

Veritabanı Yönetim Sistemleri Projesi

BiletAI

COLLABORATORS

	<i>TITLE :</i> Veritabanı Yönetim Sistemleri Projesi	<i>REFERENCE :</i>	
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY	Burak Dağılı, Grup 3 , 040060232, Şule Gündüz Öğüdücü	29.12.2011	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	Giriş	1
1.1	Proje Tanımı	1
1.2	Geliştirme ve Çalıştırma Ortamı	1
1.3	İş Bölüşümü	2
2	Kurulum	4
3	Kullanım Kılavuzu	8
4	Teknik Kılavuz	35
4.1	Veritabanı Tasarımı	35
4.2	Yazılım Tasarımı	37

List of Figures

3.1	AnaSayfada Ziyaretçi tipinden kullanıcı için bir görüntü	8
3.2	Sign Up-Log In Formun Ziyaretçi için görünümü	9
3.3	Sign Up-Log In Formun Ziyaretçi dışındaki üyeleri için görünümü	9
3.4	Admin Panelin Ziyaretçi için görünümü	10
3.5	Organizer Panelin Ziyaretçi için görünümü	10
3.6	Admin ve Organizer Panelin Yönetici üyeleri için görünümü	11
3.7	Header Panel görünümü	11
3.8	Header Panelde alt tabların görünümü	12
3.9	Header Panelde alt tabların görünümü	12
3.10	Header Panelde alt tabların görünümü	13
3.11	AnaSayfada arama motoru, duyurular ve tarihte bugünkü köşesi	13
3.12	Arama motorunda kategoriye göre etkinlik arama	14
3.13	Arama motorunda şehire göre etkinlik arama	14
3.14	Arama motorunda kategoriye göre etkinlik arama sonucu	15
3.15	Etkinlik listeleyme sayfası	16
3.16	Etkinlik listeleyme sayfasında etkinlik silme	17
3.17	Etkinlik listesinde EventDisplayPage linki	18
3.18	Yönetici için EventDisplayPage 1	19
3.19	Yönetici için EventDisplayPage 2	20
3.20	Ziyaretçi için EventDisplayPage	21
3.21	Yönetici için EventEditPage	22
3.22	Etkinlikler sayfasında Ajax kullanılarak hazırlanan arama formu	22
3.23	Arama formunda etkinlik show linki	23
3.24	Etkinlik ekleme butonu (add Event)	23
3.25	Etkinlik ekleme sayfası	24
3.26	Kategori listeleyme sayfası	24
3.27	Kategori listeleyme sayfasında yönetici için silme işlemi	25
3.28	Kategori listeleyme sayfasında CategoryDisplayPage için link	25
3.29	Ziyaretçi için CategoryDisplayPage sayfası	26
3.30	Yönetici için CategoryDisplayPage sayfasında güncelleme linki	26

3.31 Yönetici için CategoryEditPage sayfası	27
3.32 Yönetici ve Organizatör için kategori ekleme penceresi	27
3.33 Üst kategori listeleme sayfası	28
3.34 Üst kategori listeleme sayfasında yönetici için silme işlemi	28
3.35 Üst kategori listeleme sayfasında GroupDisplayPage için link	29
3.36 Ziyaretçi için GroupDisplayPage sayfası	29
3.37 Yönetici için GroupDisplayPage sayfasında güncelleme linki	30
3.38 Yönetici için GroupEditPage sayfası	30
3.39 Yönetici ve Organizatör için üst kategori ekleme penceresi	31
3.40 Kişiye ait kredi kartlarının listelendiği sayfa	32
3.41 Kişiye ait kredi kartlarını silme işlemi	32
3.42 Kredi Kartı bilgilerini içeren sayfa	33
3.43 Kredi kartı güncelleme sayfası	33
3.44 Kredi kartı ekleme penceresi	34
4.1 Veritabanı varlık-iliski diyagramı	36

Genel Bilgiler

Chapter 1

Giriş

1.1 Proje Tanımı

BiletAI Projesi, internet kullanıcılarının web ortamında, farklı illerde olan çeşitli başlıklardaki (kültür, sanat, eğitim, spor gibi) etkinliklerin tarih ve yer bilgilerini öğrenebilmelerini ve bilet satın alabilmelerini sağlamak amacıyla düşünülmüş ve hazırlanmıştır. Etkinlik detaylarını öğrenme ve bilet satın alabilmenin yanı sıra kullanıcılar duyuruları takip edebilecek, çeşitli kategorilerde haberleri ve makaleleri görebilecek, bunlara yorum yapabilecek böylece kültür, sanat, spor, müzik hakkında bilgi edinebileceklerdir. Ayrıca siteye üye olabilecekler ve sitede görevli olarak içerik oluşturabileceklerdir.

Projenin içeriği temel özellikler:

4 farklı tipten kullanıcı olacaktır. Bunlar; ziyaretçi, üye(user), organizatör(organizer) ve yöneticidir(admin).

Sistem üzerindeki kullanıcı tipleri anlatılacak olunursa:

Ziyaretçi: Sisteme üye olmayan kullanıcılardır. Bu türden kullanıcılar kategoriye, mekana, şehire göre etkinlikleri listeleyebilir, kontenjanları ve fiyatlarını görebilirler. Duyuruları ve haberleri görebilirler. Tarihte bugün köşesinin inceleyebilirler. Etkinlikler, mekanlar, şehirler hakkında bilgilere ulaşabilirler.

Üye(User): Bu kullanıcı tipi ziyaretçiden ayrı olarak kişisel bilgilerini veritabanına ekler. Ziyaretçinin sahip olduğu tüm özelliklere sahiptir. Tek farkı eklemiş olduğu Kredi Kartı bilgileri ile bilet satın alabilmektedir. Bunun haricinde aynen ziyaretçi gibi siteye hiçbir ekleme, güncelleme yapamaz, siteden içerik silemez. Sign up panelinden üye olan tüm kullanıcılar “üye” tipinden kullanıcılardır.

Organizatör(Organizer): Organizatör adlı kullanıcı yönetici tarafından sisteme eklenebilir ya da kayıtlı üyenin yönetici tarafından tipi güncellenirken organizatör seçilebilir. Bu tip kullanıcı (organizatör), üyenin yapmış olduğu sorgulamalar ve bilet alımı dışında siteye etkinlik, mekan, şehir, duyuru, yorum, haber ve tarihsel önemi olan bir olay (“Tarihte bugün için içerik”) ekleyebilir. Ancak bu içerikler ile ilgili güncelleme ve silme işlemi yapamaz.

Yönetici/Admin: Yönetici tipinden kullanıcı veritabanına önceden eklenmelidir. Ayrıca yönetici girişi yapıldıktan sonra bir kullanıcı oluştururken veya bir kullanıcı güncellenirken tipi yönetici olarak seçilebilir. Bu tip kullanıcı (yönetici), organizatör tipinden kullanıcının yapmış olduğu sorgulamalar, bilet alımı ve eklemeler dışında tüm içerikleri güncelleme ve silme yetkisine sahiptir. Ayrıca siteye üye ekleyebilir, üyeleri listeleyip güncelleyebilir, silebilir ve yeni üye tipi oluşturabilir.

1.2 Geliştirme ve Çalıştırma Ortamı

Proje Java dilinde ve Apache Wicket platformu üzerinde geliştirilmiştir. Belli bölümlerde Ajax kodları da kullanılmıştır. Veritabanı olarak SQLite JDBC sürücüsü kullanılmıştır. Veritabanı dosyası ve tabloları oluşturulurken Mozilla Firefox eklentisi olan SQLite Manager kullanılmıştır. Çalışma sırasında Mercurial ile sürüm denetimi yapılmıştır. Tüm çalışmalar Netbeans IDE 7.0.1'de yapılmıştır.

Proje *Ubuntu 10.10* işletim sistemi yüklü bilgisayarlarda Mozilla Firefox, Chrome, Chromium tarayıcılarında sorunsuz çalışmaktadır. Projenin Windows 7 işletim sistemi yüklü bilgisayarlarda sorun yaşadığını gözlemediğim.

1.3 İş Bölüşümü

Proje önerisi kabul edilip uygulama geliştirmeye başlamadan önce yapılan ilk adım veritabanının tasarımları olmuştur. Veritabanı tasarımları karşılıklı bilgi alışverişi ve tartışma çerçevesinde kolektif bir biçimde ortaya çıkmıştır. Gerekli tablolar, ilişkiler, birincil ve dış anahtarlar kararlaştırılmıştır. Ardından grup üyeleri kendilerine ait ilgili tabloları oluşturmuş ve uygulama geliştirmeye başlamışlardır.

PROJE ÜYELERİ ARASINDA İŞ BÖLÜŞÜMÜ

1. BURAK DAĞLI

1.a. Veritabanı tasarımları

1.b. Sorumlu olduğu 4 nesne için veritabanında "*Event*, *CategoryEvent*, *GroupCategory* ve *Card*" adlı tabloların oluşturulması.

1.c. *BasePage.java*, *Application.java*, *HomePage.java*, *HomePage.html*, *HeaderPanel.java*, *HeaderPanel.html* dosyalarını oluşturulması ve içeriklerinin tasarlanması.

1.d. *Application.java*, *HomePage.java*, *HomePage.html*, *HeaderPanel.java*, *HeaderPanel.html* dosyalarına sorumlu olduğu sınıflarla ilgili içeriklerin ve linklerin eklenmesi.

