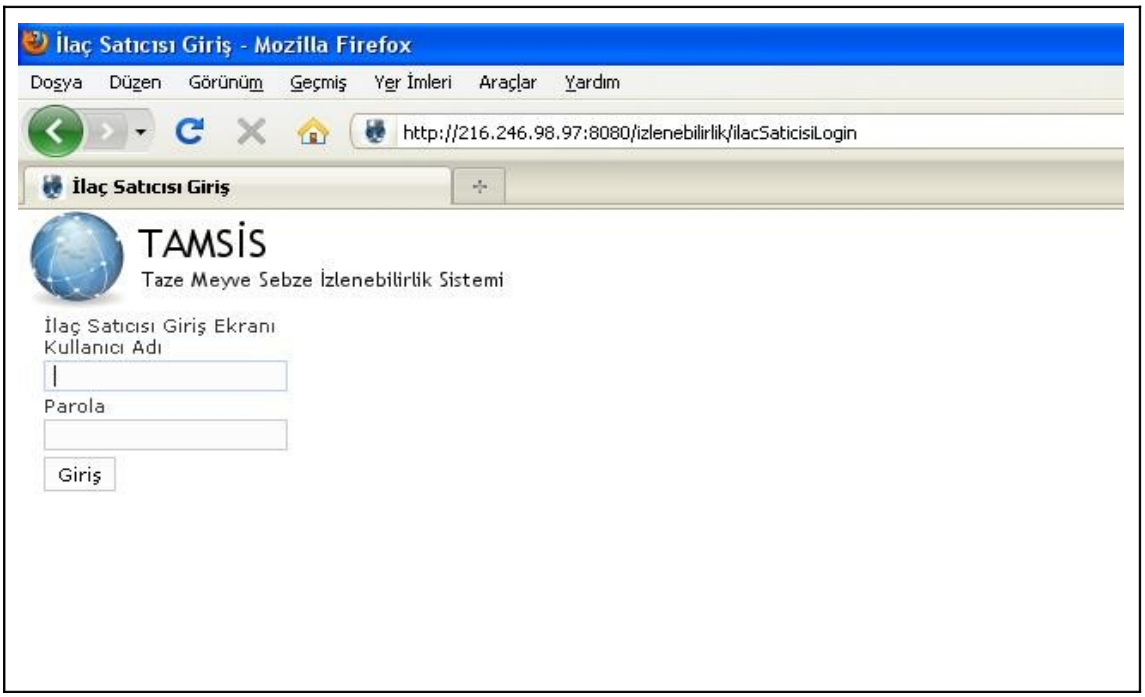
Şekil 3.23. Sistem Yöneticisi Ürün Giriş Ekranı



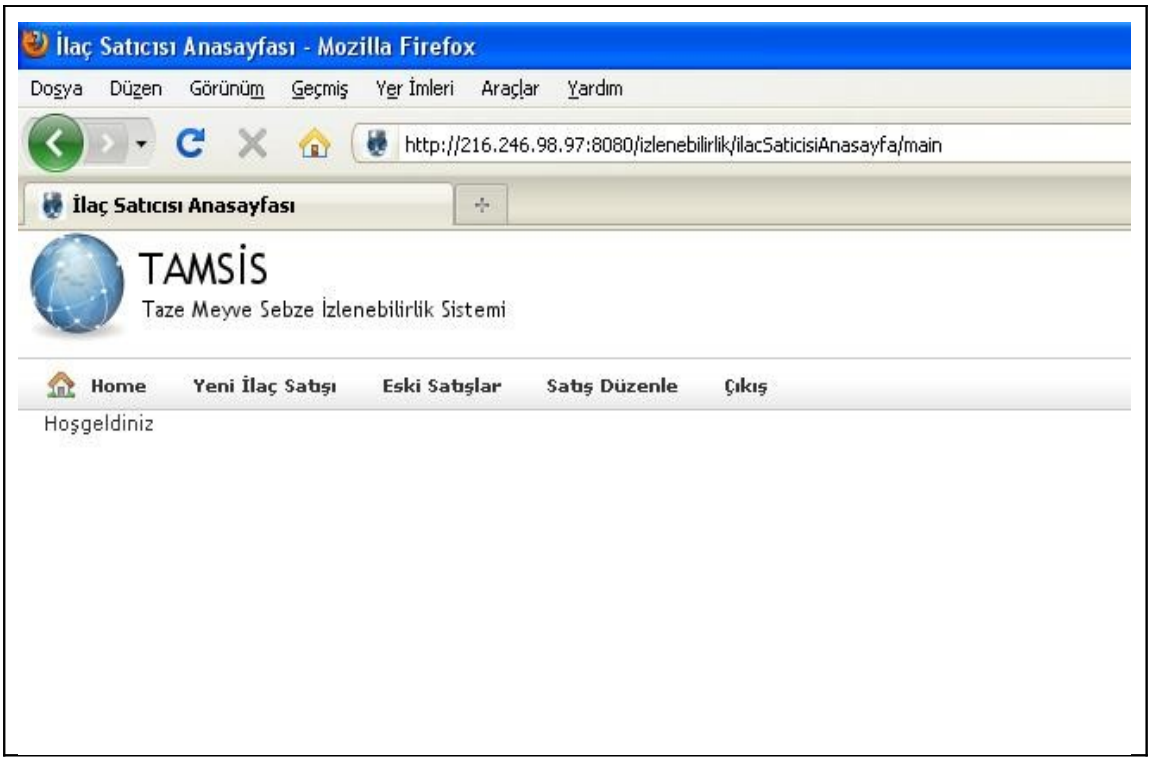
Şekil 3.24. Sistem Yöneticisi Ürün Listesi Ekranı

3.3.2. Zirai ilaç satıcısı

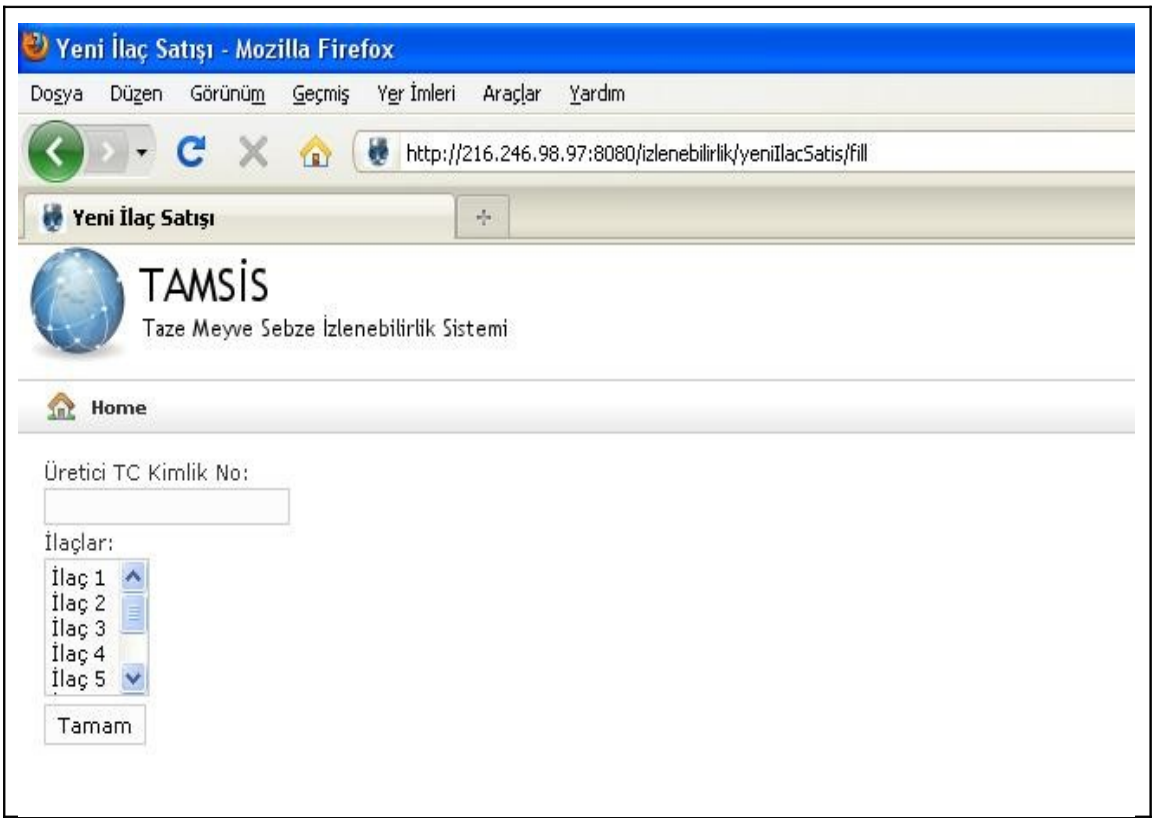
Zirai İlaç Satıcısı bir ilaç satışı gerçekleştirmek için TAMSİS Ana Sayfasındaki (Şekil 3.2.) Zirai İlaç Satıcısı bağlantısına tıklamalı ve açılan ekranda (Şekil 3.25.) daha önceden belirlenen kendisine ait kullanıcı adı ve parolasını girmelidir. Zirai İlaç Satıcısı, ana sayfadaki (Şekil 3.26.) "Yeni İlaç Satışı" sekmesini tıklamalı ve açılan ekranda (Şekil 3.27.) ilacı alacak olan çiftçinin vatandaşlık numarasını girerek, satılan ilaç ve/veya ilaçları listeden seçip satış işlemini gerçekleştirmelidir.



Şekil 3.25. Zirai İlaç Satıcısı Giriş Ekranı



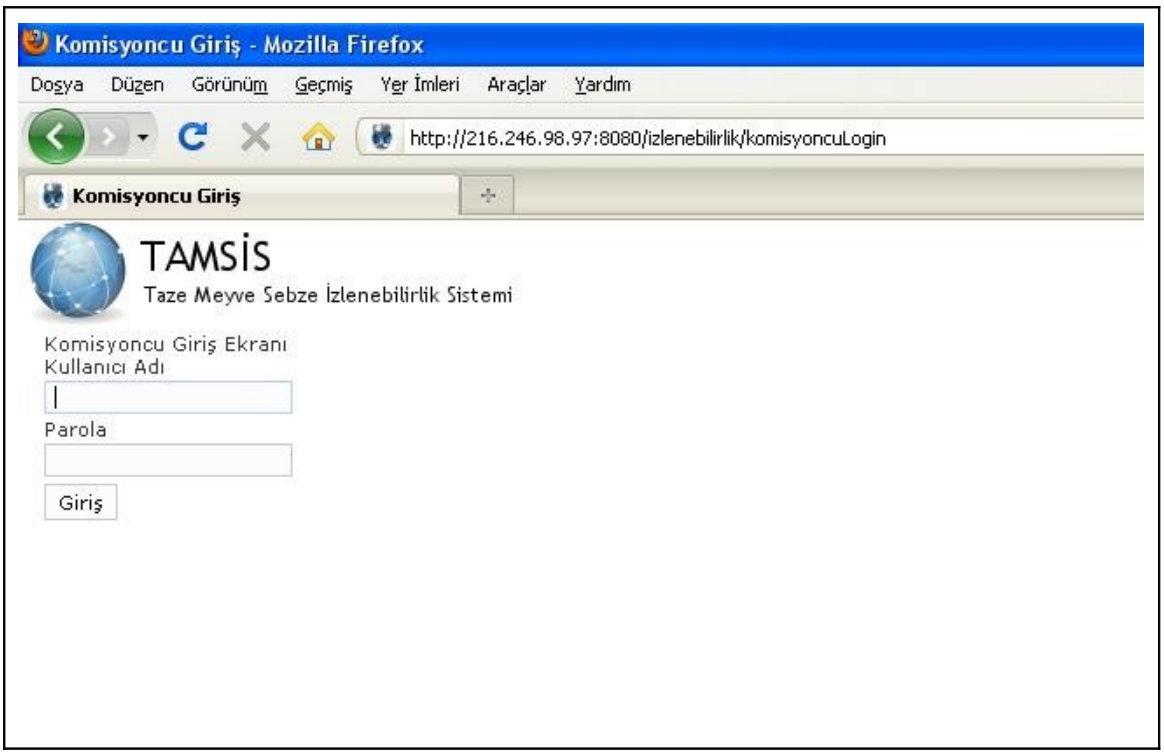
Şekil 3.26. Zirai İlaç Satıcısı Ana Ekranı



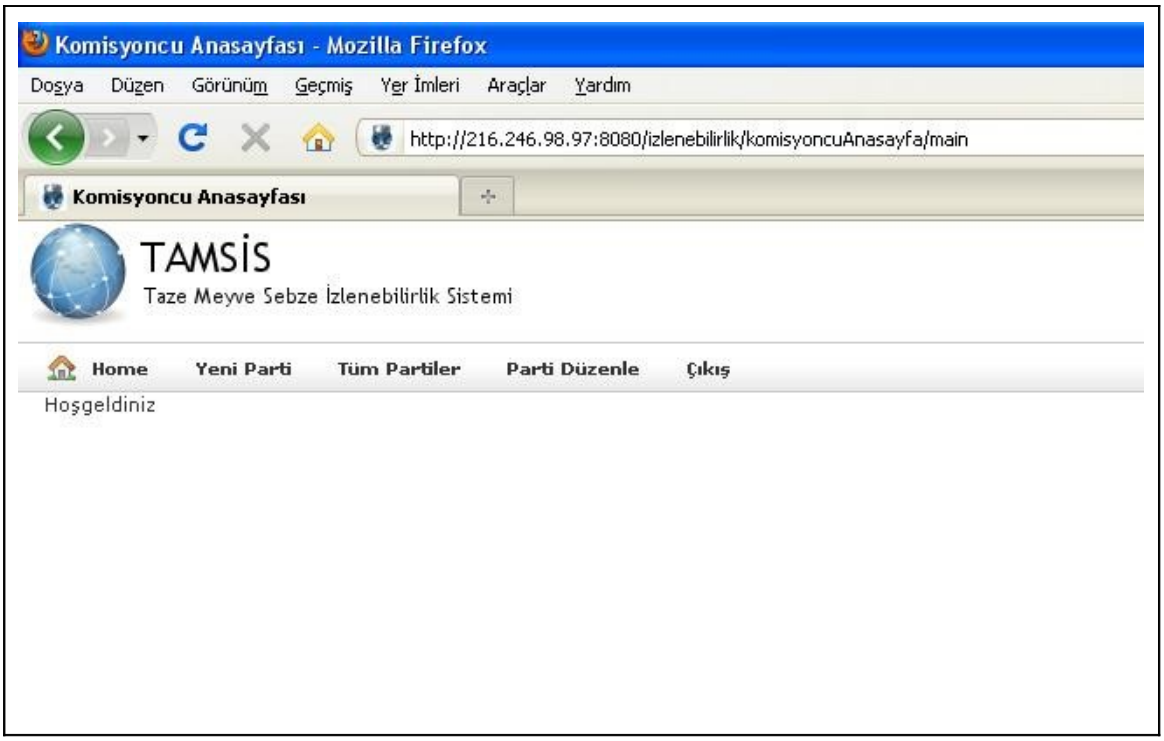
Şekil 3.27. Zirai İlaç Satıcısı Yeni İlaç Satış Ekranı

3.3.3. Komisyoncu / aracı

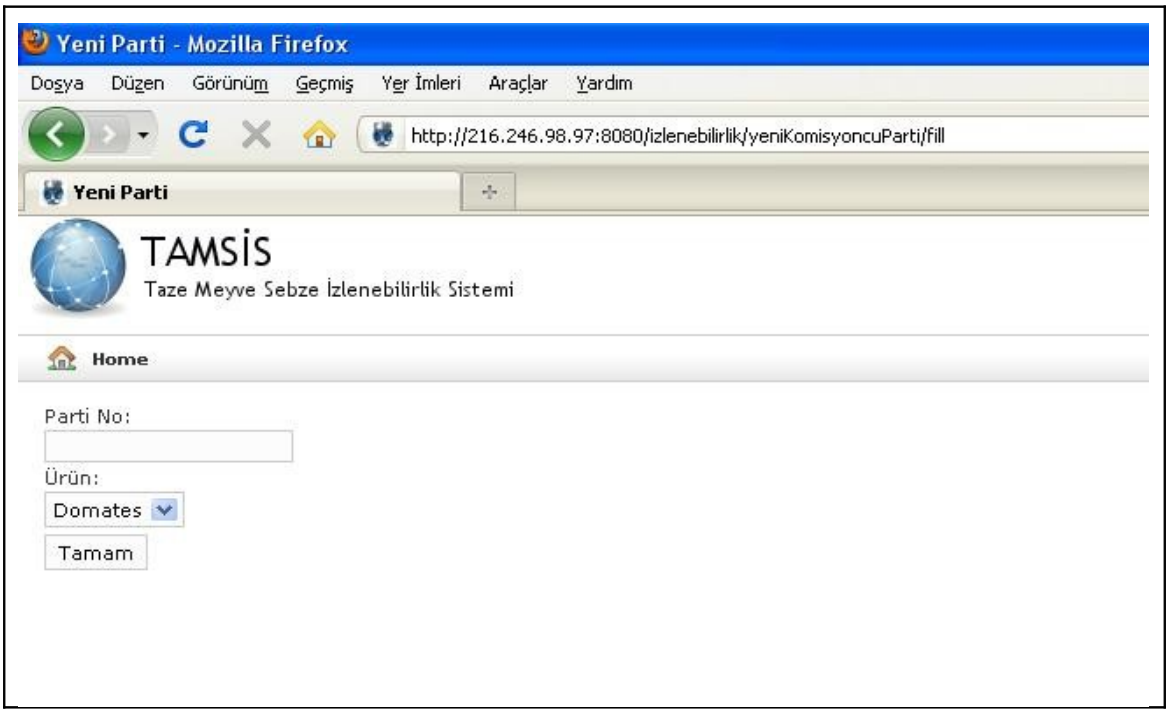
Komisyoncu aldığı ürünün girişini gerçekleştirmek için TAMSİS Ana Sayfasındaki (Şekil 3.2.) Komisyoncu/Aracı bağlantısına tıklamalı ve açılan ekranda (Şekil 3.28.) daha önceden belirlenen kendisine ait kullanıcı adı ve parolasını girmelidir. Açılan Komisyoncu Ana Sayfasındaki (Şekil 3.29.) "Yeni Parti" sekmesini tıklamalı ve Komisyoncu Ürün Parti Numarası Ekranına (Şekil 3.30.) ürün adı ve parti numarasını girerek işlemi tamamlamalıdır. İşlem tamamlandığında Komisyoncu Satın Alma Ekranı (Şekil 3.31.) açılacaktır.



Şekil 3.28. Komisyoncu Giriş Ekranı



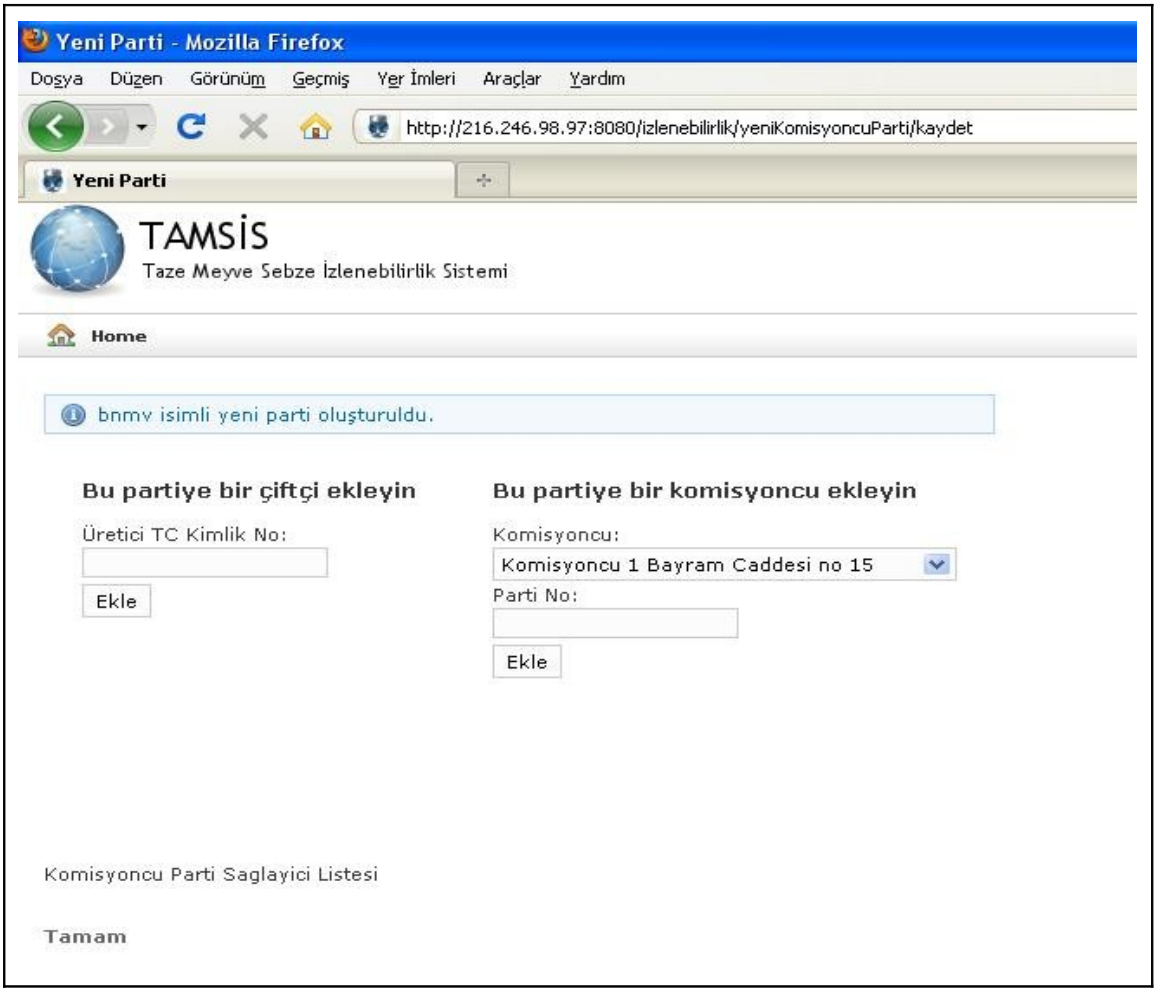
Şekil 3.29. Komisyoncu Ana Sayfası Ekranı



Şekil 3.30 Komisyoncu Ürün Parti Numarası Ekranı

Komisyoncu bu ekranda parti numarası verdiği ürünü kimlerden temin ettiğini girmelidir. Bu kapsamda, komisyoncuya ait ürün partisi sadece çiftçiden ve başka bir komisyoncudan alınarak oluşturulabileceği gibi her ikisinden alınarak da

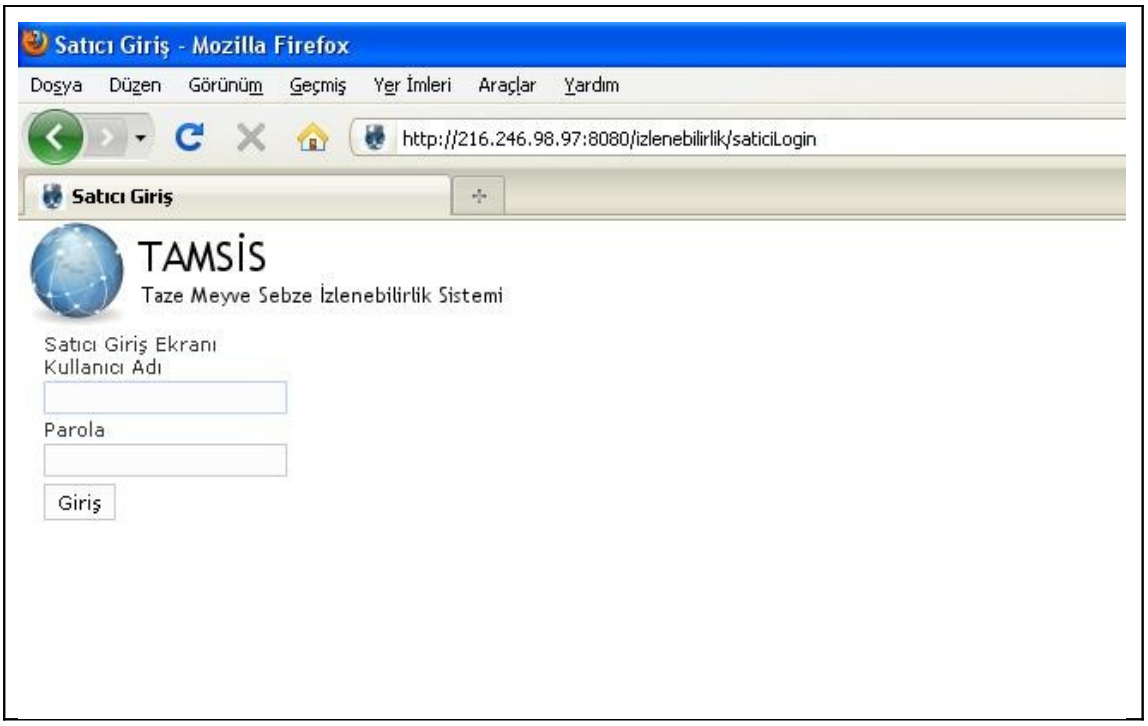
oluşturulabilir. Ürün çiftçiden alındığında çiftçinin vatandaşlık numarası ile çiftçinin üründe kullandığını beyan ettiği ilaçlar seçilerek ekle butonuna basılmak zorundadır. Eğer ürün başka bir komisyoncudan alınmış ise komisyoncu seçilmeli ve komisyoncunun ürününün parti numarası girilerek ekle butonuna basılmalıdır. Ürün parti sağlayıcıları listesi oluşturulduktan sonra tamam butonuna basılarak onaylanmalıdır.



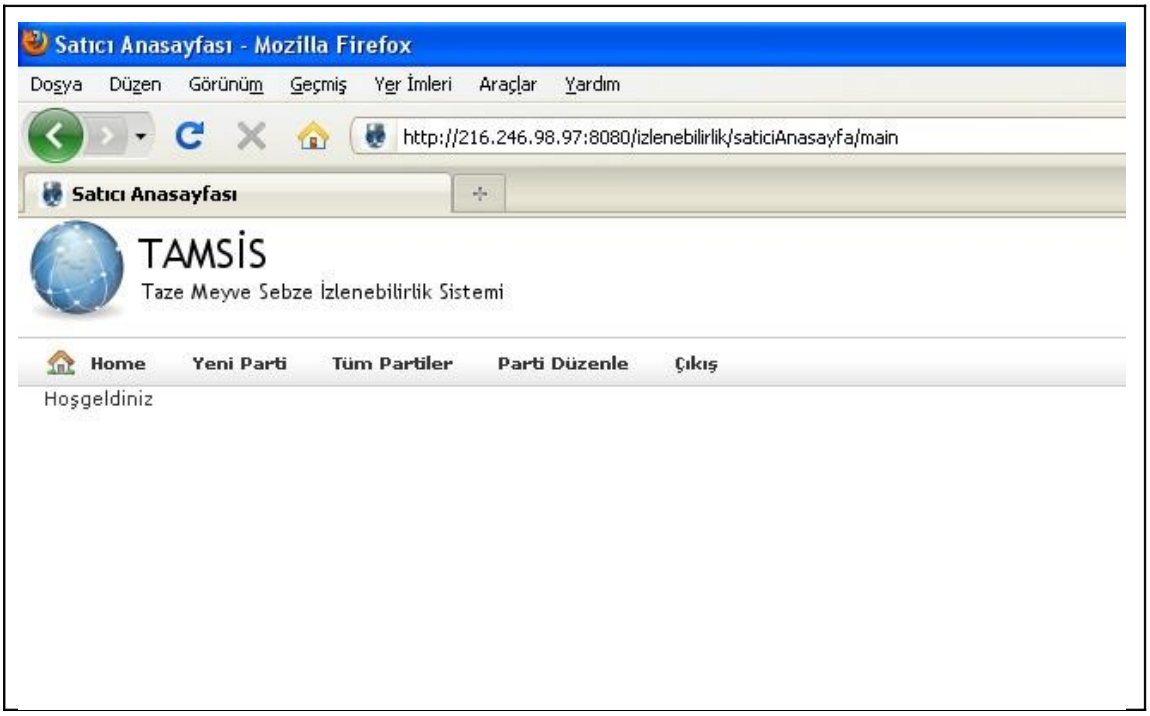
Şekil 3.31. Komisyoncu Satın Alma Ekranı

3.3.4. Satıcı / süpermarket

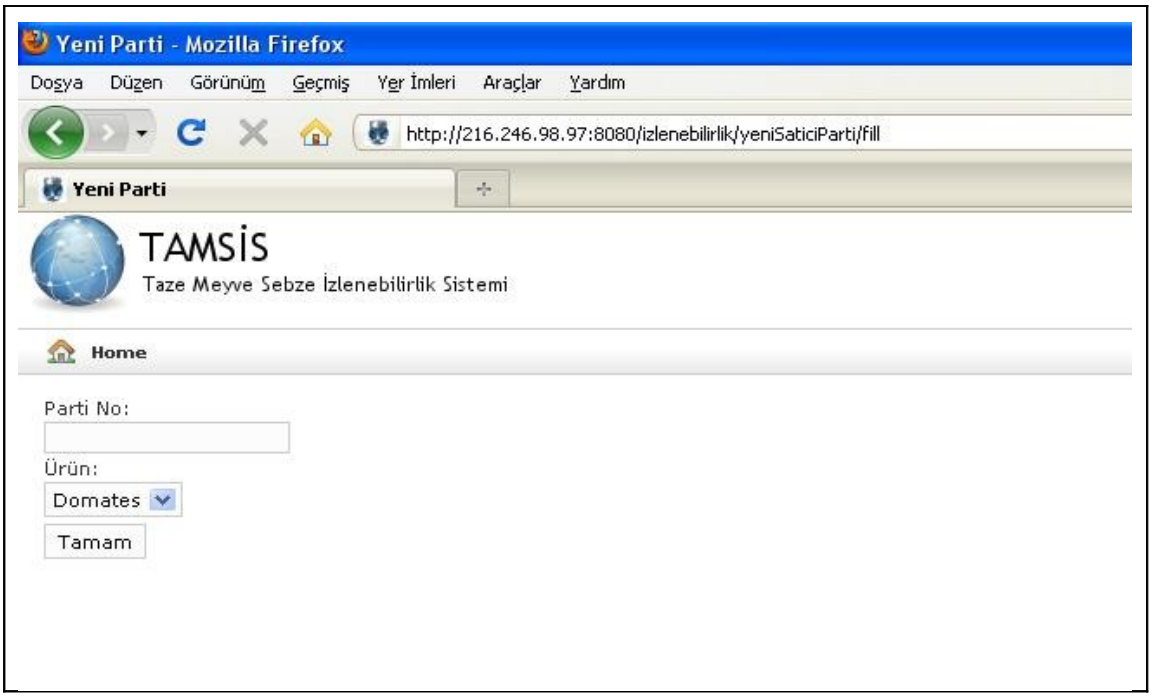
Satıcı aldığı ürünün girişini gerçekleştirmek için TAMSİS Ana Sayfasındaki (Şekil 3.2.) Satıcı bağlantısına tıklamalı ve açılan ekranda (Şekil 3.32.) daha önceden belirlenen kendisine ait kullanıcı adı ve parolasını girmelidir. Açılan Satıcı Ana Sayfasındaki (Şekil 3.33.) "Yeni Parti" sekmesini tıklamalı ve açılan ekranda (Şekil 3.34.) ürün adı ve kendi parti numarasını girerek işlemi tamamlamalıdır. İşlem tamamlandığında Satıcı Satın Alma Ekranı (Şekil 3.35.) açılacaktır.