1.e. Sorumlu olduğu *Card* nesnesi için "*Cards*" paketinin oluşturulması, içindeki html ve java dosyalarının kodlarının yazılması.

1.f. Sorumlu olduğu *Event* nesnesi için "*Events*" paketinin oluşturulması, içindeki html ve java dosyalarının kodlarının yazılması.

1.g. Sorumlu olduğu *CategoryEvent* nesnesi için "*CategoryEvents*" paketinin oluşturulması, içindeki html ve java dosyalarının kodlarının yazılması.

1.h. Sorumlu olduğu *GroupCategory* nesnesi için "*GroupCategories*" paketinin oluşturulması, içindeki html ve java dosyalarının kodlarının yazılması.

1.i. Sorumlu olduğu *Event* nesnesi için "*Search*" paketinin oluşturulması, içindeki html ve java dosyalarının kodlarının yazılması.

2. OZAN TEZCAN

2.a. Veritabanı tasarımları

2.b. Sorumlu olduğu 3 nesne için veritabanında "*Announcement*, *Comment* ve *GroupVenue*" adlı tabloların oluşturulması.

2.c. CSS (*style.css*) dosyasının oluşturulması.

2.d. *Application.java*, *HomePage.java*, *HomePage.html*, *HeaderPanel.java*, *HeaderPanel.html* dosyalarına sorumlu olduğu sınıflarla ilgili içeriklerin ve linklerin eklenmesi.

2.e. Sorumlu olduğu *Announcement* nesnesi için "*Announcements*" paketinin oluşturulması, içindeki html ve java dosyalarının kodlarının yazılması.

2.f. Sorumlu olduğu *Comment* nesnesi için "*Comments*" paketinin oluşturulması, içindeki html ve java dosyalarının kodlarının yazılması.

2.g. Sorumlu olduğu *GroupVenue* nesnesi için "*GroupVenues*" paketinin oluşturulması, içindeki html ve java dosyalarının kodlarının yazılması.

3. ECE ÇALIKUŞ

3.a. Veritabanı tasarımları

3.b. Sorumlu olduğu 3 nesne için veritabanında "*Users*, *UserCategory* ve *Ticket*" adlı tabloların oluşturulması.

3.c. *Application.java*, *HomePage.java*, *HomePage.html*, *HeaderPanel.java*, *HeaderPanel.html* dosyalarına sorumlu olduğu sınıflarla ilgili içeriklerin ve linklerin eklenmesi.

3.d. Sorumlu olduğu *Users* nesnesi için "*Users*" paketinin oluşturulması, içindeki html ve java dosyalarının kodlarının yazılması.

3.e. Sorumlu olduğu *UserCategory* nesnesi için "*UsersCategory*" paketinin oluşturulması, içindeki html ve java dosyalarının kodlarının yazılması.

3.f. Sorumlu olduğu *Ticket* nesnesi için "*Tickets*" paketinin oluşturulması, içindeki html ve java dosyalarının kodlarının yazılması.

4. İSMAİL CEM BAKIR

4.a. Veritabanı tasarımı

4.b. Sorumlu olduğu 3 nesne için veritabanında "*InHistory*, *TypeInHistory* ve *Venue*" adlı tabloların oluşturulması.

4.c. Application.java, HomePage.java, HomePage.html, HeaderPanel.java, HeaderPanel.html dosyalarına sorumlu olduğu sınıflarla ilgili içeriklerin ve linklerin eklenmesi.

4.d. Sorumlu olduğu InHistory nesnesi için "*InHistory*" paketinin oluşturulması, içindeki html ve java dosyalarının kodlarının yazılması.

4.e. Sorumlu olduğu TypeInHistory nesnesi için "*TypeInHistory*" paketinin oluşturulması, içindeki html ve java dosyalarının kodlarının yazılması.

4.f. Sorumlu olduğu Venue nesnesi için "*Venues*" paketinin oluşturulması, içindeki html ve java dosyalarının kodlarının yazılması.

5. ANIL KOCABIYIK

5.a. Veritabanı tasarımı

5.b. Sorumlu olduğu 3 nesne için veritabanında "*City*, *News* ve *TypeNews*" adlı tabloların oluşturulması.

5.c. Application.java, HomePage.java, HomePage.html, HeaderPanel.java, HeaderPanel.html dosyalarına sorumlu olduğu sınıflarla ilgili içeriklerin ve linklerin eklenmesi.

5.d. Sorumlu olduğu City nesnesi için "*Cities*" paketinin oluşturulması, içindeki html ve java dosyalarının kodlarının yazılması.

5.e. Sorumlu olduğu News nesnesi için "*News*" paketinin oluşturulması, içindeki html ve java dosyalarının kodlarının yazılması.

5.f. Sorumlu olduğu TypeNews nesnesi için "*TypeNews*" paketinin oluşturulması, içindeki html ve java dosyalarının kodlarının yazılması.

Chapter 2

Kurulum

Projenin kodları ve projede kullanılmış diğer dökümanların içeren proje ile aynı adlı klasör (BiletAl) Apache Framework yüklü NetBeans IDE'nin "Open Project" menüsü kullanılarak açılır.

Açılan projeye daha öncesinden özellikler kısmına kütüphane olarak *sqlite-jdbc-3.7.2.jar* dosyası eklenmiştir.

SQLite sunucusuna ve kullanılan bilgisayarın kullanıcı dosya dizininde (Ubuntu için /home/burak/ gibi) bulunan "bilet.db" adlı veritabanı dosyasına bağlanmamızı sağlayan kod bloğu Application.java dosyası içerisindeştir.

Application.java dosyası içerisinde çağırılan kod bloğu şu şekildedir;

```
public Application() throws SQLException {  
  
    try {  
        Class.forName("org.sqlite.JDBC");  
    } catch (ClassNotFoundException ex) {  
        throw new UnsupportedOperationException(ex.getMessage());  
    }  
  
    try {  
        String homeDir = System.getProperty("user.home");  
        String dbFileName = homeDir + File.separator + "bilet.db";  
        String jdbcURL = "jdbc:sqlite:" + dbFileName;  
        this.db = DriverManager.getConnection(jdbcURL);  
    } catch (SQLException ex) {  
        throw new UnsupportedOperationException(ex.getMessage());  
    }  
    ...  
}
```

Gösterilen kod bloğu içerisindeki ilk try-catch bloğu SQLite sunucusuna bağlanmamızı sağlar.

İkinci try-catch bloğu ile içerisinde tablolarımız bulunan "bilet.db" veritabanı dosyasına bağlanır. Ve böylelikle program çalıştırıl-maya hazır olur.

Eğer projenin çalıştırıldığı bilgisayarda kullanıcı dosya dizininde (Ubuntu için /home/burak gibi) "bilet.db" adlı veritabanı dosyası yok ise; proje çalıştırıldıkten sonra "bilet.db" adlı veritabanı dosyası, içinde tablo içermeden, kullanıcı dosya dizininde oluşturulur.

Herhangi bir SQLite yönetim programı ile "bilet.db" adlı veritabanı dosyası açılarak gerekli tablolar oluşturulur. Biz Mozilla Firefox eklentisi olan ve kolayca yüklenip çalıştırılabilen, SQLite Manager programını kullandık.