Şekil 3.32. Satıcı Giriş Ekranı

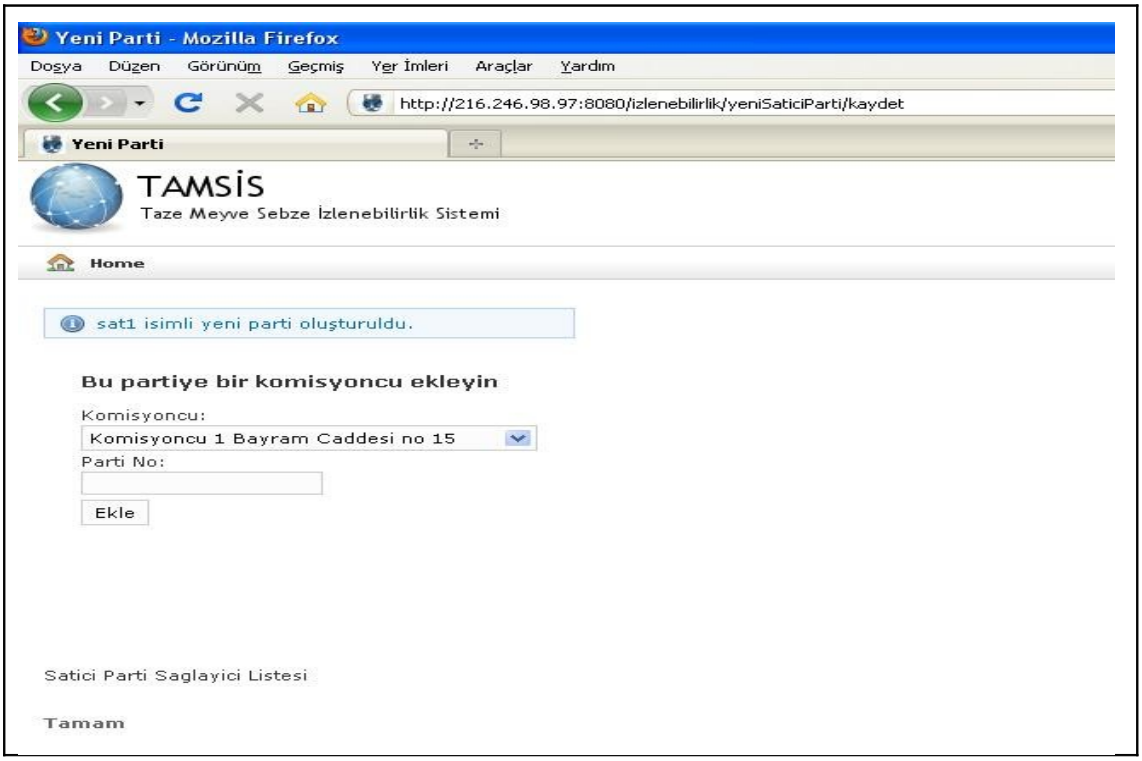


Şekil 3.33. Satıcı Ana Sayfası Ekranı



Şekil 3.34. Satıcı Ürün Parti Numarası Ekranı

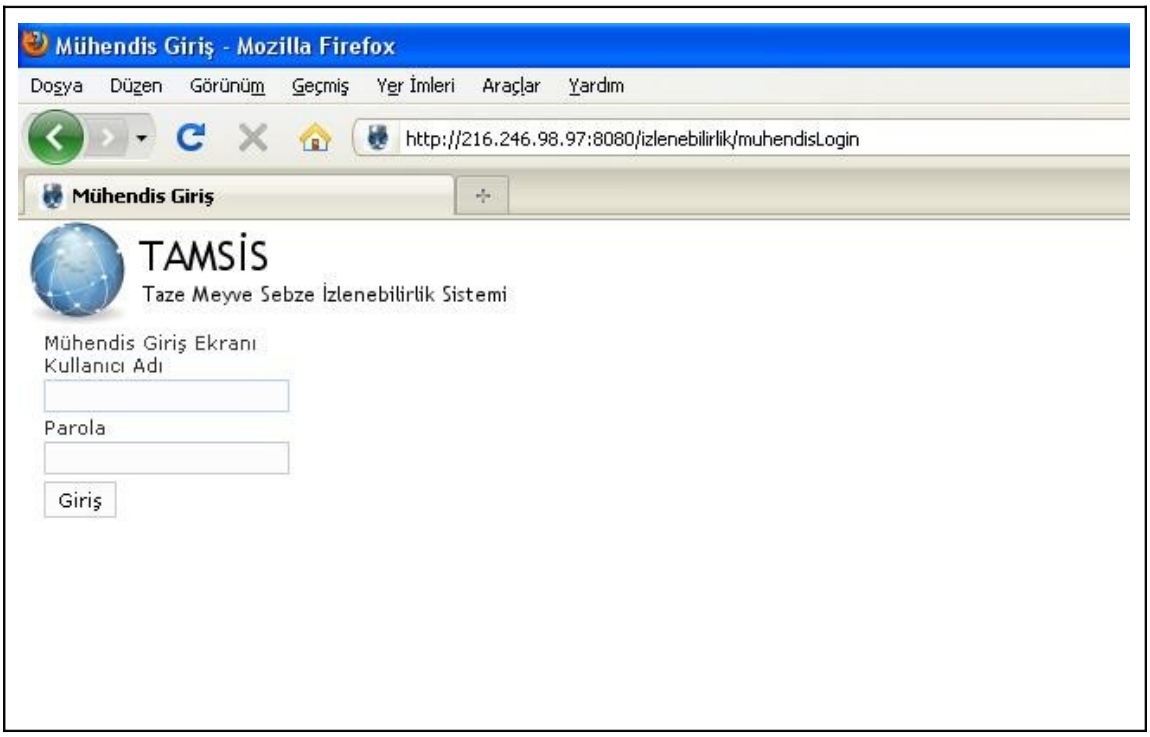
Satıcı bu ekranda (Şekil 3.35.) parti numarası verdiği ürünü hangi komisyoncu/aracılardan temin ettiğini girmelidir. Ürün parti sağlayıcıları listesi tamamlandıktan sonra tamam butonuna basılarak işlem onaylanmalıdır.



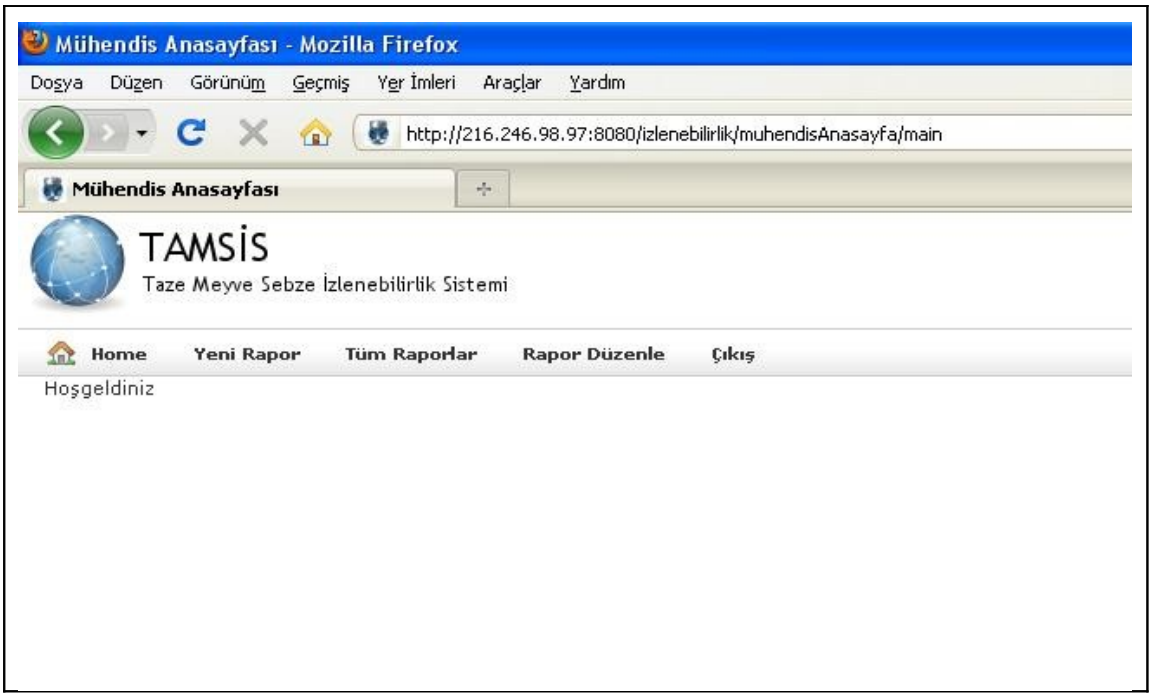
Şekil 3.35. Satıcı Satın Alma Ekranı

3.3.5. Mühendis

Mühendis kontrolünü yaptığı ürünün takibini gerçekleştirmek için öncelikle TAMSİS Ana Sayfasındaki (Şekil 3.2.) Mühendis bağlantısına tıklamalı ve açılan ekranda (Şekil 3.36.) kendisine ait kullanıcı adı ve parolasını girmelidir. Mühendisin laboratuarlardan gelen mevzuata uygun olmayan ürüne ait analiz sonuçlarını sisteme girebilmesi ve ürünü izlemesi için Mühendis Ana Sayfasındaki (Şekil 3.37.) "Yeni Rapor" sekmesini tıklaması gerekir.

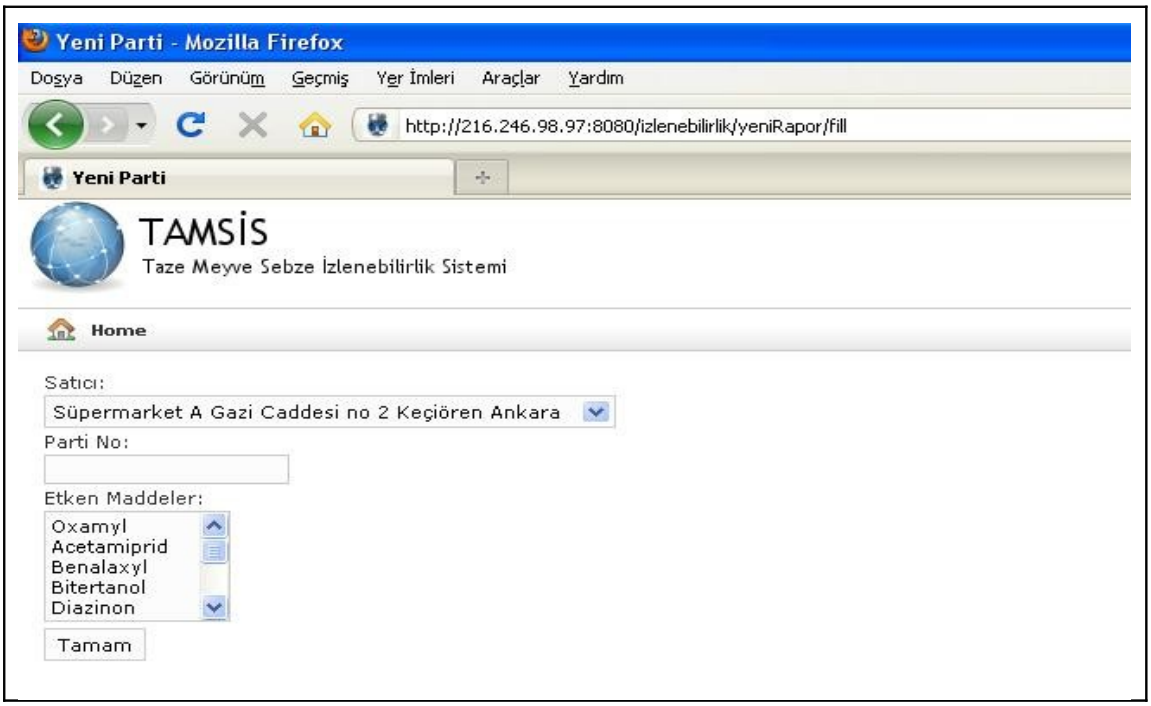


Şekil 3.36. Mühendis Giriş Ekranı



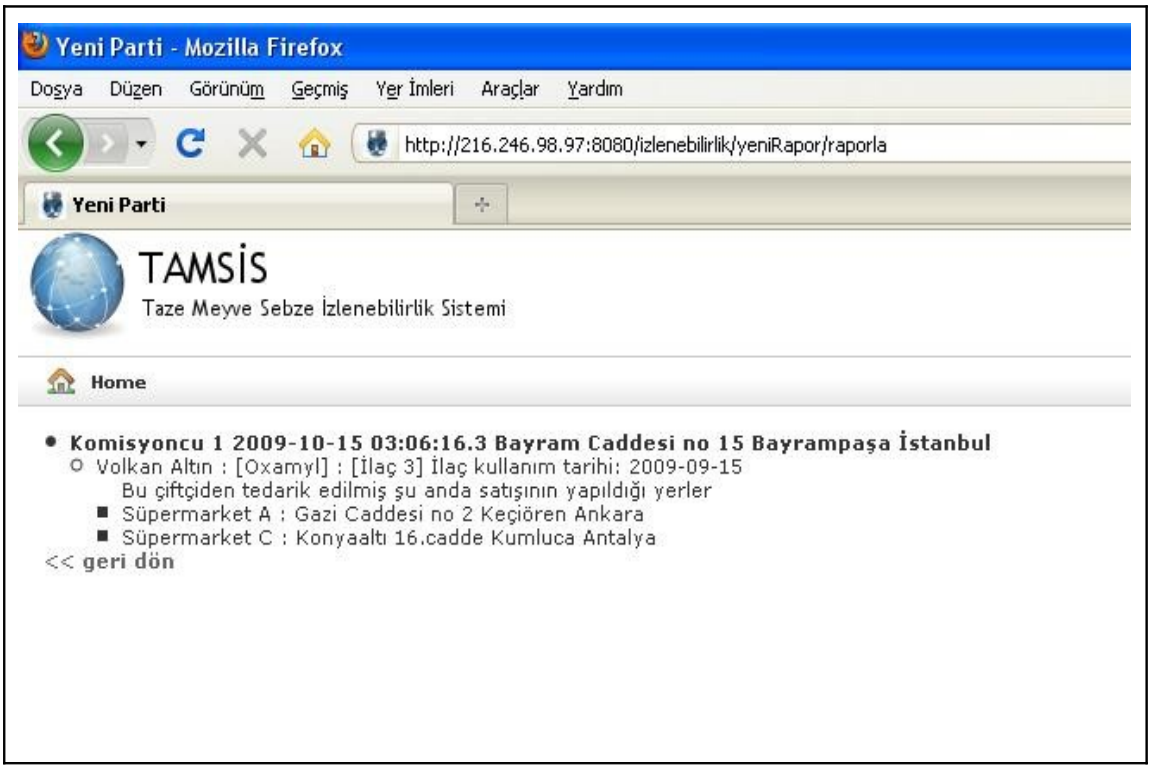
Şekil 3.37. Mühendis Ana Sayfası Ekranı

Yeni Rapor sekmesi tıklandıktan sonra açılan ekranda (Şekil 3.38.) numunenin alındığı satıcı firma, analiz sonucunda üründe mevzuata uygun olmadığı tespit edilen etken madde seçilerek ve parti numarası girilerek tamam butonuna basılmalıdır.



Şekil 3.38. Mühendis Veri Giriş Ekranı

Rapor Ekranında (Şekil 3.39.) satıcının mevzuata uygun olmayan ürününü kimlerden temin ettiği ve temin ettiği çiftçinin söz konusu ürününün başka hangi satıcılarda satışta olduğunun bilgisini göstermektedir.



Şekil 3.39. Rapor Ekranı

14. SONUÇLAR

Tam izlenebilirliğin kurulabilmesi için oluşturulan TAMSİS bilgisayar yazılımının işletimi ile ilgili olarak örnek bir durum seçilerek sunulmaktadır.

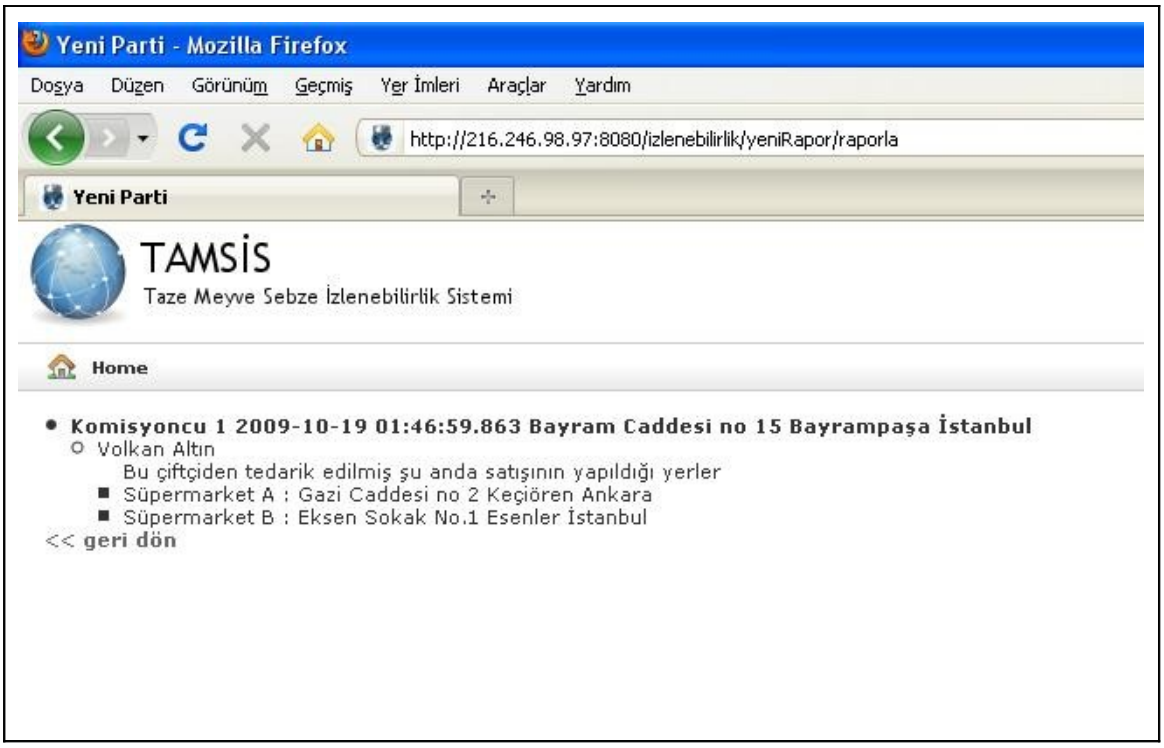
14.1. Örnek

Volkan Altın adlı çiftçinin ürettiği ve bir kısmını "Komisyoncu 1" adlı komisyoncu aracılığıyla "Süpermarket A" son satış noktasına; diğer kısmını ise "Komisyoncu 2" adlı komisyoncu aracılığıyla "Süpermarket B" son satış noktasına sattığı, mevzuata uygun olmayan domatesler örnek alınmıştır. "Süpermarket A" son satış noktasında bu domateslerden örnek alan mühendis oxamyl (İlaç 3) etken maddesini maksimum kalıntı limitlerinin üzerinde tespit etmiştir. Mühendisin bu durumda yanıtlaması gereken sorular;

1. "Süpermarket A"ya bu ürün hangi komisyoncu tarafından satılmıştır?
2. Komisyoncu bu ürünü hangi çiftçiden satın almıştır?
3. Aynı ürünün başka bir satış noktasında satışı bulunmakta mıdır?