Tabloların oluşturulması için gerekli SQL kodları aşağıdaki gibidir;

```
CREATE TABLE announcement (  
    id INTEGER NOT PRIMARY KEY,  
    title VARCHAR(80),
```

```
author VARCHAR(80),
sum VARCHAR(255),
content VARCHAR(255),
image BLOB
);

CREATE TABLE comment(
    id INTEGER NOT NULL PRIMARY KEY,
    title VARCHAR(100),
    sum VARCHAR(255),
    author VARCHAR(100),
    anno_id INTEGER NOT NULL,
FOREIGN KEY (anno_id) REFERENCES announcement(id)
);

CREATE TABLE user_category(
    id INTEGER NOT NULL PRIMARY KEY,
    name VARCHAR(100)
);

CREATE TABLE users(
    id INTEGER NOT NULL PRIMARY KEY,
    name VARCHAR(100),
    birth VARCHAR(20),
    email VARCHAR(100),
    password VARCHAR(100),
    category_id INTEGER NOT NULL,
    FOREIGN KEY (category_id) REFERENCES user_category(id)
);

CREATE TABLE card(
    id INTEGER NOT NULL PRIMARY KEY,
    no VARCHAR(100),
    password VARCHAR(100),
    user_id INTEGER NOT NULL,
FOREIGN KEY (user_id) REFERENCES users(id)
);

CREATE TABLE city(
    id INTEGER NOT NULL PRIMARY KEY,
    name VARCHAR(100),
    maplink VARCHAR(255),
    image BLOB
);

CREATE TABLE group_venue(
    id INTEGER NOT NULL PRIMARY KEY,
    name VARCHAR(100)
);

CREATE TABLE venue(
    id INTEGER NOT NULL PRIMARY KEY,
    name VARCHAR(100),
    address VARCHAR(255),
    description VARCHAR(255),
    maplink VARCHAR(255),
    group_id INTEGER NOT NULL,
    city_id INTEGER NOT NULL,
    image BLOB,
    FOREIGN KEY (city_id) REFERENCES city(id),
    FOREIGN KEY (group_id) REFERENCES group_venue(id)
);
```

```
CREATE TABLE group_category(
    id INTEGER NOT NULL PRIMARY KEY,
    name VARCHAR(100)
);

CREATE TABLE event_category(
    id INTEGER NOT NULL PRIMARY KEY,
    name VARCHAR(100),
    group_id INTEGER NOT NULL,
    FOREIGN KEY (group_id) REFERENCES group_category(id)
);

CREATE TABLE event(
    id INTEGER NOT NULL PRIMARY KEY,
    title VARCHAR(100),
    date VARCHAR(20),
    time TIME,
    description VARCHAR(255),
    category_id INTEGER NOT NULL,
    venue_id INTEGER NOT NULL,
    price INTEGER,
    quato INTEGER,
    image BLOB,
    reached INTEGER DEFAULT 0,
    FOREIGN KEY (category_id) REFERENCES event_category(id),
    FOREIGN KEY (venue_id) REFERENCES venue(id)
);

CREATE TABLE ticket(
    id INTEGER NOT NULL PRIMARY KEY,
    event_id INTEGER NOT NULL,
    user_id INTEGER NOT NULL,
    item INTEGER NOT NULL,
    FOREIGN KEY (event_id) REFERENCES event(id),
    FOREIGN KEY (user_id) REFERENCES users(id)
);

CREATE TABLE typeinhistory(
    id INTEGER NOT NULL PRIMARY KEY,
    name VARCHAR(80)
);

CREATE TABLE inhistory(
    id INTEGER NOT NULL PRIMARY KEY,
    title VARCHAR(100),
    description VARCHAR(255),
    date VARCHAR(10),
    yr VARCHAR(10),
    image BLOB,
    type_id INTEGER NOT NULL,
    FOREIGN KEY (type_id) REFERENCES typeinhistory(id)
);

CREATE TABLE typenews(
    id INTEGER NOT NULL PRIMARY KEY,
    name VARCHAR(100)
);

CREATE TABLE news(
    id INTEGER NOT NULL PRIMARY KEY,
    title VARCHAR(80),
```

```
sum VARCHAR(255),  
content VARCHAR(255),  
date VARCHAR(20),  
author VARCHAR(80),  
image BLOB,  
type_id INTEGER NOT NULL,  
FOREIGN KEY (type_id) REFERENCES typenews(id)  
)
```

Chapter 3

Kullanım Kılavuzu

Not: Hazırlanmış olan raporun Kullanım Kılavuzu kısmında kendi sorumlu olduğum nesnelere ve sayfalara ağırlık vereceğim. HomePage, HeaderPanel, Events, CategoryEvents, GroupCategories, Cards bunların en önemlileridir.

Online Bilet Satın Alma Servisi, "BiletAI" projesinin kullanılmasını ve işlemesini sağlayan sayfa, panel ve formlar aşağıdaki bölümlerden oluşmaktadır.

AnaSayfa: Her tipten kullanıcının gezinebileceği bir sayfadır. Sol üst tarafta üye olunabilecek ve üye girişi yapılabilecek "Sign Up-Log In" formu, orta üstte sadece yöneticilerin görebildiği "Admin Panel", sağ üstte yönetici ve organizatörlerin görebildiği "Organizer Panel" bulunmaktadır. Bu form ve panellerin hemen altında etkinlik, mekan, kategoriler gibi site içeriğine dahil herşeyin listelenebileceği sayfalara link taşıyan "Header Panel" bulunmaktadır. "Header Panel" bloğunun altında anasayfanın gövdesi bulunmaktadır. Bu gövdede kategori, mekan ve şeirlere göre etkinlik araması yapan, bulduğu etkinlikleri listeleyen bir arama formu bulunmaktadır. Bu formun hemen sağında tarihte bugün adlı bir köşe vardır ve içeriği listelenmektedir. Bu iki formun altında önemli duyurular bulunmaktadır.

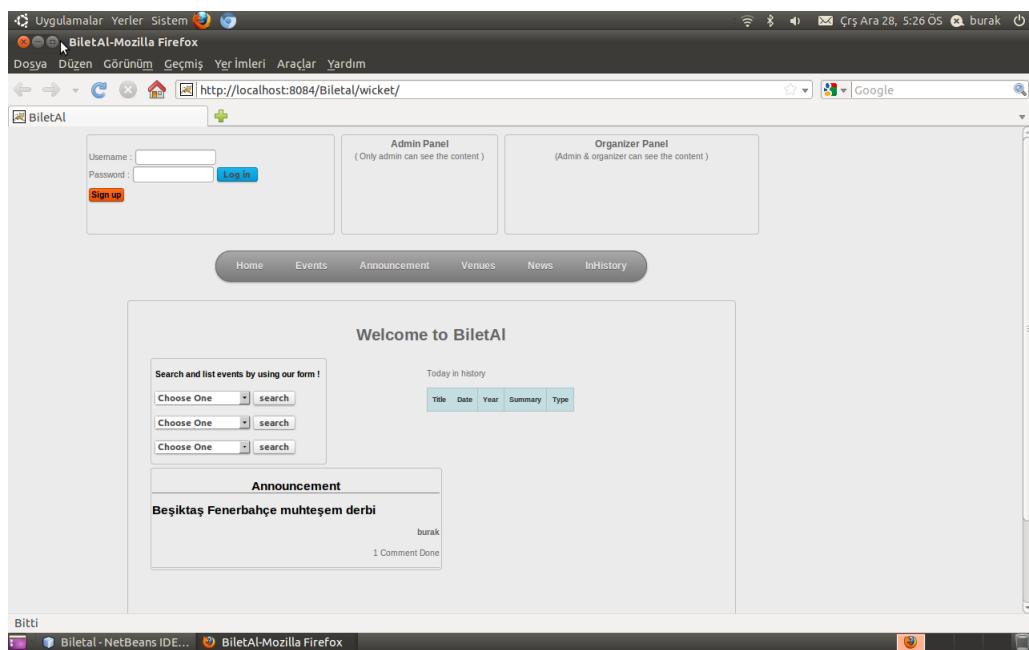


Figure 3.1: AnaSayfada Ziyaretçi tipinden kullanıcı için bir görüntü

Sign Up-Log In Form: Bilindiği üzere sistemimizde 4 tip kullanıcı vardır. Ziyaretçi için bu formda kayıt için bir link, log in için şifrenin ve kullanıcı adının yazıldığı bölgeler bulunur. Diğer 3 kullanıcı tipi için log in olunduktan sonra kredi kartlarının

listelendiği ve yeni kredi kartı oluşturulabilen MyCardPage için yine bu formda "MyCard Page" linki bulunur. Bu linkin kurucu fonksiyonu "session" çağrısu içindeki userId'ye göre kişiye ait kredi kartlarını göstererek şekilde çağrıılır.



Figure 3.2: Sign Up-Log In Formun Ziyaretçi için görünümü

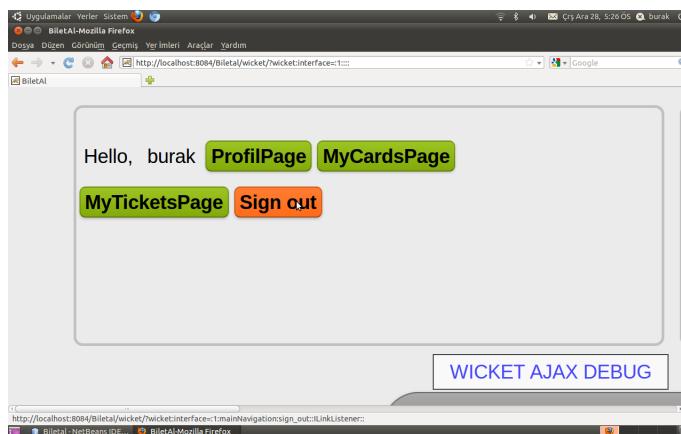


Figure 3.3: Sign Up-Log In Formun Ziyaretçi dışındaki üyeleri için görünümü

Admin Panel: AnaSayfa içerisinde görüntülenen, fakat HeaderPanele ait bu panel, uygun session userGroupId sorguları ve Wicket componentleri ile ancak yönetici tipinden bir kullanıcı log in olduğunda görülmektedir. Aksi takdirde boş görülmektedir.

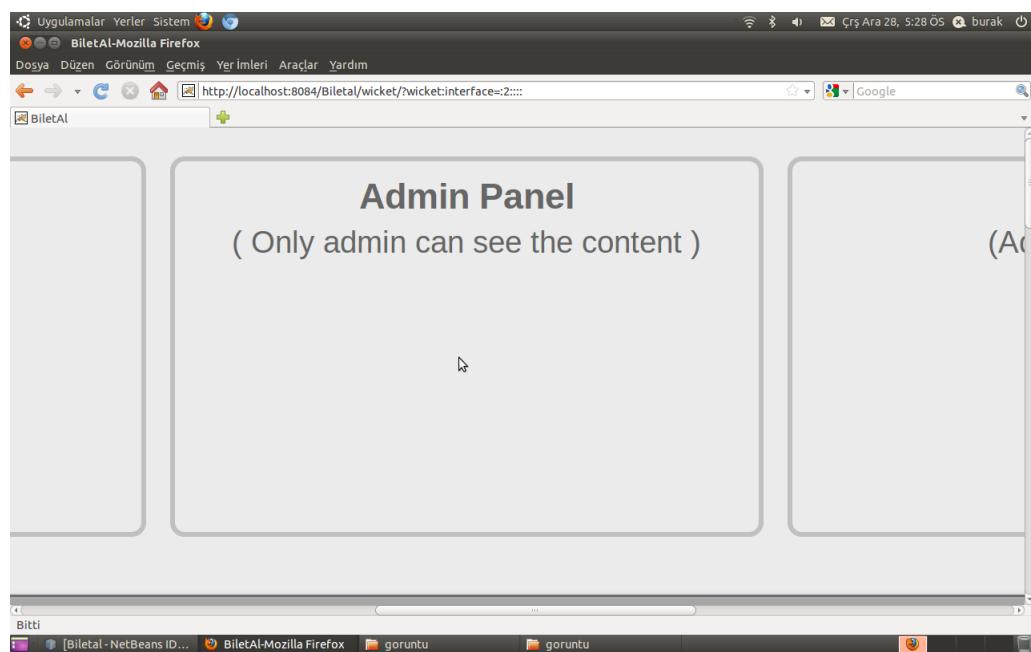


Figure 3.4: Admin Panelin Ziyaretçi için görünümü

Organizer Panel: AnaSayfa içerisinde görüntülenen, fakat HeaderPanel'e ait bu panel, uygun session userGroupId sorguları ve Wicket componentleri ile ancak yönetici ve organizatör tipinden bir kullanıcı log in olduğunda görülmektedir. Aksi takdirde boş görülmektedir. Bu paneldeki linkler sayesinde kullanıcı siteye etkinlik, mekan, şehir, kategori, haber, duyuru gibi içerikleri ekleyebilir.

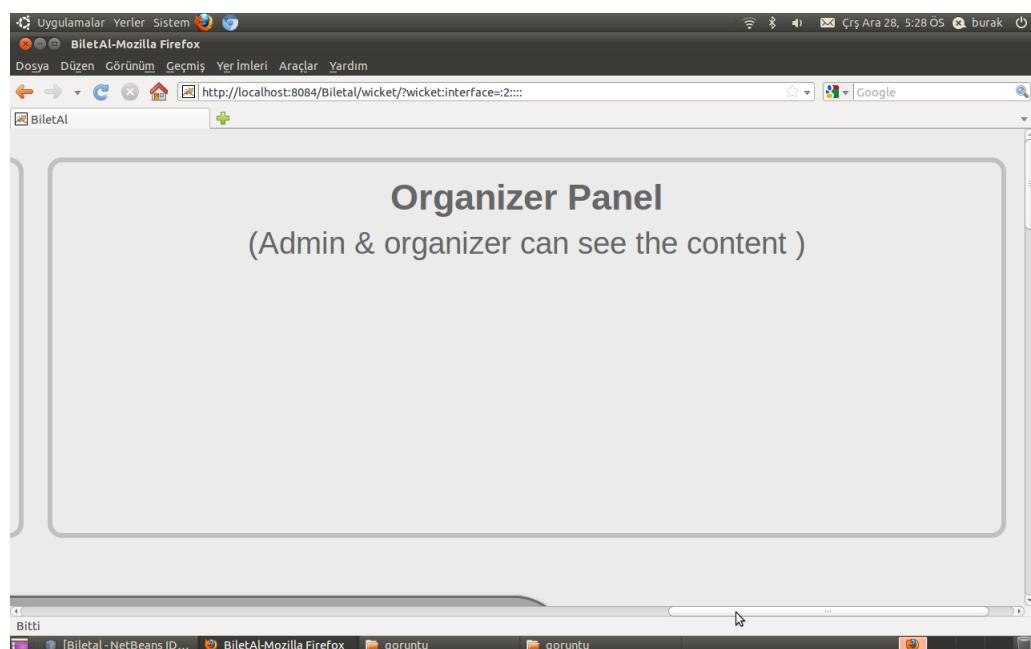


Figure 3.5: Organizer Panelin Ziyaretçi için görünümü

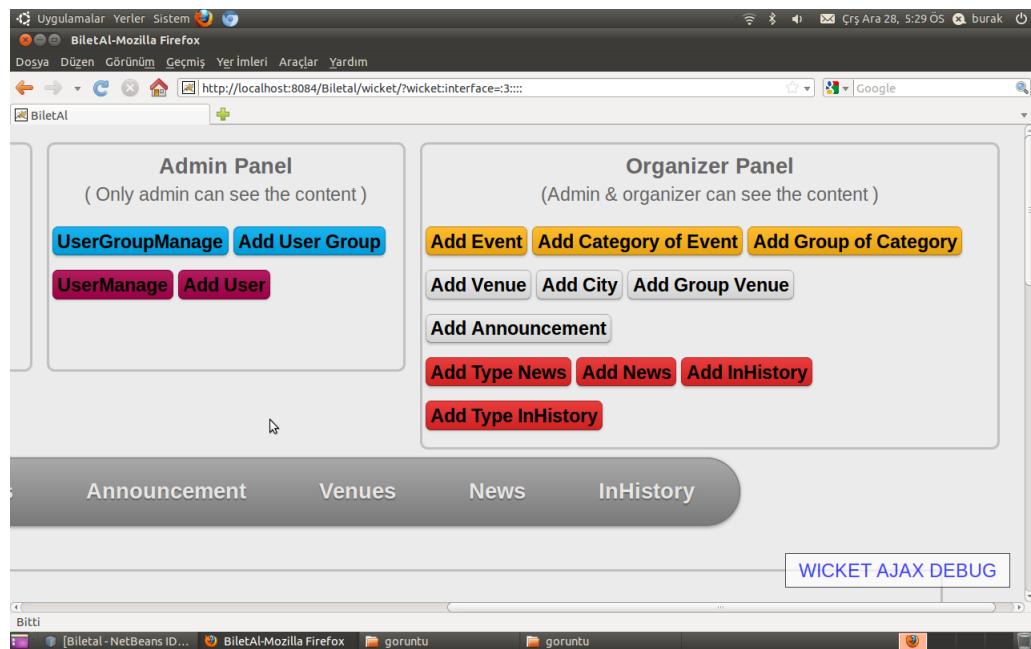


Figure 3.6: Admin ve Organizer Panelin Yönetici üyeleri için görünümü

Header Panel: AnaSayfa içerisinde görüntülenen bu panelde, etkinlik, mekan, kategoriler, haberler, tarihte yaşananlar gibi site içeriğine dahil olan herşeyin listelenebileceği sayfaların linkleri bulunmaktadır. Ayrıca bu linklerin üzerinde gelindiğinde bunların tümünün tipleri için alt tablar açılmaktadır.