Ürünün diğer satış noktalarından da hızlı bir biçimde geri çekilebilmesi için bu soruların yanıtlarına hemen ulaşılması büyük önem taşımaktadır. Şu anda mevcut olan sistemle (fatura takibi gibi) bu yanıtlara ulaşılması ya hiç mümkün olmamakta ya da sonuca ulaşıldığında (haftalarca sürebilir) ürün tüketilmiş olmaktadır.

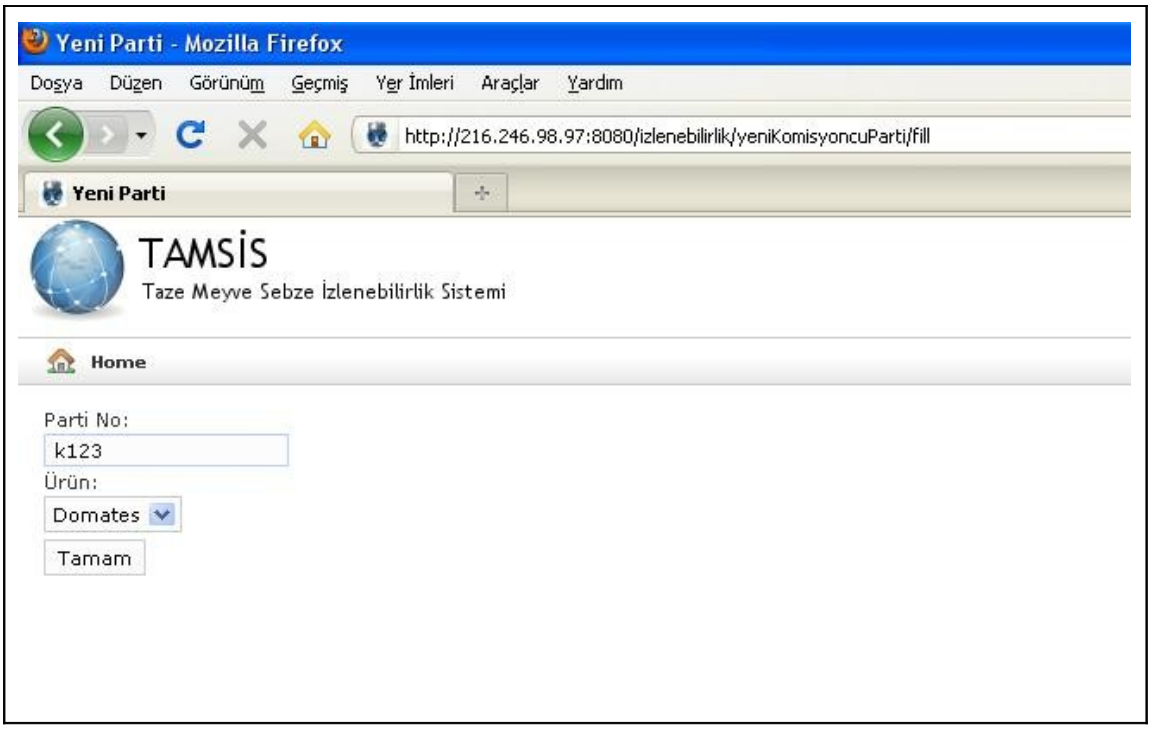
Mühendis yeni raporu sisteme yüklemek için kendine ait kullanıcı adı ve parolasını Bölüm 3.'de yer alan Şekil 3.36.'daki Mühendis Giriş Ekranına girerek Şekil 3.37.'deki Mühendis Ana Sayfasını açar. Bu ekranda Yeni Rapor sekmesine tıkladıktan sonra açılan sayfada (Şekil 3.38.) numunenin alındığı satıcı firma ile analiz sonucunda mevzuata uygun olmadığı tespit edilen etken maddeyi seçerek ve parti numarasını girip tamam butonuna basarak raporunu tamamlar. Sistem otomatik olarak Rapor Sonuç Sayfasını (Şekil 4.1.) ekrana getirir. Şekil 4.1.'de görüldüğü gibi mühendis bu raporla yukarıda sorulan soruların yanıtına ulaşır.



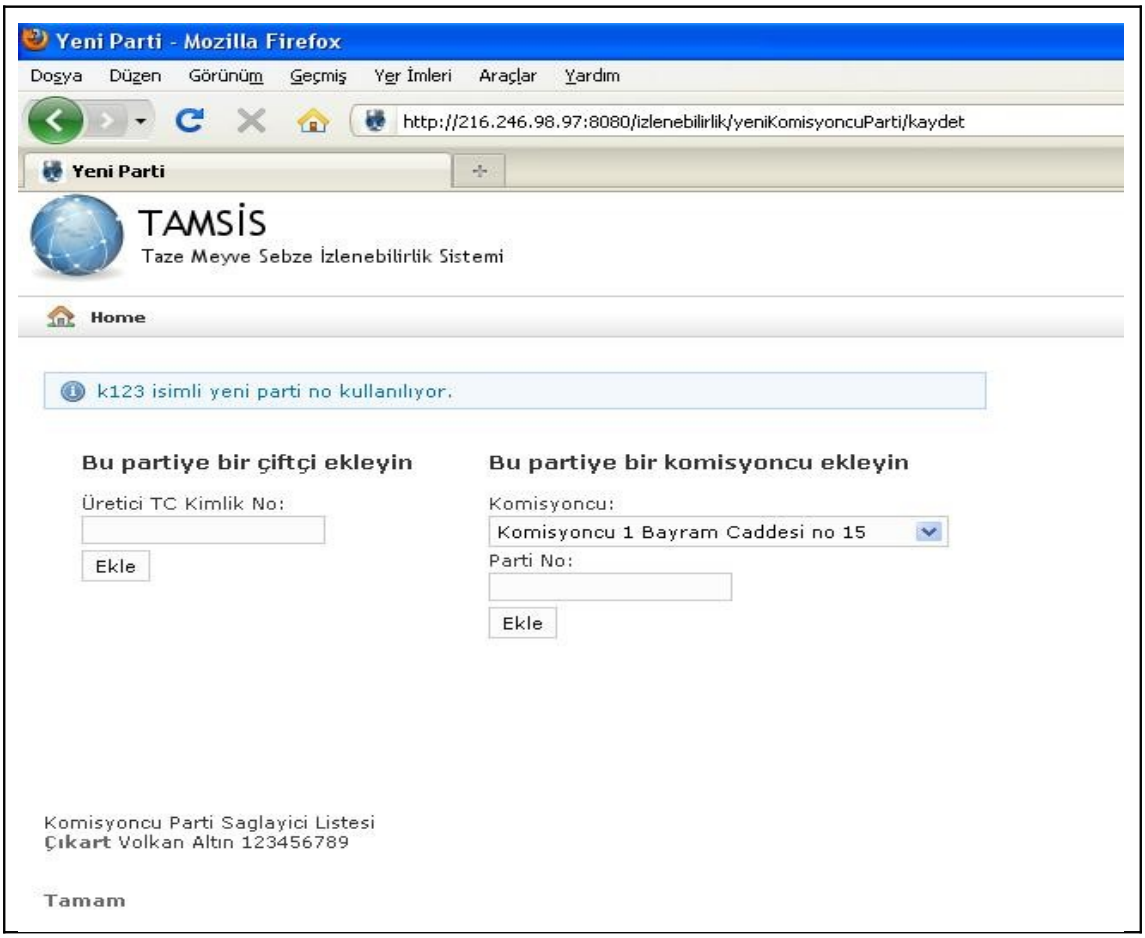
Şekil 4.1. Rapor Ekranı

Mühendisin bu rapora (Şekil 4.1.) ulaşabilmesi için sisteme giriş yapan paydaşların verilerini tam ve eksiksiz olarak girmeleri gerekmektedir. Buna göre ilk veri girişini komisyoncunun gerçekleştirmesi zorunludur.

“Komisyoncu 1” adlı komisyoncu Volkan Altın adlı çiftçiden aldığı domatesin girişini gerçekleştirmek için TAMSİS Ana Sayfasındaki (Şekil 3.2.) Komisyoncu/Aracı bağlantısına tıklamalı ve açılan ekranda (Şekil 3.28.) daha önceden belirlenen kendisine ait kullanıcı adı ve parolasını girmelidir. Açılan Komisyoncu Ana Sayfasındaki (Şekil 3.29.) “Yeni Parti” sekmesini tıkladıktan sonra ekrana gelen Komisyoncu Ürün Parti Numarası Ekranı sayfasında (Şekil 4.2.) ürün adı (domates) seçerek ve parti numarasını (k123) girerek işlemi tamamlamış olmalıdır. İşlem tamamlandığında Komisyoncu Satın Alma Ekranı (Şekil 4.3.) açılacaktır.



Şekil 4.2. Komisyoncu Ürün Parti Numarası Ekranı

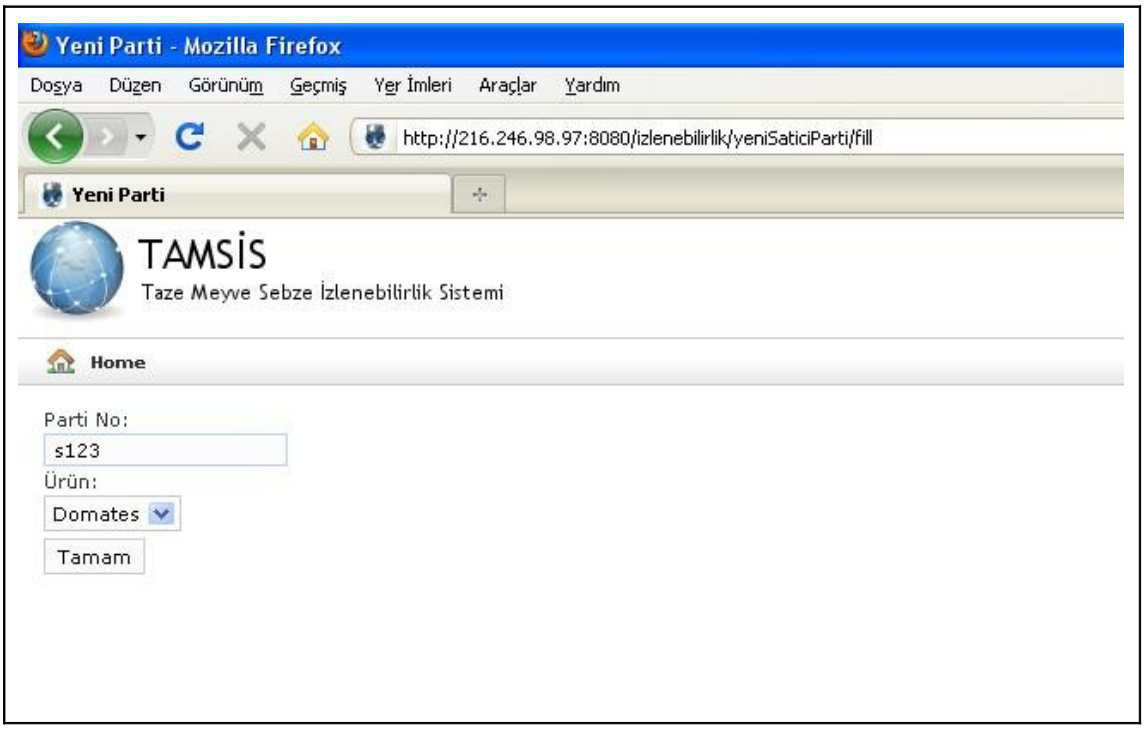


Şekil 4.3. Komisyoncu Satın Alma Ekranı

“Komisyoncu 1” adlı komisyoncu, ürünü satın aldığı “Volkan Altın” adlı çiftçinin vatandaşlık numarasını girerek çiftçiden domates ürününü satın aldığını onaylamıştır.

Komisyoncunun ürünü “Süpermarket A” ya sattığının bilgisi son satış noktası olan “Süpermarket A”nın sisteme yükleyeceği veri ile mümkündür.

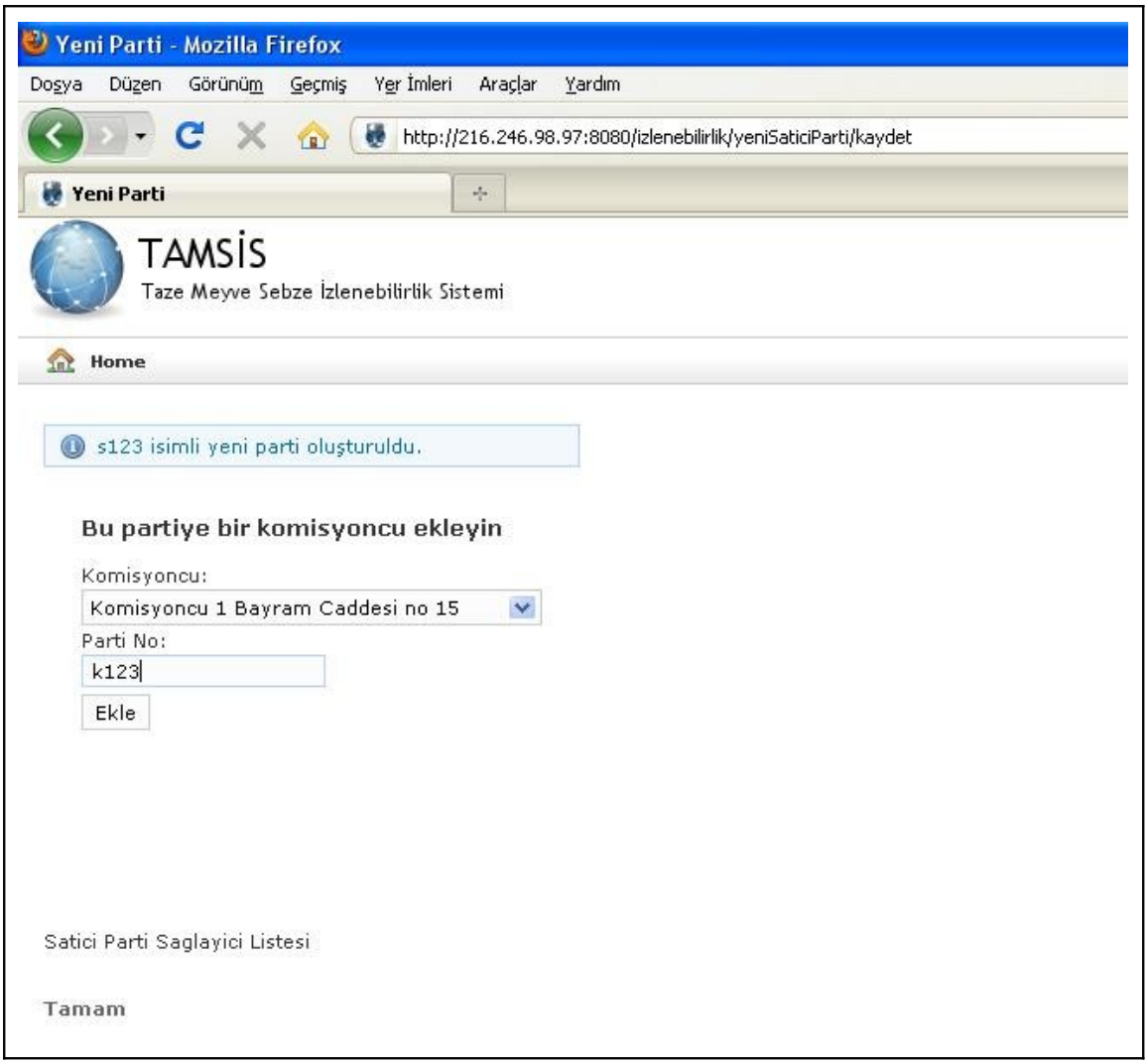
“Süpermarket A” adlı satıcı aldığı ürünün girişini gerçekleştirmek için TAMSİS Ana Sayfasındaki (Şekil 3.2.) Satıcı/Süpermarket bağlantısına tıklamalı ve açılan ekranda (Şekil 3.32.) daha önceden belirlenen kendisine ait kullanıcı adını ve parolasını girmelidir. Açılan Satıcı Ana Sayfasındaki (Şekil 3.33.) “Yeni Parti” sekmesini tıkladıktan sonra ekrana gelen Satıcı Ürün Parti Numarası Ekranı sayfasında (Şekil, 4.4.) ürün adı (domates) ve parti numarasını (s123) girerek işlemi tamamlamış olmalıdır. İşlem tamamlandığında Satıcı Satın Alma Ekranı (Şekil 4.5.) açılacaktır.



Şekil 4.4. Satıcı Ürün Parti Numarası Ekranı

“Süpermarket A” son satış noktası, bir önceki ekranda kendi parti numarası ile tanımladığı ürünün bilgilerini Şekil 4.5.’deki Satıcı Satın Alma Ekranında vermek zorundadır. Bu bilgiler ürünün hangi komisyoncudan (Komisyoncu 1) ve hangi parti numarasıyla (k123) satın alındığı bilgileridir. Bu işlem tamamlandığında

“Süpermarket A” nın komisyoncudan domates ürününü satın aldığı onaylanmış olacaktır.



Şekil 4.5. Satıcı Satın Alma Ekranı

Şekil 4.1.’de yer alan Rapor Ekranı’nda Volkan Altın adlı çiftçiye ait domateslerin “Süpermarket A” nın yanı sıra “Süpermarket B” son satış noktasında da satışa sunulmuş olduğu bilgisi yer almaktadır. Sistem bu bilgiye “Süpermarket B” son satış noktasının “Süpermarket A” ile aynı tarihlerde yüklediği veriyle ulaşmaktadır. “Süpermarket B” verilerine göre ürün “Komisyoncu 2” adlı komisyoncudan satın alınmıştır. “Süpermarket B”nin domatese verdiği parti numarası ile “Komisyoncu 2” tarafından Volkan Altın adlı çiftçiden satın alınan domatese verilen parti numarası sistem tarafından eşlenerek çiftçinin bilgilerine ulaşılmaktadır.

15. SONUÇLAR VE TARTIŞMA

Mevcut yasal düzenlemeler, gıda zincirinde bir adım ileri ve geriye izlenebilirliği zorunlu kılmakta ve böylelikle yasalara göre bir işletme, satın aldığı ve sattığı mal ve hizmetlere ilişkin kayıtlardan sorumlu tutulmaktadır. Tam izlenebilirlik ise ideal bir sistem olup gıda zincirinin her kademesinde tüm ilgili taraflarca yapılan her türlü işlem ve uygulamanın izlenebilirliğini amaçlayan bir sistemdir. Tüm gıda zincirinde yatay ve dikey izlenebilirlik ancak tümleşik bilgi sistemleri ve uygun teknolojilerin kullanılmasıyla gerçekleştirilebilecek kadar kapsamlıdır.

Bu çalışmada hazırlanan TAMSİS izlenebilirlik bilgisayar programı,

1. İzlenebilirlik için gerekli verilerin etkin, düşük maliyetli ve modern yöntem ve tekniklerle bilgi toplanmasını ve kayıt altına alınmasını (mümkün olduğunca elektronik olanaklarla otomatik olarak) sağlayacak bir sistemdir.

2. Verilerin birbirinden bağımsız uygulama ve çözümler yerine birbiriyle tümleşik ve mümkün olduğunca gerçek zamanlı işlenmesi ve analizini mümkün kılacaktır.

Bu programla gerçekleştirilecek etkin izlenebilirlik sistemi herhangi bir gıda güvenliği sorununda hızlı bilgi toplamayı dolayısıyla sorunun kaynağı ve nedenini mümkün olduğunca çabuk biçimde saptamayı gerçekleştirebilecek ve tedarik zincirinde gıda güvenliğinin sürdürülebilirliğini sağlayacaktır. Geleneksel kağıt tabanlı izleme sistemleri yerini elektronik tabanlı TAMSİS gibi sistemlere bırakmak zorundadır. İyi ve kabul edilebilir bir izlenebilirlik sisteminin elektronik olsa da sadece mali belgelere dayanan bir izlenebilirlik yerine yeni gelişmelere açık, belli standartlarla istenen belge ve bilgileri sağlayan ve genişleyebilir nitelikte olması gereklidir.

TAMSİS programı:

- Merkezi yönetim ve denetime olanak sağlamakta,
- Üretim sistemini dikey olarak kapsamakta,
- Uluslararası izlenebilirlik standartları ile uyumlu,

- Etkin, esnek ve genişleyebilir nitelikte,
- Düşük maliyetle uygulanabilir bir programdır.

KAYNAKLAR

Akbay, C. O., 2005, Türkiye'de Yaş Meyve ve Sebze Ürünleri Üretim ve Pazarlaması, Kahramanmaraş.

Cebeci, Z., 2006, Gıda İzlenebilirliğinde Bilgi Teknolojileri, Ulusal Tarım Kurultayı, Çukurova Üniversitesi, Bildiriler 189-195.

Codex Alimentarius, 2008, Resmi Internet Sitesi, www.codexalimentarius.net

Furness, A. and Osman K. A., 2006, Developing traceability systems across the food supply chain. Improving Traceability in Food Processing and Distribution. Smith, I. and Furness, A. (eds), Woodhead, Cambridge. pp. 3 -24.

GS1, 2008, Global Standarts 1, Resmi İnternet Sitesi, www.gs1.org/about/overwiev.html

Kaplan, E. D. and Hegarty, C., 2005, Understanding GPS: Principles and Applications, Second Edition Artech House Inc.685 Canton Street, Norwood, MA, , 02062 , ABD.

Klafft, M., Huen, J., Kuhn, C., Huen, E. and Wößner, S., 2006, Including process information in traceability. Improving Traceability in Food Processing and Distribution. Smith, I. and Furness, A. (eds), Woodhead, Cambridge. pp. 107-121

Lenstra, J. A., 2006. DNA markers for animal and plant traceability. Improving Traceability in Food Processing and Distribution. Smith, I. and Furness, A. (eds), Woodhead, Cambridge. pp. 147-159.

Opara, L. U., 2003, Traceability in agriculture and food supply chain: a review of basic concepts, technological implications, and future prospects, Food, Agriculture & Environment, 1,1,101-106

Özkan, B., 2006, Türkiye'de Yaş Meyve ve Sebze Tedarik Zinciri, Akdeniz Üniversitesi Ziraat Fakültesi Tarım Ekonomisi Bölümü.

TKİB, 2004, Tarım ve Köyişleri Bakanlığı, II. Tarım Şurası II. Komisyon Çalışma Belgesi, Ankara, s4.

TKİB, 2008, Tarım ve Köyişleri Bakanlığı, Resmi Internet Sitesi, www.tarim.gov.tr

TOBB, 2006, Türkiye Odalar ve Borsalar Birliği, Resmi İnternet Sitesi, Türkiye GS1 Sistemi Tanımlama, Numaralandırma ve Barkod Standartları Uygulama Kılavuzu, www.tobb.org.tr.

TOBB, 2007, Türkiye Odalar ve Borsalar Birliği, Resmi İnternet Sitesi, GS1 Sistem Tanıtım Kitapçığı, www.tobb.org.tr

Verdenius, F., 2006, Using traceability systems to optimise business performance. Improving Traceability in Food Processing and Distribution. Smith, I. and Furness, A. (eds), Woodhead, Cambridge. pp. 26-51.

Yurdakul, O., Tarım Ürünleri Pazarlaması, Çukurova Üniversitesi, Ziraat Fakültesi Ders Kitapları Yayın No: A-39., 2002.