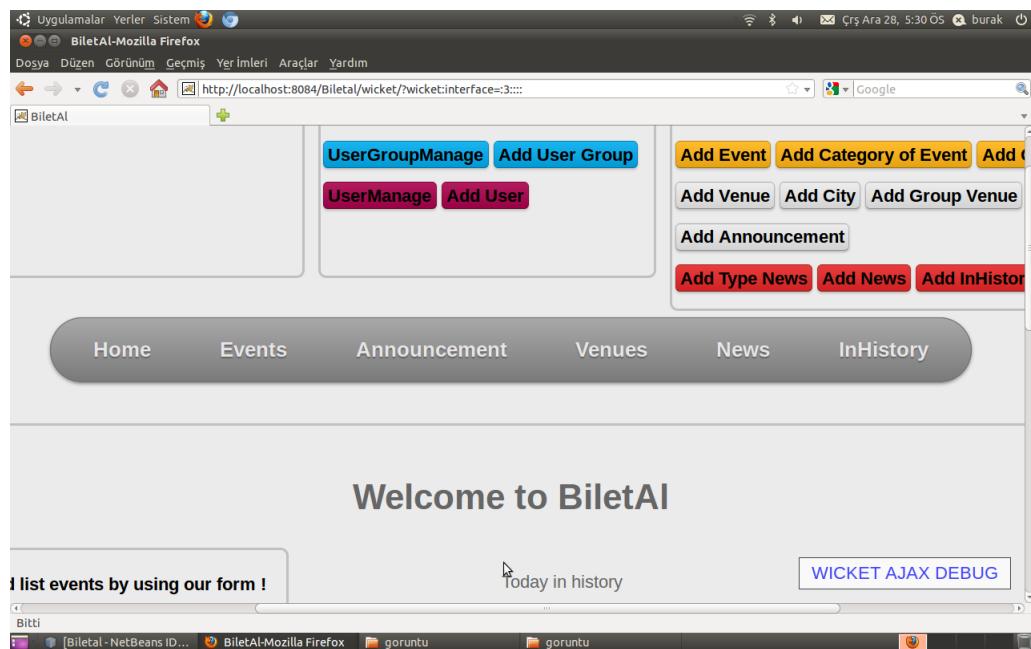


Figure 3.7: Header Panel görünümü

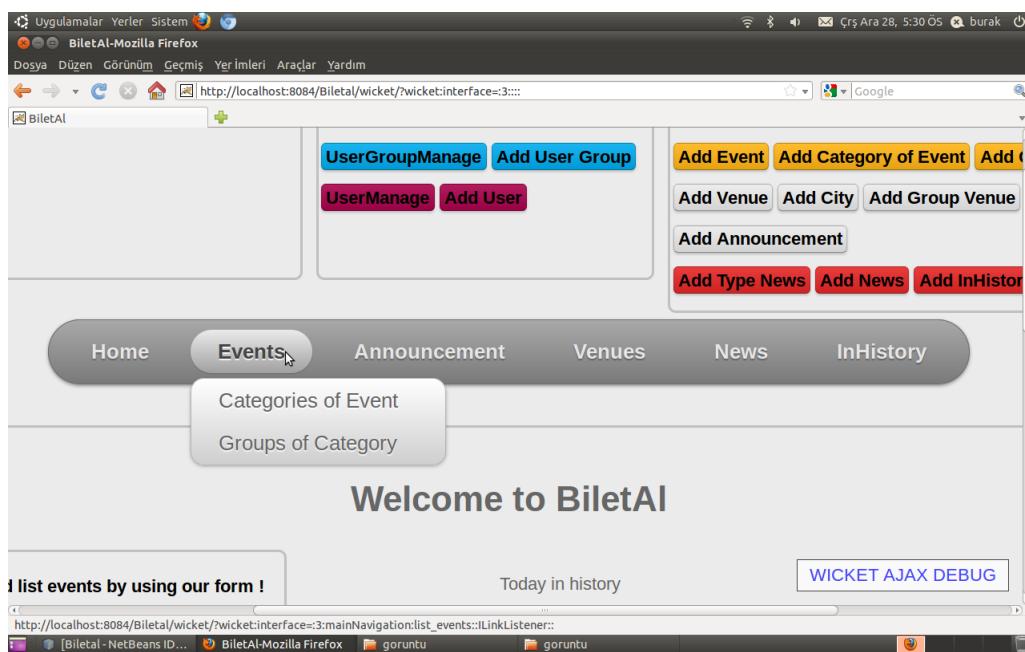


Figure 3.8: Header Panelde alt tabların görünümü

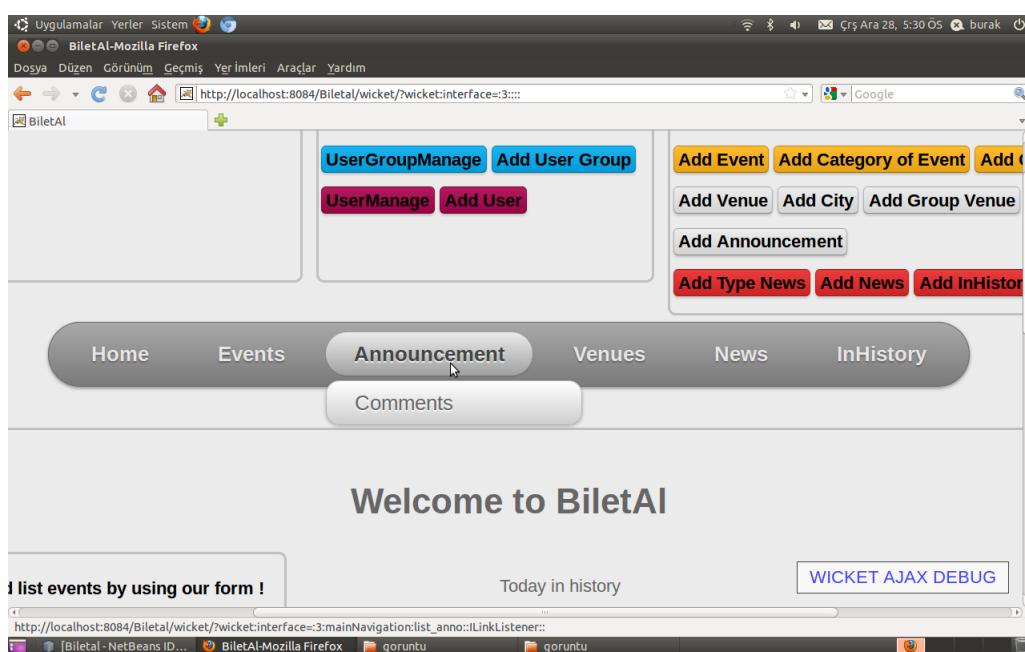


Figure 3.9: Header Panelde alt tabların görünümü

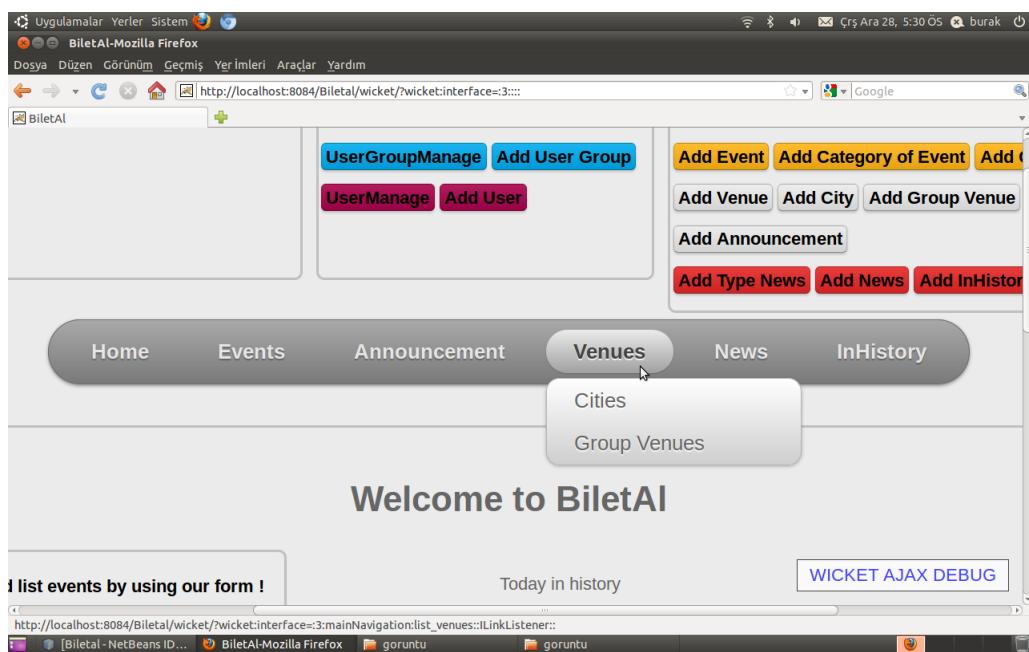


Figure 3.10: Header Panelde alt tabların görünümü

Bu girişle beraber detaylı bir kullanım için devam edelim. Bunun için AnaSayfa üzerinde devam edelim.

"Header Panel" bloğunun altında anasayfanın gövdesi bulunmaktadır. Bu gövdede kategori, mekan ve şehirlere göre etkinlik araması yapan, bulunduğu etkinlikleri listeleyen bir arama formu bulunmaktadır. Bu formun hemen sağında tarihte bugün adlı bir köşe vardır ve içeriği listelenmektedir. Bu iki formun altında önemli duyurular bulunmaktadır.

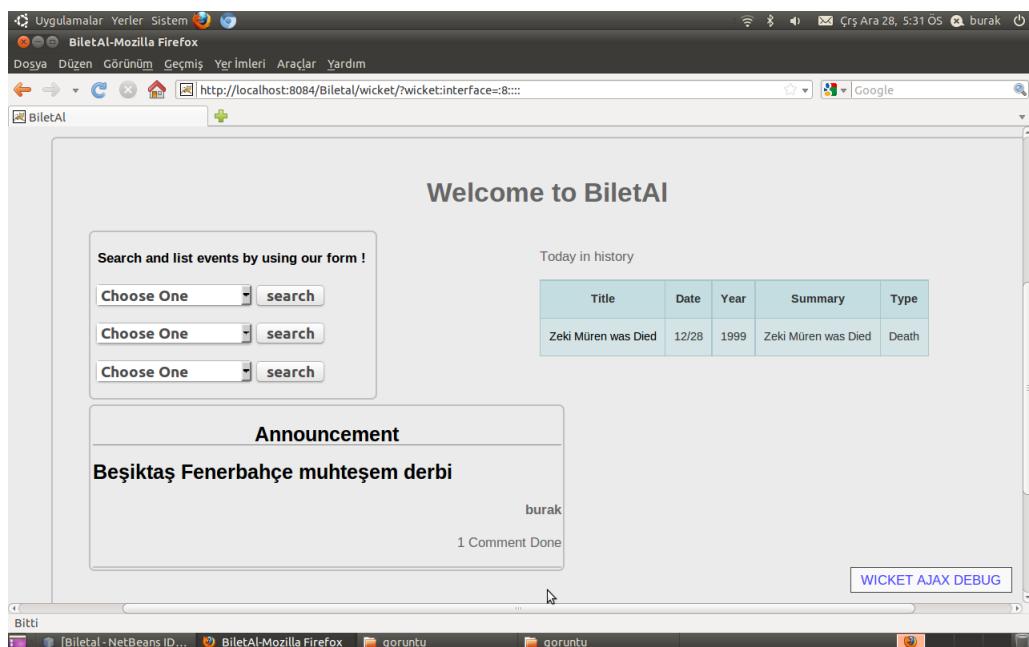


Figure 3.11: AnaSayfada arama motoru, duyurular ve tarihte bugün kölesi

Arama motoru kullanıcı tipi sınırlaması olmaksızın, kategori, şehir, ve mekana göre etkinlileri arayıp listeleyen kullanışlı bir formdur. Gelen etkinlik listesinde etkinliklere dair bilgiler (mekan, tarih, kontenjan, fiyat gibi) mevcuttur. Bilet satın alma,

daha detaylı görüntüleme ya da güncelleme işlemi için listelenen elemanların "title" kısmında mevcut olan linki tıklamak gerekmektedir. Bununla beraber etkinliği detaylı görüntülediğimizde yapılan kullanıcı girişinin tipi göre "edit" ve "buy" linkleri açılır. Bilindiği üzere sadece yönetici güncelleme yetkisine sahipken, ziyaretçi dışındaki kullanıcı tiplerinin tamamı bilet satın alabilmektedir.