EKLER

```
class EtkenMaddeController {
    def index = { redirect(action:list,params:params) }
    // the delete, save and update actions only accept POST requests
    static allowedMethods = [delete:'POST', save:'POST', update:'POST']
    def list = {
      params.max = Math.min( params.max ? params.max.toInteger() :
10,100)
        [ etkenMaddeInstanceList: EtkenMadde.list( params ),
etkenMaddeInstanceTotal: EtkenMadde.count() ]}
    def show = {
        def etkenMaddeInstance = EtkenMadde.get( params.id )
        if(!etkenMaddeInstance) {
            flash.message = "EtkenMadde not found with id ${params.id}"
            redirect(action:list)}
        else { return [ etkenMaddeInstance : etkenMaddeInstance ] }}
    def delete = {
        def etkenMaddeInstance = EtkenMadde.get( params.id )
        if(etkenMaddeInstance) {
            try {
                etkenMaddeInstance.delete(flush:true)
                flash.message = "EtkenMadde ${params.id} deleted"
                redirect(action:list)}
         catch(org.springframework.dao.DataIntegrityViolationException e)
{
             flash.message = "EtkenMadde ${params.id} could not be
deleted"
                redirect(action:show,id:params.id)}}
        else {
            flash.message = "EtkenMadde not found with id ${params.id}"
            redirect(action:list)}}
    def edit = {
        def etkenMaddeInstance = EtkenMadde.get( params.id )
        if(!etkenMaddeInstance) {
            flash.message = "EtkenMadde not found with id ${params.id}"
            redirect(action:list)}
        else {
            return [ etkenMaddeInstance : etkenMaddeInstance ]}}
    def update = {
        def etkenMaddeInstance = EtkenMadde.get( params.id )
        if(etkenMaddeInstance) {
            if(params.version) {
                def version = params.version.toLong()
                if(etkenMaddeInstance.version > version) {
                    etkenMaddeInstance.errors.rejectValue("version",
"etkenMadde.optimistic.locking.failure", "Another user has updated this
EtkenMadde while you were editing.")
    render(view:'edit',model:[etkenMaddeInstance:etkenMaddeInstance])
                    return}}
            etkenMaddeInstance.properties = params
            if(!etkenMaddeInstance.hasErrors() &&
etkenMaddeInstance.save()) {
                flash.message = "EtkenMadde ${params.id} updated"
                redirect(action:show,id:etkenMaddeInstance.id)}
            else {
render(view:'edit',model:[etkenMaddeInstance:etkenMaddeInstance])}}
        else {
            flash.message = "EtkenMadde not found with id ${params.id}"
            redirect(action:list)}}
```

```
    def create = {
        def etkenMaddeInstance = new EtkenMadde()
        etkenMaddeInstance.properties = params
        return ['etkenMaddeInstance':etkenMaddeInstance]}

    def save = {
        def etkenMaddeInstance = new EtkenMadde(params)
        if(!etkenMaddeInstance.hasErrors() && etkenMaddeInstance.save())
{
            flash.message = "EtkenMadde ${etkenMaddeInstance.id} created"
            redirect(action:show,id:etkenMaddeInstance.id)}
        else {
    render(view:'create',model:
[etkenMaddeInstance:etkenMaddeInstance])}}}


class IlacSaticisiAnasayfaController {
  def beforeInterceptor = [action: this.&checkUser]
  def checkUser() {
    if (!session.ilacSaticisi) {
      redirect(controller: 'ilacSaticisiLogin', action: 'index')
      return false}}
  def index = {
    redirect(action: 'main')}
  def main = {}
  def logout = {
    session.ilacSaticisi = null
    redirect(controller: '.')}}


class EskiIlacSatislariController {
  def index = { redirect(action: fill, params: params) }
  def fill = {
    [ilacCiftciSatislari: IlacCiftciSatis.findAll()]}}


class YoneticiLoginController {
  def index = {}
  def login = {
    if (params.containsKey("kullanici") && params.containsKey("parola"))
{
      def yonetici = Yonetici.findWhere(kullanici: params["kullanici"],
parola: params["parola"]);
      if (yonetici) {
        session.yonetici = yonetici
        redirect(action: redirect)
      } else {
        redirect(action: error)}
    } else {
      redirect(action: error)}}
  def error = {}
  def redirect = {}}


class IlacSaticisiLoginController {
  def index = {}
  def login = {
    if (params.containsKey("kullanici") && params.containsKey("parola"))
{
```

```
      def ilacSaticisi = IlacSaticisi.findWhere(kullanici:
params["kullanici"], parola: params["parola"]);
      if (ilacSaticisi) {
        session.ilacSaticisi = ilacSaticisi
        redirect(action: redirect)
      } else {
        redirect(action: error)}
    } else {
      redirect(action: error)}}
  def error = {}
  def redirect = {}}


class YoneticiAnasayfaController {
  def beforeInterceptor = [action: this.&checkUser]
  def checkUser() {
    if (!session.yonetici) {
      redirect(controller: 'yoneticiLogin', action: 'index')
      return false}}
  def index = {
    redirect(action: 'main')}
  def main = {}
  def logout = {
    session.yonetici = null
    redirect(controller: '.')}}


class KomisyoncuAnasayfaController {
  def beforeInterceptor = [action: this.&checkUser]
  def checkUser() {
    if (!session.komisyoncu) {
      redirect(controller: 'komisyoncuLogin', action: 'index')
      return false}}
  def index = {
    redirect(action: 'main')}
  def main = {}
  def logout = {
    session.komisyoncu = null
    redirect(controller: '.')}}


class MuhendisLoginController {
  def index = {}
  def login = {
    if (params.containsKey("kullanici") && params.containsKey("parola"))
{
      def muhendis = Muhendis.findWhere(kullanici: params["kullanici"],
parola: params["parola"]);
      if (muhendis) {
        session.muhendis = muhendis
        redirect(action: redirect)
      } else {
        redirect(action: error)}
    } else {
      redirect(action: error)}}
  def error = {}
  def redirect = {}}


class SaticiAnasayfaController {
```

```
  def beforeInterceptor = [action: this.&checkUser]
  def checkUser() {
    if (!session.satici) {
      redirect(controller: 'saticiLogin', action: 'index')
      return false}}
  def index = {
    redirect(action: 'main')}
  def main = {}
  def logout = {
    session.satici = null
    redirect(controller: '.')}}


class MuhendisAnasayfaController {
 def beforeInterceptor = [action: this.&checkUser]
  def checkUser() {
    if (!session.muhendis) {
      redirect(controller: 'muhendisLogin', action: 'index')
      return false}}
  def index = {
    redirect(action: 'main')}
  def main = {}
  def logout = {
    session.muhendis = null
    redirect(controller: '.')}}


class SaticiLoginController {
  def index = {}
  def login = {
    if (params.containsKey("kullanici") && params.containsKey("parola"))
{
      def satici = Satici.findWhere(kullanici: params["kullanici"],
parola: params["parola"]);
      if (satici) {
        session.satici = satici
        redirect(action: redirect)
      } else {
        redirect(action: error)}
    } else {
      redirect(action: error)}}
  def error = {}
  def redirect = {}}


class KomisyoncuLoginController {
  def index = {}
  def login = {
    if (params.containsKey("kullanici") && params.containsKey("parola"))
{
      def komisyoncu = Komisyoncu.findWhere(kullanici:
params.get("kullanici"), parola: params.get("parola"));
      if (komisyoncu) {
        session.komisyoncu = komisyoncu
        redirect(action: redirect)
      } else {
        redirect(action: error)}
    } else {
      redirect(action: error)}}
  def error = {}
```

```
  def redirect = {}}


class UrunController {
  def beforeInterceptor = [action: this.&checkUser]
  def checkUser() {
    if (!session.yonetici) {
      redirect(controller: 'yoneticiLogin', action: 'index')
      return false}}
  def index = { redirect(action: list, params: params) }
  // the delete, save and update actions only accept POST requests
  static allowedMethods = [delete: 'POST', save: 'POST', update: 'POST']
  def list = {
    params.max = Math.min(params.max ? params.max.toInteger() : 10, 100)
    [urunInstanceList: Urun.list(params), urunInstanceTotal:
Urun.count()]}
  def show = {
    def urunInstance = Urun.get(params.id)
    if (!urunInstance) {
      flash.message = "Urun not found with id ${params.id}"
      redirect(action: list)}
    else { return [urunInstance: urunInstance] }}
  def delete = {
    def urunInstance = Urun.get(params.id)
    if (urunInstance) {
      try {
        urunInstance.delete(flush: true)
        flash.message = "Urun ${params.id} deleted"
        redirect(action: list)}
      catch (org.springframework.dao.DataIntegrityViolationException e) {
        flash.message = "Urun ${params.id} could not be deleted"
        redirect(action: show, id: params.id)}}
    else {
      flash.message = "Urun not found with id ${params.id}"
      redirect(action: list)}}
  def edit = {
    def urunInstance = Urun.get(params.id)
    if (!urunInstance) {
      flash.message = "Urun not found with id ${params.id}"
      redirect(action: list)}
    else {
      return [urunInstance: urunInstance]}}
  def update = {
    def urunInstance = Urun.get(params.id)
    if (urunInstance) {
      if (params.version) {
        def version = params.version.toLong()
        if (urunInstance.version > version) {
          urunInstance.errors.rejectValue("version",
"urun.optimistic.locking.failure", "Another user has updated this Urun
while you were editing.")
          render(view: 'edit', model: [urunInstance: urunInstance])
          return}   }
      urunInstance.properties = params
      if (!urunInstance.hasErrors() && urunInstance.save()) {
        flash.message = "Urun ${params.id} updated"
        redirect(action: show, id: urunInstance.id)}
      else {
        render(view: 'edit', model: [urunInstance: urunInstance])}}
    else {
      flash.message = "Urun not found with id ${params.id}"
```

```
      redirect(action: list)    }  }
  def create = {
    def urunInstance = new Urun()
    urunInstance.properties = params
    return ['urunInstance': urunInstance]  }
  def save = {
    def urunInstance = new Urun(params)
    if (!urunInstance.hasErrors() && urunInstance.save()) {
      flash.message = "Urun ${urunInstance.id} created"
      redirect(action: show, id: urunInstance.id)    }
    else {
      render(view: 'create', model: [urunInstance: urunInstance])}}}


class SaticiController {
  def beforeInterceptor = [action: this.&checkUser]
  def checkUser() {
    if (!session.yonetici) {
      redirect(controller: 'yoneticiLogin', action: 'index')
      return false}}
  def index = { redirect(action: list, params: params) }
  // the delete, save and update actions only accept POST requests
  static allowedMethods = [delete: 'POST', save: 'POST', update: 'POST']
  def list = {
    params.max = Math.min(params.max ? params.max.toInteger() : 10, 100)
    [saticiInstanceList: Satici.list(params), saticiInstanceTotal:
Satici.count()]}
  def show = {
    def saticiInstance = Satici.get(params.id)
    if (!saticiInstance) {
      flash.message = "Satici not found with id ${params.id}"
      redirect(action: list)}
    else { return [saticiInstance: saticiInstance] }}
  def delete = {
    def saticiInstance = Satici.get(params.id)
    if (saticiInstance) {
      try {
        saticiInstance.delete(flush: true)
        flash.message = "Satici ${params.id} deleted"
        redirect(action: list)}
      catch (org.springframework.dao.DataIntegrityViolationException e) {
        flash.message = "Satici ${params.id} could not be deleted"
        redirect(action: show, id: params.id)}}
    else {
      flash.message = "Satici not found with id ${params.id}"
      redirect(action: list)}}
  def edit = {
    def saticiInstance = Satici.get(params.id)
    if (!saticiInstance) {
      flash.message = "Satici not found with id ${params.id}"
      redirect(action: list)    }
    else {
      return [saticiInstance: saticiInstance]}  }
  def update = {
    def saticiInstance = Satici.get(params.id)
    if (saticiInstance) {
      if (params.version) {
        def version = params.version.toLong()
        if (saticiInstance.version > version) {
```

```
          saticiInstance.errors.rejectValue("version",
"satici.optimistic.locking.failure", "Another user has updated this
Satici while you were editing.")
          render(view: 'edit', model: [saticiInstance: saticiInstance])
          return        }}
      saticiInstance.properties = params
      if (!saticiInstance.hasErrors() && saticiInstance.save()) {
        flash.message = "Satici ${params.id} updated"
        redirect(action: show, id: saticiInstance.id)}
      else {
        render(view: 'edit', model: [saticiInstance: saticiInstance])}}
    else {
      flash.message = "Satici not found with id ${params.id}"
      redirect(action: list)}}
  def create = {
    def saticiInstance = new Satici()
    saticiInstance.properties = params
    return ['saticiInstance': saticiInstance]}
  def save = {
    def saticiInstance = new Satici(params)
    if (!saticiInstance.hasErrors() && saticiInstance.save()) {
      flash.message = "Satici ${saticiInstance.id} created"
      redirect(action: show, id: saticiInstance.id)}
    else {
      render(view: 'create', model: [saticiInstance: saticiInstance])}}}


class MuhendisController {
  def beforeInterceptor = [action: this.&checkUser]
  def checkUser() {
    if (!session.yonetici) {
      redirect(controller: 'yoneticiLogin', action: 'index')
      return false}}
  def index = { redirect(action: list, params: params) }
  // the delete, save and update actions only accept POST requests
  static allowedMethods = [delete: 'POST', save: 'POST', update: 'POST']
  def list = {
    params.max = Math.min(params.max ? params.max.toInteger() : 10, 100)
    [muhendisInstanceList: Muhendis.list(params), muhendisInstanceTotal:
Muhendis.count()]}
  def show = {
    def muhendisInstance = Muhendis.get(params.id)
    if (!muhendisInstance) {
      flash.message = "Muhendis not found with id ${params.id}"
      redirect(action: list)}
    else { return [muhendisInstance: muhendisInstance] }}
  def delete = {
    def muhendisInstance = Muhendis.get(params.id)
    if (muhendisInstance) {
      try {
        muhendisInstance.delete(flush: true)
        flash.message = "Muhendis ${params.id} deleted"
        redirect(action: list)      }
      catch (org.springframework.dao.DataIntegrityViolationException e) {
        flash.message = "Muhendis ${params.id} could not be deleted"
        redirect(action: show, id: params.id)      }    }
    else {
      flash.message = "Muhendis not found with id ${params.id}"
      redirect(action: list)}  }
  def edit = {
    def muhendisInstance = Muhendis.get(params.id)
```

```
    if (!muhendisInstance) {
      flash.message = "Muhendis not found with id ${params.id}"
      redirect(action: list)}
    else {
      return [muhendisInstance: muhendisInstance]}}
  def update = {
    def muhendisInstance = Muhendis.get(params.id)
    if (muhendisInstance) {
      if (params.version) {
        def version = params.version.toLong()
        if (muhendisInstance.version > version) {
          muhendisInstance.errors.rejectValue("version",
"muhendis.optimistic.locking.failure", "Another user has updated this
Muhendis while you were editing.")
          render(view: 'edit', model: [muhendisInstance:
muhendisInstance])
          return}}
      muhendisInstance.properties = params
      if (!muhendisInstance.hasErrors() && muhendisInstance.save()) {
        flash.message = "Muhendis ${params.id} updated"
        redirect(action: show, id: muhendisInstance.id)}
      else {
        render(view: 'edit', model: [muhendisInstance:
muhendisInstance])}}}
    else {
      flash.message = "Muhendis not found with id ${params.id}"
      redirect(action: list)}}
  def create = {
    def muhendisInstance = new Muhendis()
    muhendisInstance.properties = params
    return ['muhendisInstance': muhendisInstance]}
  def save = {
    def muhendisInstance = new Muhendis(params)
    if (!muhendisInstance.hasErrors() && muhendisInstance.save()) {
      flash.message = "Muhendis ${muhendisInstance.id} created"
      redirect(action: show, id: muhendisInstance.id)}
    else {
   render(view: 'create', model: [muhendisInstance: muhendisInstance])}}}


class KomisyoncuController {
  def beforeInterceptor = [action: this.&checkUser]
  def checkUser() {
    if (!session.yonetici) {
      redirect(controller: 'yoneticiLogin', action: 'index')
      return false}  }
  def index = { redirect(action: list, params: params) }
  // the delete, save and update actions only accept POST requests
  static allowedMethods = [delete: 'POST', save: 'POST', update: 'POST']
  def list = {
    params.max = Math.min(params.max ? params.max.toInteger() : 10, 100)
    [komisyoncuInstanceList: Komisyoncu.list(params),
komisyoncuInstanceTotal: Komisyoncu.count()]}
  def show = {
    def komisyoncuInstance = Komisyoncu.get(params.id)
    if (!komisyoncuInstance) {
      flash.message = "Komisyoncu not found with id ${params.id}"
      redirect(action: list)}
    else { return [komisyoncuInstance: komisyoncuInstance] }}
  def delete = {
    def komisyoncuInstance = Komisyoncu.get(params.id)
```

```
    if (komisyoncuInstance) {
      try {
        komisyoncuInstance.delete(flush: true)
        flash.message = "Komisyoncu ${params.id} deleted"
        redirect(action: list)
      }
      catch (org.springframework.dao.DataIntegrityViolationException e) {
        flash.message = "Komisyoncu ${params.id} could not be deleted"
        redirect(action: show, id: params.id)}}
    else {
      flash.message = "Komisyoncu not found with id ${params.id}"
      redirect(action: list)}}
  def edit = {
    def komisyoncuInstance = Komisyoncu.get(params.id)
    if (!komisyoncuInstance) {
      flash.message = "Komisyoncu not found with id ${params.id}"
      redirect(action: list)}
    else {
      return [komisyoncuInstance: komisyoncuInstance]}}
  def update = {
    def komisyoncuInstance = Komisyoncu.get(params.id)
    if (komisyoncuInstance) {
      if (params.version) {
        def version = params.version.toLong()
        if (komisyoncuInstance.version > version) {
          komisyoncuInstance.errors.rejectValue("version",
"komisyoncu.optimistic.locking.failure", "Another user has updated this
Komisyoncu while you were editing.")
          render(view: 'edit', model: [komisyoncuInstance:
komisyoncuInstance])
          return}}
      komisyoncuInstance.properties = params
      if (!komisyoncuInstance.hasErrors() && komisyoncuInstance.save()) {
        flash.message = "Komisyoncu ${params.id} updated"
        redirect(action: show, id: komisyoncuInstance.id)}
      else {
        render(view: 'edit', model: [komisyoncuInstance:
komisyoncuInstance])}}
    else {
      flash.message = "Komisyoncu not found with id ${params.id}"
      redirect(action: list)}}
  def create = {
    def komisyoncuInstance = new Komisyoncu()
    komisyoncuInstance.properties = params
    return ['komisyoncuInstance': komisyoncuInstance]}
  def save = {
    def komisyoncuInstance = new Komisyoncu(params)
    if (!komisyoncuInstance.hasErrors() && komisyoncuInstance.save()) {
      flash.message = "Komisyoncu ${komisyoncuInstance.id} created"
      redirect(action: show, id: komisyoncuInstance.id)}
    else {
      render(view: 'create', model: [komisyoncuInstance:
komisyoncuInstance])}}


class IlController {
  def beforeInterceptor = [action: this.&checkUser]
  def checkUser() {
    if (!session.yonetici) {
      redirect(controller: 'yoneticiLogin', action: 'index')
      return false}}
```

```
  def index = { redirect(action: list, params: params) }
  // the delete, save and update actions only accept POST requests
  static allowedMethods = [delete: 'POST', save: 'POST', update: 'POST']
  def list = {
    params.max = Math.min(params.max ? params.max.toInteger() : 10, 100)
    [ilInstanceList: Il.list(params), ilInstanceTotal: Il.count()]}
  def show = {
    def ilInstance = Il.get(params.id)
    if (!ilInstance) {
      flash.message = "Il not found with id ${params.id}"
      redirect(action: list)}
    else { return [ilInstance: ilInstance] }}
  def delete = {
    def ilInstance = Il.get(params.id)
    if (ilInstance) {
      try {
        ilInstance.delete(flush: true)
        flash.message = "Il ${params.id} deleted"
        redirect(action: list)}
      catch (org.springframework.dao.DataIntegrityViolationException e) {
        flash.message = "Il ${params.id} could not be deleted"
        redirect(action: show, id: params.id)}}
    else {
      flash.message = "Il not found with id ${params.id}"
      redirect(action: list)}}
  def edit = {
    def ilInstance = Il.get(params.id)
    if (!ilInstance) {
      flash.message = "Il not found with id ${params.id}"
      redirect(action: list)}
    else {
      return [ilInstance: ilInstance]}}
  def update = {
    def ilInstance = Il.get(params.id)
    if (ilInstance) {
      if (params.version) {
        def version = params.version.toLong()
        if (ilInstance.version > version) {
          ilInstance.errors.rejectValue("version",
"il.optimistic.locking.failure", "Another user has updated this Il while
you were editing.")
          render(view: 'edit', model: [ilInstance: ilInstance])
          return}}
      ilInstance.properties = params
      if (!ilInstance.hasErrors() && ilInstance.save()) {
        flash.message = "Il ${params.id} updated"
        redirect(action: show, id: ilInstance.id)}
      else {
        render(view: 'edit', model: [ilInstance: ilInstance])}}
    else {
      flash.message = "Il not found with id ${params.id}"
      redirect(action: list)}}
  def create = {
    def ilInstance = new Il()
    ilInstance.properties = params
    return ['ilInstance': ilInstance]}
  def save = {
    def ilInstance = new Il(params)
    if (!ilInstance.hasErrors() && ilInstance.save()) {
      flash.message = "Il ${ilInstance.id} created"
      redirect(action: show, id: ilInstance.id)}
```

```
    else {
      render(view: 'create', model: [ilInstance: ilInstance])}}}


class IlceController {
  def beforeInterceptor = [action: this.&checkUser]
  def checkUser() {
    if (!session.yonetici) {
      redirect(controller: 'yoneticiLogin', action: 'index')
      return false}}
  def index = { redirect(action: list, params: params) }
  // the delete, save and update actions only accept POST requests
  static allowedMethods = [delete: 'POST', save: 'POST', update: 'POST']
  def list = {
    params.max = Math.min(params.max ? params.max.toInteger() : 10, 100)
    [ilceInstanceList: Ilce.list(params), ilceInstanceTotal:
Ilce.count()]}
  def show = {
    def ilceInstance = Ilce.get(params.id)
    if (!ilceInstance) {
      flash.message = "Ilce not found with id ${params.id}"
      redirect(action: list)}
    else { return [ilceInstance: ilceInstance] }}
  def delete = {
    def ilceInstance = Ilce.get(params.id)
    if (ilceInstance) {
      try {
        ilceInstance.delete(flush: true)
        flash.message = "Ilce ${params.id} deleted"
        redirect(action: list)}
      catch (org.springframework.dao.DataIntegrityViolationException e) {
        flash.message = "Ilce ${params.id} could not be deleted"
        redirect(action: show, id: params.id)}}
    else {
      flash.message = "Ilce not found with id ${params.id}"
      redirect(action: list)}}
  def edit = {
    def ilceInstance = Ilce.get(params.id)
    if (!ilceInstance) {
      flash.message = "Ilce not found with id ${params.id}"
      redirect(action: list)}
    else {
      return [ilceInstance: ilceInstance]}}
  def update = {
    def ilceInstance = Ilce.get(params.id)
    if (ilceInstance) {
      if (params.version) {
        def version = params.version.toLong()
        if (ilceInstance.version > version) {
          ilceInstance.errors.rejectValue("version",
"ilce.optimistic.locking.failure", "Another user has updated this Ilce
while you were editing.")
          render(view: 'edit', model: [ilceInstance: ilceInstance])
          return}}
      ilceInstance.properties = params
      if (!ilceInstance.hasErrors() && ilceInstance.save()) {
        flash.message = "Ilce ${params.id} updated"
        redirect(action: show, id: ilceInstance.id)}
      else {
        render(view: 'edit', model: [ilceInstance: ilceInstance])}}
    else {
```

```
      flash.message = "Ilce not found with id ${params.id}"
      redirect(action: list)}}
  def create = {
    def ilceInstance = new Ilce()
    ilceInstance.properties = params
    return ['ilceInstance': ilceInstance]}
  def save = {
    def ilceInstance = new Ilce(params)
    if (!ilceInstance.hasErrors() && ilceInstance.save()) {
      flash.message = "Ilce ${ilceInstance.id} created"
      redirect(action: show, id: ilceInstance.id)}
    else {
      render(view: 'create', model: [ilceInstance: ilceInstance])}}}


class IlacSaticisiController {
  def beforeInterceptor = [action: this.&checkUser]
  def checkUser() {
    if (!session.yonetici) {
      redirect(controller: 'yoneticiLogin', action: 'index')
      return false    }  }
  def index = { redirect(action: list, params: params) }
  // the delete, save and update actions only accept POST requests
  static allowedMethods = [delete: 'POST', save: 'POST', update: 'POST']
  def list = {
    params.max = Math.min(params.max ? params.max.toInteger() : 10, 100)
    [ilacSaticisiInstanceList: IlacSaticisi.list(params),
ilacSaticisiInstanceTotal: IlacSaticisi.count()]}
  def show = {
    def ilacSaticisiInstance = IlacSaticisi.get(params.id)
    if (!ilacSaticisiInstance) {
      flash.message = "IlacSaticisi not found with id ${params.id}"
      redirect(action: list)    }
    else { return [ilacSaticisiInstance: ilacSaticisiInstance] }}
  def delete = {
    def ilacSaticisiInstance = IlacSaticisi.get(params.id)
    if (ilacSaticisiInstance) {
      try {
        ilacSaticisiInstance.delete(flush: true)
        flash.message = "IlacSaticisi ${params.id} deleted"
        redirect(action: list)
      }
      catch (org.springframework.dao.DataIntegrityViolationException e) {
        flash.message = "IlacSaticisi ${params.id} could not be deleted"
        redirect(action: show, id: params.id)      }    }
    else {
      flash.message = "IlacSaticisi not found with id ${params.id}"
      redirect(action: list)    }  }
  def edit = {
    def ilacSaticisiInstance = IlacSaticisi.get(params.id)
    if (!ilacSaticisiInstance) {
      flash.message = "IlacSaticisi not found with id ${params.id}"
      redirect(action: list)    }
    else {
      return [ilacSaticisiInstance: ilacSaticisiInstance]    }  }
  def update = {
    def ilacSaticisiInstance = IlacSaticisi.get(params.id)
    if (ilacSaticisiInstance) {
      if (params.version) {
        def version = params.version.toLong()
        if (ilacSaticisiInstance.version > version) {
```

```
          ilacSaticisiInstance.errors.rejectValue("version",
"ilacSaticisi.optimistic.locking.failure", "Another user has updated this
IlacSaticisi while you were editing.")
          render(view: 'edit', model: [ilacSaticisiInstance:
ilacSaticisiInstance])
          return}}
      ilacSaticisiInstance.properties = params
   if (!ilacSaticisiInstance.hasErrors() && ilacSaticisiInstance.save()){
        flash.message = "IlacSaticisi ${params.id} updated"
        redirect(action: show, id: ilacSaticisiInstance.id)    }
      else {
        render(view: 'edit', model: [ilacSaticisiInstance:
ilacSaticisiInstance])      }    }
    else {
      flash.message = "IlacSaticisi not found with id ${params.id}"
      redirect(action: list)}}
  def create = {
    def ilacSaticisiInstance = new IlacSaticisi()
    ilacSaticisiInstance.properties = params
    return ['ilacSaticisiInstance': ilacSaticisiInstance]}
  def save = {
    def ilacSaticisiInstance = new IlacSaticisi(params)
    if (!ilacSaticisiInstance.hasErrors() && ilacSaticisiInstance.save())
{
      flash.message = "IlacSaticisi ${ilacSaticisiInstance.id} created"
      redirect(action: show, id: ilacSaticisiInstance.id)}
    else {
      render(view: 'create', model: [ilacSaticisiInstance:
ilacSaticisiInstance])}}}


class IlacController {
  def beforeInterceptor = [action: this.&checkUser]
  def checkUser() {
    if (!session.yonetici) {
      redirect(controller: 'yoneticiLogin', action: 'index')
      return false}}
  def index = { redirect(action: list, params: params) }
  // the delete, save and update actions only accept POST requests
  static allowedMethods = [delete: 'POST', save: 'POST', update: 'POST']
  def list = {
    params.max = Math.min(params.max ? params.max.toInteger() : 10, 100)
    [ilacInstanceList: Ilac.list(params), ilacInstanceTotal:
Ilac.count()]}
  def show = {
    def ilacInstance = Ilac.get(params.id)
    if (!ilacInstance) {
      flash.message = "Ilac not found with id ${params.id}"
      redirect(action: list)}
    else { return [ilacInstance: ilacInstance] }}
  def delete = {
    def ilacInstance = Ilac.get(params.id)
    if (ilacInstance) {
      try {
        ilacInstance.delete(flush: true)
        flash.message = "Ilac ${params.id} deleted"
        redirect(action: list)}
      catch (org.springframework.dao.DataIntegrityViolationException e) {
        flash.message = "Ilac ${params.id} could not be deleted"
        redirect(action: show, id: params.id)}}
    else {
```

```
      flash.message = "Ilac not found with id ${params.id}"
      redirect(action: list)}}
  def edit = {
    def ilacInstance = Ilac.get(params.id)
    if (!ilacInstance) {
      flash.message = "Ilac not found with id ${params.id}"
      redirect(action: list)}
    else {
      return [ilacInstance: ilacInstance]}}
  def update = {
    def ilacInstance = Ilac.get(params.id)
    if (ilacInstance) {
      if (params.version) {
        def version = params.version.toLong()
        if (ilacInstance.version > version) {
          ilacInstance.errors.rejectValue("version",
"ilac.optimistic.locking.failure", "Another user has updated this Ilac
while you were editing.")
          render(view: 'edit', model: [ilacInstance: ilacInstance])
          return}}
      ilacInstance.properties = params
      if (!ilacInstance.hasErrors() && ilacInstance.save()) {
        flash.message = "Ilac ${params.id} updated"
        redirect(action: show, id: ilacInstance.id)}
      else {
        render(view: 'edit', model: [ilacInstance: ilacInstance])}}
    else {
      flash.message = "Ilac not found with id ${params.id}"
      redirect(action: list)}}
  def create = {
    def ilacInstance = new Ilac()
    ilacInstance.properties = params
    return ['ilacInstance': ilacInstance]}
  def save = {
    def ilacInstance = new Ilac(params)
    if (!ilacInstance.hasErrors() && ilacInstance.save()) {
      flash.message = "Ilac ${ilacInstance.id} created"
      redirect(action: show, id: ilacInstance.id)}
    else {
      render(view: 'create', model: [ilacInstance: ilacInstance])}}}


class CiftciController {
  def beforeInterceptor = [action: this.&checkUser]
  def checkUser() {
    if (!session.yonetici) {
      redirect(controller: 'yoneticiLogin', action: 'index')
      return false   }  }
  def index = { redirect(action: list, params: params) }
  // the delete, save and update actions only accept POST requests
  static allowedMethods = [delete: 'POST', save: 'POST', update: 'POST']
  def list = {
    params.max = Math.min(params.max ? params.max.toInteger() : 10, 100)
    [ciftciInstanceList: Ciftci.list(params), ciftciInstanceTotal:
Ciftci.count()]  }
  def show = {
    def ciftciInstance = Ciftci.get(params.id)
    if (!ciftciInstance) {
      flash.message = "Ciftci not found with id ${params.id}"
      redirect(action: list)}
    else { return [ciftciInstance: ciftciInstance] }}
```

```
  def delete = {
    def ciftciInstance = Ciftci.get(params.id)
    if (ciftciInstance) {
      try {
        ciftciInstance.delete(flush: true)
        flash.message = "Ciftci ${params.id} deleted"
        redirect(action: list)}
      catch (org.springframework.dao.DataIntegrityViolationException e) {
        flash.message = "Ciftci ${params.id} could not be deleted"
        redirect(action: show, id: params.id)}}
    else {
      flash.message = "Ciftci not found with id ${params.id}"
      redirect(action: list)}}
  def edit = {
    def ciftciInstance = Ciftci.get(params.id)
    if (!ciftciInstance) {
      flash.message = "Ciftci not found with id ${params.id}"
      redirect(action: list)}
    else {
      return [ciftciInstance: ciftciInstance]}}
  def update = {
    def ciftciInstance = Ciftci.get(params.id)
    if (ciftciInstance) {
      if (params.version) {
        def version = params.version.toLong()
        if (ciftciInstance.version > version) {
          ciftciInstance.errors.rejectValue("version",
"ciftci.optimistic.locking.failure", "Another user has updated this
Ciftci while you were editing.")
          render(view: 'edit', model: [ciftciInstance: ciftciInstance])
          return        }      }
      ciftciInstance.properties = params
      if (!ciftciInstance.hasErrors() && ciftciInstance.save()) {
        flash.message = "Ciftci ${params.id} updated"
        redirect(action: show, id: ciftciInstance.id)}
      else {
        render(view: 'edit', model: [ciftciInstance: ciftciInstance])}}
    else {
      flash.message = "Ciftci not found with id ${params.id}"
      redirect(action: list)}}
  def create = {
    def ciftciInstance = new Ciftci()
    ciftciInstance.properties = params
    return ['ciftciInstance': ciftciInstance]  }
  def save = {
    def ciftciInstance = new Ciftci(params)
    if (!ciftciInstance.hasErrors() && ciftciInstance.save()) {
      flash.message = "Ciftci ${ciftciInstance.id} created"
      redirect(action: show, id: ciftciInstance.id)}
    else {
      render(view: 'create', model: [ciftciInstance: ciftciInstance])}}}


class YeniSaticiPartiController {
  def beforeInterceptor = [action: this.&checkUser]
  def checkUser() {
    if (!session.satici) {
      redirect(controller: 'saticiLogin', action: 'index')
      return false
    } else {}}
  def index = { redirect(action: fill, params: params) }
```

```
  def fill = {
    [uruns: Urun.findAll()]  }
  def kaydet = {
    if (params.partiNo != null && params.urunid != null) {
      def kp = SaticiParti.findAllBySaticiAndPartiNo(session.satici,
params.partiNo)
      if (kp) {
        flash.message = params.partiNo + " isimli parti daha Ã¶nceden
oluÅŸturulmuÅŸ."
        redirect(action: fill)
        return false
      } else {
        session.yeniSaticiPartiNo = params.partiNo
        session.yeniSaticiPartiUrunid = params.urunid
        def urun = Urun.findById(params.urunid)
        def saticiParti = new SaticiParti()
        saticiParti.setUrun(urun)
        saticiParti.setOlusturulmaTarihi(new Date())
        saticiParti.setSatici(session.satici)
        saticiParti.setPartiNo(params.partiNo)
        saticiParti.save()
        session.saticiParti = saticiParti;
        flash.message = params.partiNo + " isimli yeni parti
oluÅŸturuldu."
        [saticiPartiSaglayicilari:
SaticiPartiSaglayicisi.findAllBySaticiParti(session.saticiParti),
                komisyoncus: Komisyoncu.findAll()]}
    } else {
      if (session.yeniSaticiPartiNo != null &&
session.yeniSaticiPartiUrunid != null) {
        flash.message = session.yeniSaticiPartiNo + " isimli yeni parti
no kullanÄ±lÄ±yor."
        [saticiPartiSaglayicilari:
SaticiPartiSaglayicisi.findAllBySaticiParti(session.saticiParti),
                komisyoncus: Komisyoncu.findAll()]
      } else {
        flash.message = "Bir parti no giriniz."
        redirect(action: fill)}}}
  def ciftciEkle = {
    if (params.tckno != null && params.tckno.trim().length() > 0) {
      def ciftci = Ciftci.findWhere(tckno: params.tckno)
      def saticiPartiSaglayicisi = new SaticiPartiSaglayicisi()
      saticiPartiSaglayicisi.setSaticiParti(session.saticiParti)
      saticiPartiSaglayicisi.setCiftci(ciftci)
      saticiPartiSaglayicisi.save()
    } else {
      flash.message2 = "Bir TC Kimlik No giriniz."    }
    redirect(action: kaydet)}
  def ciftciSil = {
    if (params.id != null && params.id.trim().length() > 0) {
      def ciftci = Ciftci.findById(params.id)
      def saticiPartiSaglayicisilar =
SaticiPartiSaglayicisi.findAllBySaticiPartiAndCiftci(session.saticiParti,
ciftci);
      saticiPartiSaglayicisilar.each {
        it.delete()}
      flash.message2 = ciftci.tckno + " TC Kimlik No'lu Ã§iftÃ§i bu
partiden Ã§Ä±karÄ±ldÄ±."
    } else {
      flash.message2 = "Bir Ã§iftÃ§i seÃ§iniz."}
    redirect(action: kaydet)}
```

```
  def komisyoncuEkle = {
    if (params.komisyoncuid != null && params.partiNo != null) {
      def komisyoncu = Komisyoncu.findById(params.komisyoncuid)
      def komisyoncuParti =
KomisyoncuParti.findAllByKomisyoncuAndPartiNo(komisyoncu, params.partiNo)
      if (komisyoncuParti) {
        def saticiPartiSaglayicisi = new SaticiPartiSaglayicisi()
        saticiPartiSaglayicisi.setSaticiParti(session.saticiParti)
        saticiPartiSaglayicisi.setKomisyoncu(komisyoncu)