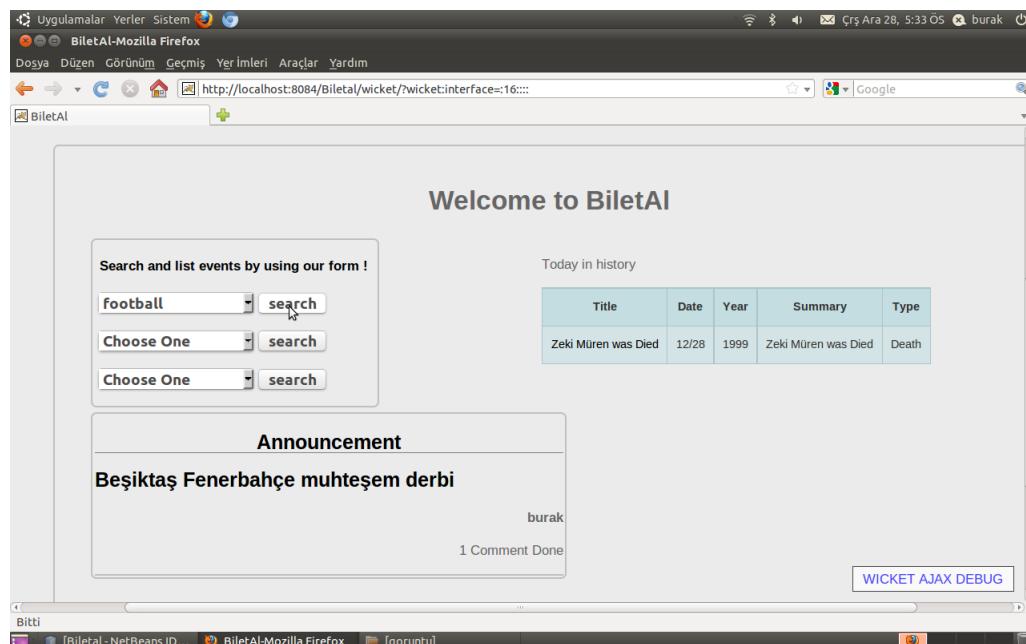


Figure 3.12: Arama motorunda kategoriye göre etkinlik arama

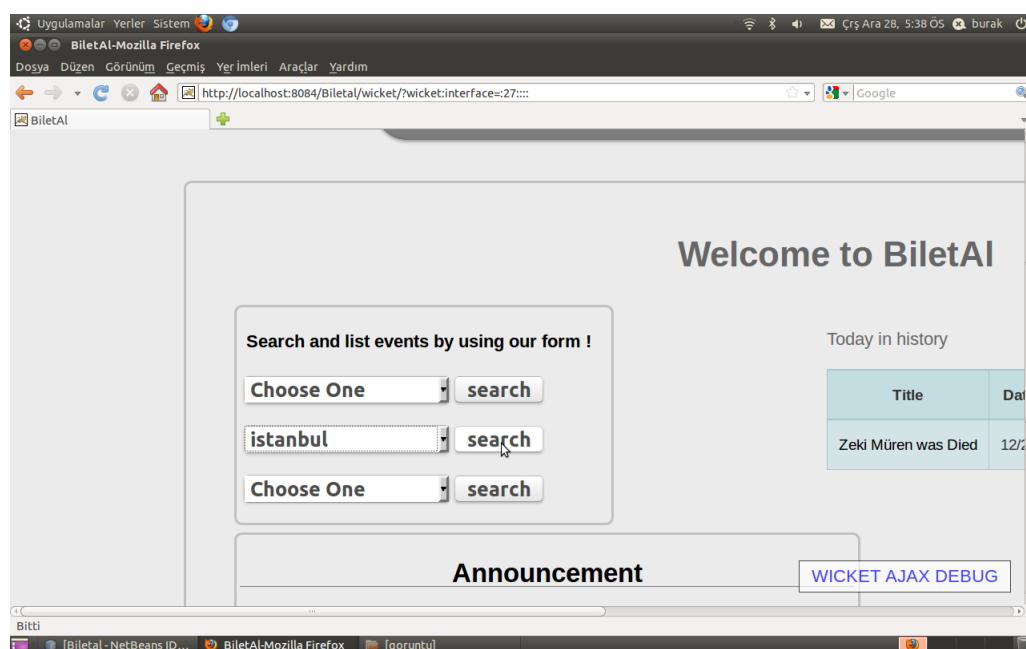


Figure 3.13: Arama motorunda şehire göre etkinlik arama

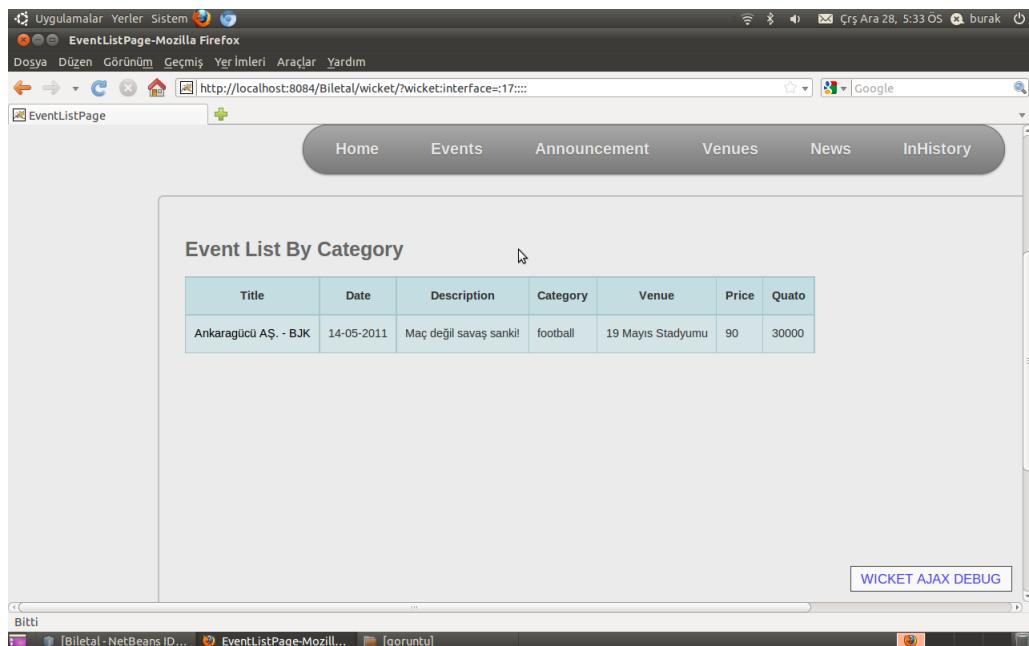


Figure 3.14: Arama motorunda kategoriye göre etkinlik arama sonucu

Etkinlikler: Etkinlikler sayfası arama motoru haricinde de görüntülenebilir. HeaderPanel'de Events adlı linkle etkinlikleri görüntüleyebiliriz. Tüm kullanıcılar için bu sayfa görüntülenebilmektedir. Açılan sayfada, etkinlikler bilgileri ile beraber düzenli bir biçimde görüntülenebilir. Ayrıca yönetici girişi yapılmışsa bu etkinliklerden "checkbox" ile seçilenler silinebilir. Yönetici girişi yapılmamışsa silinemez. Ayrıca etkinlikler için "title" üzerinde olan linkle beraber "EventDisplayPage" sayfasına yönlendirilir. Bu sayfada etkinlik detayları yer alır. Eğer ziyaretçi dışındaki kullanıcı tipleri ile giriş yapılmışsa "buy" linki açıktır. Etkinlik için her bilet alısta bir adet bilet satın alınabilir. Etkinliğin belli sayıda kontenjanı vardır ve ayrı bir veri olarak o ana kadar satın alınan bilet sayısı tutulur. Kontenjan dolumu bu şekilde gösterilir. Kontenjan dolmuşsa bilet satın alınamaz. Kullanıcı tipi yönetici ise etkinlik için "edit" linki de açılır. Böylelikle etkinlikler güncellenebilir. Etkinlikler listesinin hemen altında "Ajax" ile tasarlanmış bir arama formu vardır. Burada yazılan katarları içeren etkinlikler hemen bu form altında görünür. Üzerine tıklandığında form içine yazılır. Altındaki "show" linki ile bu etkinliğin sayfası açılır. Organizatör ve yönetici tipinden kullanıcılar etkinlik eklemek için *Organizer Panel*' de bulunan "add event" butonunu kullanırlar. Açılan sayfada etkinlik eklenir.

Etkinlikler için ekran görüntüleri aşağıdaki gibidir;

EventList

Select	Title	Date	Description	Category	Venue	Price	Quato	Reach
<input type="checkbox"/>	İTÜ ÖĞRENCİ ŞENLİĞİ	03-05-2011	25. Geleneksel İTÜ öğrenci Şenliği Bedri Karafakioğlu anısına	college festival	Şeref Bey Stadi	0	10000	0
<input type="checkbox"/>	Şebnem Ferah Konseri	14-05-2011	Yüzyıllık Yalnızlık!	alternative music	Ghetto	200	525	0
<input type="checkbox"/>	Ankaragücü AŞ. - BJK	14-05-2011	Maç değil savaş sankıl!	football	19 Mayıs Stadyumu	90	30000	0
<input type="checkbox"/>	Neşet Ertaş Konseri	11-12-2011	Ahu Gözlerini Sevdigim Dilber	show	CRR Konser Salonu	120	600	1
<input type="checkbox"/>	Benimle Delirir misin?	08-03-2012	"Her Yöne 90 dk" Komedi Oyununun Yapımcisından Yine İddialı Bir Komedi: Benimle Delirir misin? Komedi 2 Perde	theatre	ODTÜ KKM	30	120	0

Delete

Search event via name

Show

http://localhost:8084/Biletal/wicket/?wicket:interface=:18:event_list_form:selected_events:event_list:0:event_link::ILinkListener::

Figure 3.15: Etkinlik listeleme sayfası

Select	Title	Date	Description	Category	Venue	Price	Quato	Reach
<input type="checkbox"/>	İTÜ ÖĞRENCİ ŞENLİĞİ	03-05-2011	25. Geleneksel İTÜ öğrenci Şenliği Bedri Karafakioğlu anısına	college festival	Şeref Bey Stadi	0	10000	0
<input type="checkbox"/>	Şebnem Ferah Konseri	14-05-2011	Yüzyıllık Yalnızlık!	alternative music	Ghetto	200	525	0
<input type="checkbox"/>	Ankaragücü AS. - BJK	14-05-2011	Maç değil savaş sankıl!	football	19 Mayıs Stadyumu	90	30000	0
<input type="checkbox"/>	Neşet Ertaş Konseri	11-12-2011	Ahu Gözlerini Sevdigim Dilber	show	CRR Konser Salonu	120	600	1
<input checked="" type="checkbox"/>	Benimle Delirir misin?	08-03-2012	"Her Yöne 90 dk" Komedi Oyununun Yapımcisından Yine İddialı Bir Komedi: Benimle Delirir misin? Komedi 2 Perde	theatre	ODTÜ KKM	30	120	0