        saticiPartiSaglayicisi.setPartiNo(params.partiNo)
        saticiPartiSaglayicisi.save()
      } else {
        flash.message2 = "Girilen parti no bu komisyoncuya ait deÄŸil." }
    } else {
      flash.message2 = "Bir komisyoncu seÃ§iniz ve o komisyoncuya ait bir
parti no giriniz."}
    redirect(action: kaydet)}
  def komisyoncuSil = {
    if (params.id != null && params.id.trim().length() > 0) {
      def komisyoncu = Komisyoncu.findById(params.id)
      def kps =
SaticiPartiSaglayicisi.findAllBySaticiPartiAndKomisyoncu(session.saticiPa
rti, komisyoncu)
      kps.each {
        it.delete()      }
      flash.message2 = komisyoncu.ad + " " + komisyoncu.adres + "
komisyoncusu bu partiden Ã§Ä±karÄ±ldÄ±."
    } else {
      flash.message2 = "Bir Ã§iftÃ§i seÃ§iniz."    }
    redirect(action: kaydet)}}


class YeniIlacSatisController {
  def beforeInterceptor = [action: this.&checkUser]
  def checkUser() {
    if (!session.ilacSaticisi) {
      redirect(controller: 'ilacSaticisiLogin', action: 'index')
      return false}}
  def index = { redirect(action: fill, params: params) }
  def fill = {
    [ilacs: Ilac.findAll()]}
  def kaydet = {
    if(params.tckno!=null && params.ilacids!=null){
      def ciftci = Ciftci.findWhere(tckno:params.tckno)
      if(ciftci!=null){
        params.ilacids
        def ilacs = [];
        params.ilacids.each {
          def ilacCiftciSatis = new IlacCiftciSatis();
          ilacCiftciSatis.setIslemTarihi(new Date());
          def ilacSaticisi = session.ilacSaticisi;
          ilacCiftciSatis.setIlacSaticisi(ilacSaticisi);
            def i = Ilac.findById(it);
            ilacs.add(i)
          ilacCiftciSatis.setIlac(i);
          ilacCiftciSatis.setCiftci(ciftci);
          ilacCiftciSatis.save();}
        flash.message = "SatÄ±ÅŸ baÅŸarÄ±lÄ±; "+params.tckno+" TC Kimlik
Nolu Ã§iftÃ§iye ÅŸu ilaÃ§lar satÄ±ldÄ±: "+ilacs
      }else{
```

```
        flash.message = "Bu TCK No ile kayÄ±tlÄ± bir Ã§iftÃ§i yok, TC
Kimlik No: "+params.tckno}
    }else{
      flash.message = "Bir TC Kimlik No giriniz ve en az bir ilaÃ§
seÃ§iniz."
      redirect(action: fill)}}}


class YeniKomisyoncuPartiController {
  def beforeInterceptor = [action: this.&checkUser]
  def checkUser() {
    if (!session.komisyoncu) {
      redirect(controller: 'komisyoncuLogin', action: 'index')
      return false
    } else {}}
  def index = { redirect(action: fill, params: params) }
  def fill = {
    [uruns: Urun.findAll()]}
  def kaydet = {
    if (params.partiNo != null && params.urunid != null) {
      def kp =
KomisyoncuParti.findAllByKomisyoncuAndPartiNo(session.komisyoncu,
params.partiNo)
      if (kp) {
        flash.message = params.partiNo + " isimli parti daha Ã¶nceden
oluÅŸturulmuÅŸ."
        redirect(action: fill)
        return false
      } else {
        session.yeniKomisyoncuPartiNo = params.partiNo
        session.yeniKomisyoncuPartiUrunid = params.urunid
        def urun = Urun.findById(params.urunid)
        def komisyoncuParti = new KomisyoncuParti()
        komisyoncuParti.setUrun(urun)
        komisyoncuParti.setOlusturulmaTarihi(new Date())
        komisyoncuParti.setKomisyoncu(session.komisyoncu)
        komisyoncuParti.setPartiNo(params.partiNo)
        komisyoncuParti.save()
        session.komisyoncuParti = komisyoncuParti;
        flash.message = params.partiNo + " isimli yeni parti
oluÅŸturuldu."
        [ilacs: Ilac.findAll(),
                KomisyoncuPartiSaglayicilari:
KomisyoncuPartiSaglayicisi.findAllByKomisyoncuParti(session.komisyoncuPar
ti),             komisyoncus: Komisyoncu.findAll()]}
    } else {
      if (session.yeniKomisyoncuPartiNo != null &&
session.yeniKomisyoncuPartiUrunid != null) {
        flash.message = session.yeniKomisyoncuPartiNo + " isimli yeni
parti no kullanÄ±lÄ±yor."
        [ilacs: Ilac.findAll(),
                KomisyoncuPartiSaglayicilari:
KomisyoncuPartiSaglayicisi.findAllByKomisyoncuParti(session.komisyoncuPar
ti),                komisyoncus: Komisyoncu.findAll()]
      } else {
        flash.message = "Bir parti no giriniz."
        redirect(action: fill)}}}
  def ciftciEkle = {
    if (params.tckno != null && params.tckno.trim().length() > 0) {
      def ciftci = Ciftci.findWhere(tckno: params.tckno)
      def komisyoncuPartiSaglayicisi = new KomisyoncuPartiSaglayicisi()
```

```
     komisyoncuPartiSaglayicisi.setKomisyoncuParti(session.komisyoncuPart
i)
      komisyoncuPartiSaglayicisi.setCiftci(ciftci)
      Calendar calendar = new GregorianCalendar()
      if (params.ilacKullanimTarihi_year != null &&
params.ilacKullanimTarihi_month != null &&
params.ilacKullanimTarihi_day != null) {
        calendar.set(Integer.valueOf(params.ilacKullanimTarihi_year),
Integer.valueOf(params.ilacKullanimTarihi_month),
Integer.valueOf(params.ilacKullanimTarihi_day))}
      komisyoncuPartiSaglayicisi.setIlacKullanimTarihi(calendar.getTime()
)
      komisyoncuPartiSaglayicisi.save()
      String str = params.ilacids.toString();
      if (params.ilacids != null) {
        if (str.lastIndexOf(',') != -1) {
          params.ilacids.each {
            def ilac = Ilac.findById(it)
            def komisyoncuPartiCiftciIlac = new
KomisyoncuPartiCiftciIlac()
komisyoncuPartiCiftciIlac.setKomisyoncuPartiSaglayicisi(komisyoncuPartiSa
glayicisi)
            komisyoncuPartiCiftciIlac.setIlac(ilac);
            komisyoncuPartiCiftciIlac.save()}
        } else {
          def ilac = Ilac.findById(params.ilacids)
          def komisyoncuPartiCiftciIlac = new KomisyoncuPartiCiftciIlac()
komisyoncuPartiCiftciIlac.setKomisyoncuPartiSaglayicisi(komisyoncuPartiSa
glayicisi)
          komisyoncuPartiCiftciIlac.setIlac(ilac);
          komisyoncuPartiCiftciIlac.save()}}
    } else {
      flash.message2 = "Bir TC Kimlik No giriniz."}
    redirect(action: kaydet)}
  def ciftciSil = {
    if (params.id != null && params.id.trim().length() > 0) {
      def ciftci = Ciftci.findById(params.id)
      def komisyoncuPartiSaglayicisilar =
KomisyoncuPartiSaglayicisi.findAllByKomisyoncuPartiAndCiftci(session.komi
syoncuParti, ciftci);
      komisyoncuPartiSaglayicisilar.each {
        def komisyoncuPartiCiftciIlac =
KomisyoncuPartiCiftciIlac.findAllByKomisyoncuPartiSaglayicisi(it)
        komisyoncuPartiCiftciIlac.each {
          it.delete()}
        it.delete()}
      flash.message2 = ciftci.tckno + " TC Kimlik No'lu Ã§iftÃ§i bu
partiden Ã§Ã±karÃ±ldÃ±."
    } else {
      flash.message2 = "Bir Ã§iftÃ§i seÃ§iniz."}
    redirect(action: kaydet)}
  def komisyoncuEkle = {
    if (params.komisyoncuid != null && params.partiNo != null) {
      def komisyoncu = Komisyoncu.findById(params.komisyoncuid)
      def komisyoncuParti =
KomisyoncuParti.findAllByKomisyoncuAndPartiNo(komisyoncu, params.partiNo)
      if (komisyoncuParti) {
        def komisyoncuPartiSaglayicisi = new KomisyoncuPartiSaglayicisi()
  komisyoncuPartiSaglayicisi.setKomisyoncuParti(session.komisyoncuParti)
        komisyoncuPartiSaglayicisi.setKomisyoncu(komisyoncu)
        komisyoncuPartiSaglayicisi.setPartiNo(params.partiNo)
```

```
        komisyoncuPartiSaglayicisi.save()
      } else {
        flash.message2 = "Girilen parti no bu komisyoncuya ait deÄŸil."}
    } else {
      flash.message2 = "Bir komisyoncu seÃ§iniz ve o komisyoncuya ait bir
parti no giriniz."}
    redirect(action: kaydet)}
  def komisyoncuSil = {
    if (params.id != null && params.id.trim().length() > 0) {
      def komisyoncu = Komisyoncu.findById(params.id)
      def kps =
KomisyoncuPartiSaglayicisi.findAllByKomisyoncuPartiAndKomisyoncu(session.
komisyoncuParti, komisyoncu)
      kps.each {
        it.delete()}
      flash.message2 = komisyoncu.ad + " " + komisyoncu.adres + "
komisyoncu bu partiden Ã§Ä±karÄ±ldÄ±."
    } else {
      flash.message2 = "Bir Ã§iftÃ§i seÃ§iniz."}
    redirect(action: kaydet)}}


class YeniRaporController {
  def beforeInterceptor = [action: this.&checkUser]
  def checkUser() {
    if (!session.muhendis) {
      redirect(controller: 'muhendisLogin', action: 'index')
      return false
    } else {}}
  def index = { redirect(action: fill, params: params) }
  def fill = {    [
            saticis: Satici.findAll(),
            etkenMaddes: EtkenMadde.findAll(),
            uruns: Urun.findAll()]}
  KomisyoncuParti topkomisyoncuParti(KomisyoncuParti kp) {
    KomisyoncuPartiSaglayicisi kps =
KomisyoncuPartiSaglayicisi.findByKomisyoncuAndPartiNo(kp.getKomisyoncu(),
kp.getPartiNo())
    if (kps != null) {
      return topkomisyoncuParti(kps.getKomisyoncuParti())}
    return kp}
  def raporla = {
    //if (params.saticiid != null && params.partiNo != null &&
params.etkenMaddeids != null) {
    if (params.saticiid != null && params.partiNo != null) {
      def selectedEtkenMaddes = new ArrayList<EtkenMadde>()
      String str = params.etkenMaddeids.toString();
      if(params.etkenMaddeids!=null){
        if(str.lastIndexOf(',')!=-1){
          params.etkenMaddeids.each {
            selectedEtkenMaddes.add(EtkenMadde.findById(it))}
        }else{
     selectedEtkenMaddes.add(EtkenMadde.findById(params.etkenMaddeids))}}
      def saticiParti = SaticiParti.findWhere(satici:
Satici.findById(params.saticiid), partiNo: params.partiNo)
      if (saticiParti) {
        def saticiPartiSaglayicilari =
SaticiPartiSaglayicisi.findAllBySaticiParti(saticiParti)
        List<ResultNode> resultNodes = new ArrayList<ResultNode>();
        saticiPartiSaglayicilari.each {
          def saticiPartiSaglayicisi = (SaticiPartiSaglayicisi) it;
```

```
          def komisyoncu = saticiPartiSaglayicisi.getKomisyoncu()
          def partiNo = saticiPartiSaglayicisi.getPartiNo();
          ResultNode resultNode = new ResultNode();
          if (komisyoncu) {
            resultNode.setKomisyoncu(komisyoncu)
            resultNode.setPartiNo(partiNo)
       resultNode.setOlusturulmaTarihi(saticiParti.getOlusturulmaTarihi()
)
            resultNode.getResultNodes().addAll(this.findAllList(komisyonc
u, partiNo, selectedEtkenMaddes))}
          resultNodes.add(resultNode)}
        [resultNodes: resultNodes]
      } else {
        flash.message = "SeÃ§ilen satÄ±cÄ±ya ait bÃ¶yle bir parti no
yok."
        redirect(action: fill)}
    } else {
      flash.message = "SatÄ±cÄ±, parti no ve etken madde alanlarÄ± boÅŸ
bÄ±rakÄ±lamaz."
      redirect(action: fill)}}
  List<ResultNode> findAllList(Komisyoncu komisyoncu, String partiNo,
ArrayList<EtkenMadde> selectedEtkenMaddes) {
    List<ResultNode> resultNodes = new ArrayList<ResultNode>();
    def komisyoncuParti =
KomisyoncuParti.findByKomisyoncuAndPartiNo(komisyoncu, partiNo)
    def komisyoncuPartiSaglayicilari =
KomisyoncuPartiSaglayicisi.findAllByKomisyoncuParti(komisyoncuParti)
    komisyoncuPartiSaglayicilari.each {
      ResultNode resultNode = new ResultNode();
      KomisyoncuPartiSaglayicisi komisyoncuPartiSaglayicisi =
(KomisyoncuPartiSaglayicisi) it;
      if (komisyoncuPartiSaglayicisi.getKomisyoncu() != null) {
      resultNode.setKomisyoncu(komisyoncuPartiSaglayicisi.getKomisyoncu()
)
        resultNode.setPartiNo(komisyoncuPartiSaglayicisi.getPartiNo())
    resultNode.setOlusturulmaTarihi(komisyoncuParti.getOlusturulmaTarihi(
))
resultNode.getResultNodes().addAll(this.findAllList(komisyoncuPartiSaglay
icisi.getKomisyoncu(), komisyoncuPartiSaglayicisi.getPartiNo(),
selectedEtkenMaddes))
      } else {
        resultNode.setIsCiftci(true)
        resultNode.setCiftci(komisyoncuPartiSaglayicisi.getCiftci())
resultNode.setIlacKullanimTarihi(komisyoncuPartiSaglayicisi.getIlacKullan
imTarihi())
        List<Satici> saticis = new ArrayList<Satici>();
        def cKomisyoncuSaglayicisis =
KomisyoncuPartiSaglayicisi.findAllByCiftci(resultNode.getCiftci())
         KomisyoncuSaglayicisi.each {
          KomisyoncuPartiSaglayicisi ckps = (KomisyoncuPartiSaglayicisi)
it
          KomisyoncuParti cKomisyoncuParti = ckps.getKomisyoncuParti()
          KomisyoncuParti topKomisyoncuParti =
this.topkomisyoncuParti(cKomisyoncuParti)
          SaticiPartiSaglayicisi saticiPartiSaglayicisi =
SaticiPartiSaglayicisi.findByKomisyoncuAndPartiNo(topKomisyoncuParti.getK
omisyoncu(), topKomisyoncuParti.getPartiNo())
          if (saticiPartiSaglayicisi != null) {
            Satici satici =
saticiPartiSaglayicisi.getSaticiParti().getSatici()
            if (!saticis.contains(satici)) {
```

```
              saticis.add(satici)}}}
        resultNode.setCiftciSaticis(saticis)
        def komisyoncuPartiCiftciIlaclari =
KomisyoncuPartiCiftciIlac.findAllByKomisyoncuPartiSaglayicisi(komisyoncuP
artiSaglayicisi)
        def ilacs = new ArrayList<Ilac>()
        def etkenMaddes = new ArrayList<EtkenMadde>()
        komisyoncuPartiCiftciIlaclari.each {
          KomisyoncuPartiCiftciIlac komisyoncuPartiCiftciIlac =
(KomisyoncuPartiCiftciIlac) it;
          Ilac ilac = komisyoncuPartiCiftciIlac.getIlac();
          ilacs.add(ilac)
          def ems = ilac.getEtkenMaddes()
          ems.each {
            EtkenMadde em = (EtkenMadde) it
            selectedEtkenMaddes.each {
              EtkenMadde sem = (EtkenMadde) it
              if (em.getId().equals(sem.getId())) {
                resultNode.setContainsSelectedEtkenMadde(true)}}}
          etkenMaddes.addAll(ems)}
        resultNode.setIlacs(ilacs)
        resultNode.setEtkenMaddes(etkenMaddes)}
      resultNodes.add(resultNode)}
    return resultNodes}}


class SaticiParti {
  Long id
  Urun urun
  Satici satici
  Date olusturulmaTarihi
  String partiNo
  static constraints = {}
  static mapping = {
    table 'satici_parti'
    version false}
  public String toString() {
    return "partiNo:"+partiNo + " urun.ad: " + urun.ad + " satici.ad: " +
satici.ad  + " olusturulmaTarihi: " +  olusturulmaTarihi}}


class SaticiPartiSaglayicisi {
  Long id
  SaticiParti saticiParti
  Komisyoncu komisyoncu
  String partiNo
  Ciftci ciftci
  static constraints = {
    komisyoncu(nullable:true)
    partiNo(nullable:true)
    ciftci(nullable:true)}
  static mapping = {
    table 'satici_parti_saglayicisi'
    version false}
  public String toString() {
    return " id: " + id + " partiNo: " + partiNo + " saticiParti: " +
saticiParti + " komisyoncu: " + komisyoncu + " ciftci: " + ciftci}}


class KomisyoncuPartiSaglayicisi {
```

```
  Long id
  KomisyoncuParti komisyoncuParti
  Komisyoncu komisyoncu
  String partiNo
  Ciftci ciftci
  Date ilacKullanimTarihi
  static constraints = {
    komisyoncu(nullable:true)
    partiNo(nullable:true)
    ciftci(nullable:true)
    ilacKullanimTarihi(nullable:true)}
  static mapping = {
    table 'komisyoncu_parti_saglayicisi'
    version false}
  public String toString() {
    return " id: " + id + " : " + partiNo + " : " + komisyoncuParti + " :
" + komisyoncu + " : " + ciftci}}


class Muhendis {
  Long id
  String kullanici
  String parola
  String adsoyad
  static constraints = {}
  static mapping = {
    table 'muhendis'
    version false}
  public String toString() {
    return adsoyad}}


class Ilac {
  Long id
  String ad
  def hasMany = [etkenMaddes: EtkenMadde]
  Set etkenMaddes = new HashSet()
  static constraints = {}
  static mapping = {
    table 'ilac'
    version false}
  public String toString() {
    return ad}}


class EtkenMadde {
  Long id
  String ad
  def hasMany = [ilacs: Ilac]
  def belongsTo = Ilac
  Set ilacs = new HashSet();
  static constraints = {}
  static mapping = {
    table 'etken_madde'
    version false}
  public String toString() {
    return ad}}


class KomisyoncuPartiCiftciIlac {
```

```
  Long id
  KomisyoncuPartiSaglayicisi komisyoncuPartiSaglayicisi
  Ilac ilac
  static constraints = {}
  static mapping = {
    table 'komisyoncu_parti_ciftci_ilac'
    version false}
  public String toString() {
    return komisyoncuPartiSaglayicisi + " : " + ilac.ad}}


class KomisyoncuParti {
  Long id
  Urun urun
  Komisyoncu komisyoncu
  Date olusturulmaTarihi
  String partiNo
  static constraints = {}
  static mapping = {
    table 'komisyoncu_parti'
    version false}
  public String toString() {
    return partiNo + " : " + urun.ad + " : " + komisyoncu.ad  + " : " +
olusturulmaTarihi}}


class Urun {
  Long id
  String ad
  static constraints = {}
  static mapping = {
    table 'urun'
    version false}
  public String toString() {
    return ad}}


class IlacCiftciSatis {
  Long id
  Date islemTarihi
  IlacSaticisi ilacSaticisi
  Ilac ilac
  Ciftci ciftci
  static constraints = {}
  static mapping = {
    table 'ilac_ciftci_satis'
    version false}
  public String toString() {
    return ilacSaticisi.ad+" : "+ilac.ad+" : "+ciftci.tckno}}


class Yonetici {
  Long id
  String kullanici
  String parola
  static constraints = {}
  static mapping = {
    table 'yonetici'
    version false}
  public String toString() {
```

```
    return kullanici}}


class Satici {
  Long id
  String kullanici
  String parola
  String ad
  String adres
  String telefon
  Ilce ilce
  static belongsTo = Ilce
  static constraints = {}
  static mapping = {
    table 'satici'
    version false}
  public String toString() {
    return ad + " : " + ilce.ad + " : " + adres}}


class IlacSaticisi {
  Long id
  String kullanici
  String parola
  String ad
  String adres
  String telefon
  Ilce ilce
  static belongsTo = Ilce
  static constraints = {}
  static mapping = {
    table 'ilac_saticisi'
    version false}
  public String toString() {
    return ad + " : " + ilce.ad + " : "+ adres}}


class Ciftci {
  Long id
  String ad
  String soyad
  String tckno
  static constraints = {}
  static mapping = {
    table 'ciftci'
    version false}
  public String toString() {
    return ad + " " + soyad}}


class Komisyoncu {
  Long id
  String kullanici
  String parola
  String ad
  String adres
  String telefon
  Ilce ilce
  static belongsTo = Ilce
  static constraints = {}
```

```
  static mapping = {
    table 'komisyoncu'
    version false}
  public String toString() {
    return ad + " : " + ilce.ad + " : "+ adres}}


class Ilce {
  Long id
  String ad
  Il il
  static belongsTo = Il
  static constraints = {}
  static mapping = {
    table 'ilce'
    version false}
  public String toString() {
    return il.ad+" : "+ad}}


class Il {
  Long id
  String ad
  static hasMany = [ilces: Ilce]
  static constraints = {}
  static mapping = {
    table 'il'
    version false}
  public String toString() {
    return ad}}
```

ÖZGEÇMİŞ

Adı Soyadı : EMRAH ORAL

Doğum Yeri : GİRESUN

Doğum Yılı : 1981

Medeni Hali : BEKAR

Eğitim ve Akademik Durumu:

Lise 1994 – 1998

Lisans 1998 – 2004

Yabancı Dil : Kamu Personeli Dil Sınavı (İngilizce) – 73 puan

İş Tecrübesi:

21.09.2005 - ... Tarım ve Köyişleri Bakanlığı, Koruma ve Kontrol Genel Müdürlüğü