Delete

Search event via name

Show

Figure 3.16: Etkinlik listeleme sayfasında etkinlik silme

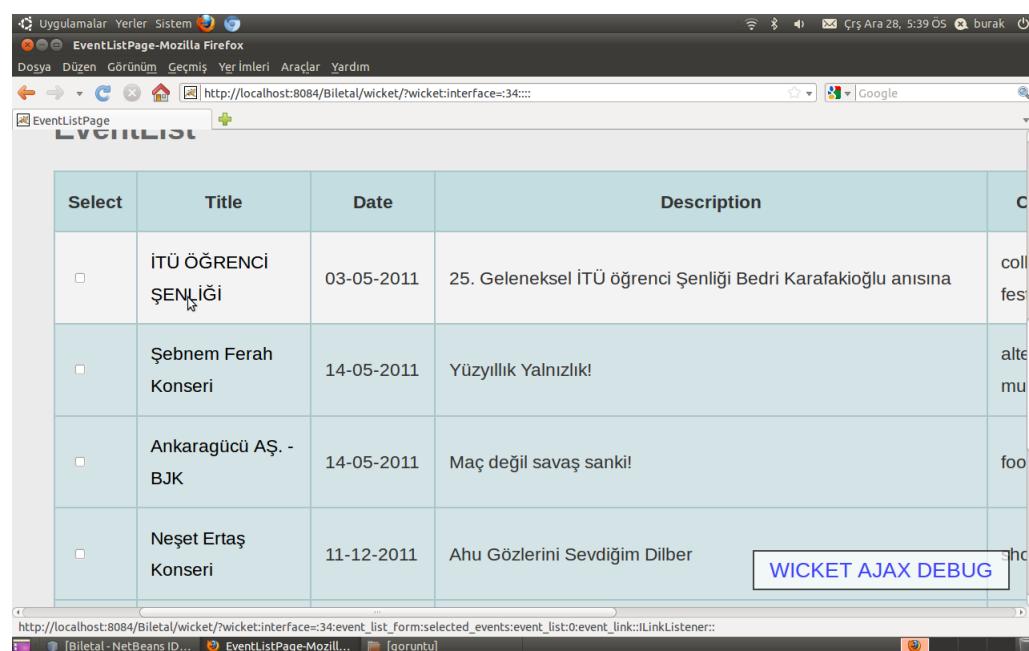


Figure 3.17: Etkinlik listesinde EventDisplayPage linki

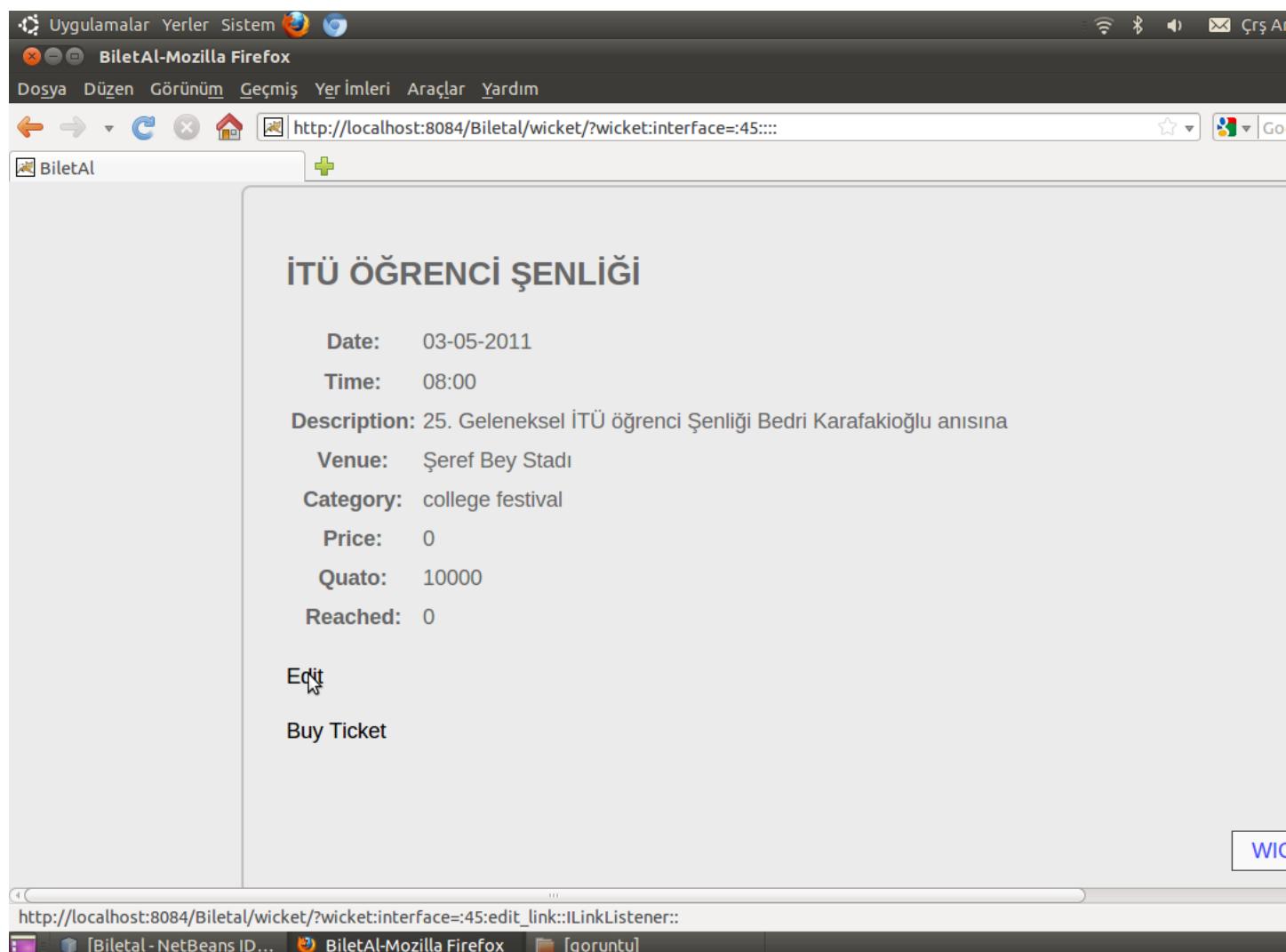


Figure 3.18: Yönetici için EventDisplayPage 1

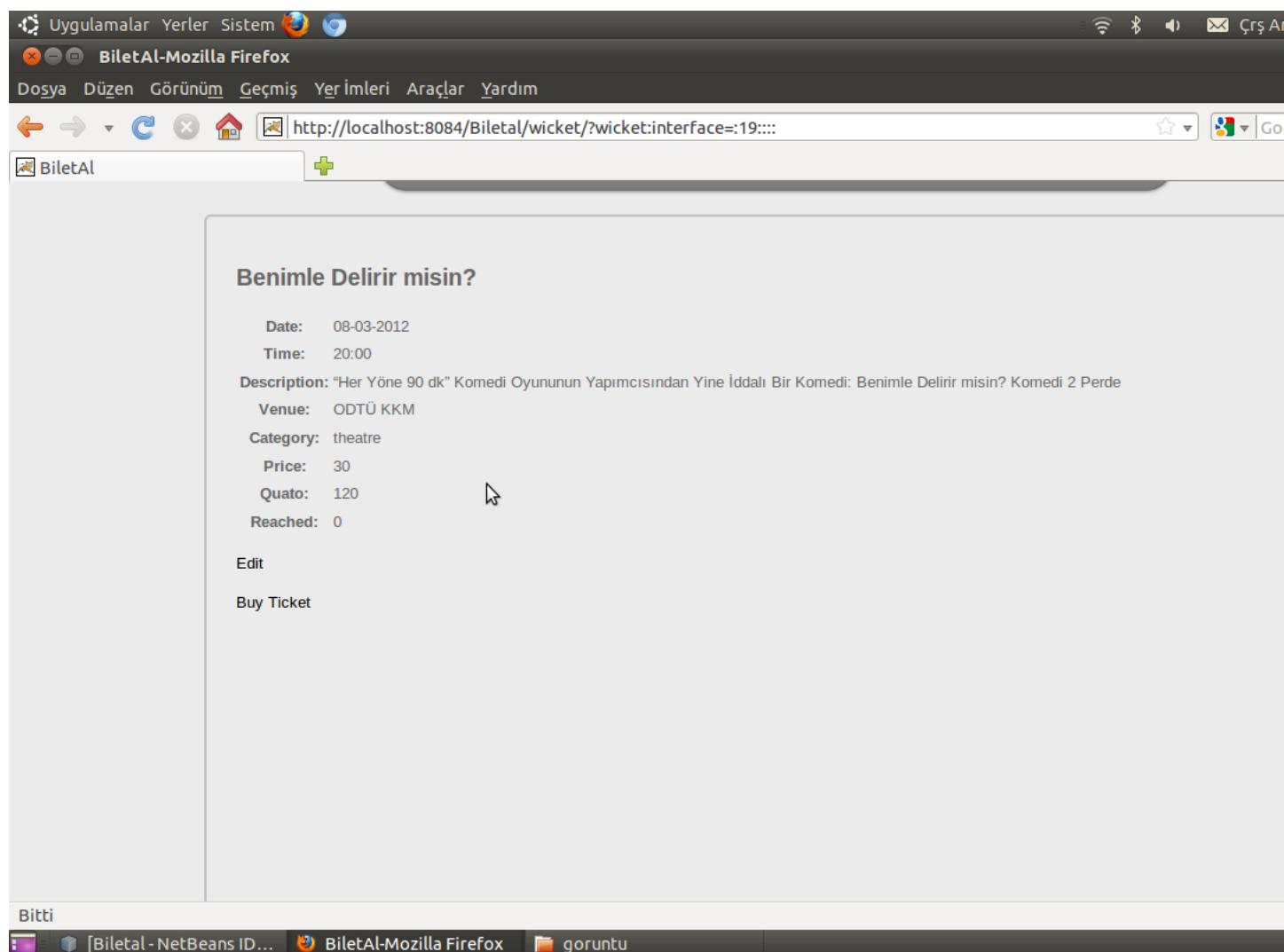


Figure 3.19: Yönetici için EventDisplayPage 2

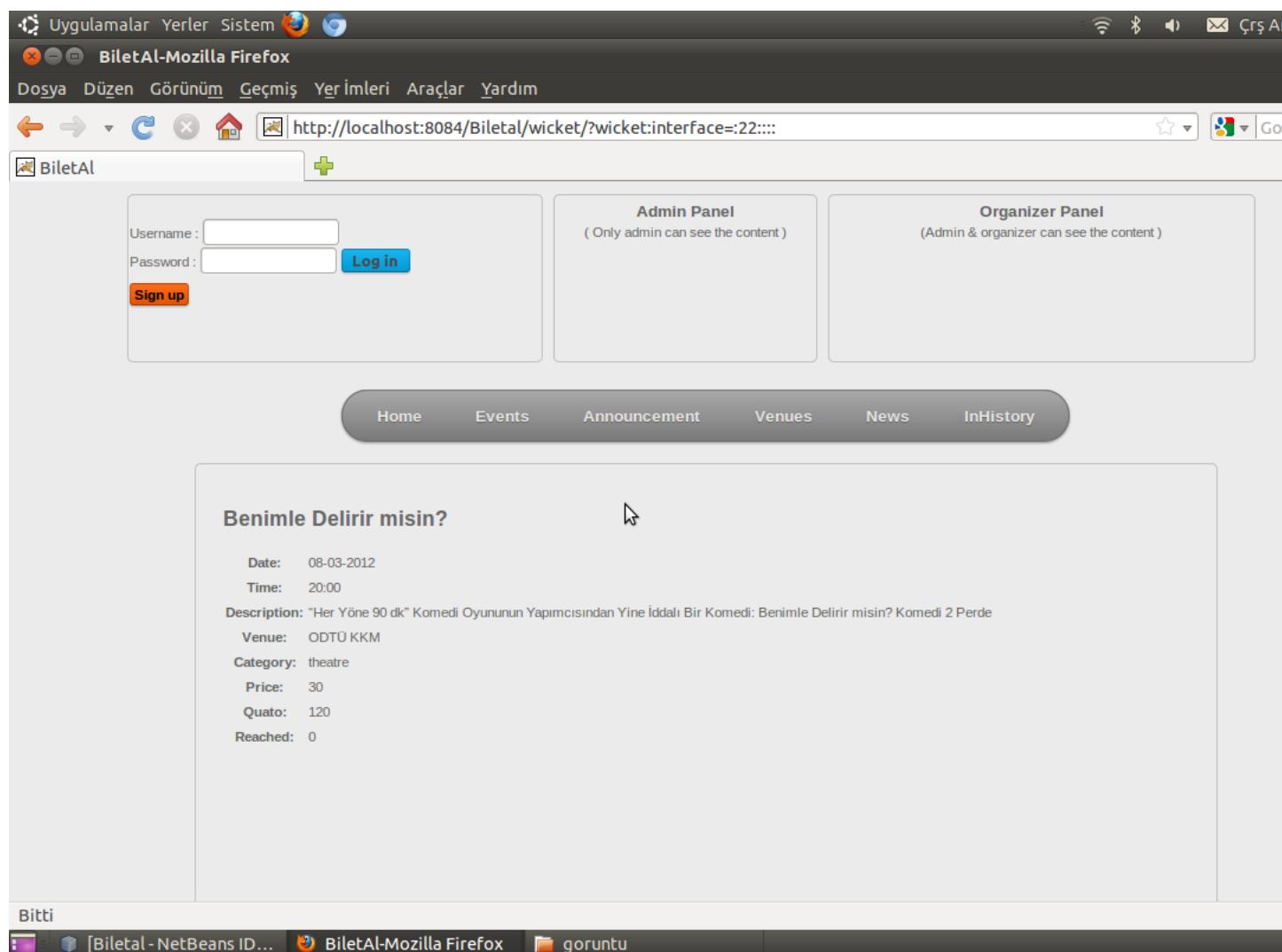


Figure 3.20: Ziyaretçi için EventDisplayPage

Uygulamalar Yerler Sistem BiletAl-Mozilla Firefox
Dosya Düzen Görünüm Geçmiş Yerimleri Araçlar Yardım
BiletAl http://localhost:8084/BiletAl/wicket/?wicket:interface=:43:::
WICKET AJAX DEBUG

Title: İTÜ ÖĞRENCİ ŞENLİĞİ
Date: 03-05-2
Time: 08:00
Price: 0
Quato: 10000
Category: Choose One
Venue: Choose One
Description
Save

Figure 3.21: Yönetici için EventEditPage

Uygulamalar Yerler Sistem EventListPage-Mozilla Firefox
Dosya Düzen Görünüm Geçmiş Yerimleri Araçlar Yardım
EventListPage http://localhost:8084/BiletAl/wicket/?wicket:interface=:18::::
WICKET AJAX DEBUG

Select	Title	Date	Description	Category	Venue	Price	Quato	Reached
<input type="checkbox"/>	ITU ÖĞRENCİ ŞENLİĞİ	03-05-2011	25. Geleneksel ITU öğrenci Şenliği Bedri Karatakoğlu anısına	college festival	Şeref Bey Stadi	0	10000	0
<input type="checkbox"/>	Şebnem Ferah Konseri	14-05-2011	Yüzyclopedia Yarışması!	alternative music	Ghetto	200	525	0
<input type="checkbox"/>	Ankaragücü AŞ - BJK	14-05-2011	Maç değil sevap sankıl	football	19 Mayıs Stadyumu	90	30000	0
<input type="checkbox"/>	Neyet Erteş Konseri	11-12-2011	Ahu Gözlerini Sevdigim Dilber	show	CRR Konser Salonu	120	600	1
<input type="checkbox"/>	Benimle Delir misin?	08-03-2012	"Her Yöne 90 dk" Komedy Oyununun Yapımcsından Yine İddalı Bir Komedy: Benimle Delir misin? Komedy 2 Perde	theatre	ODTÜ KKM	30	120	0

Delete
Search event via name
Benim
Show
• Benimle Delir misin?

Figure 3.22: Etkinlikler sayfasında Ajax kullanılarak hazırlanan arama formu

The screenshot shows a Firefox browser window titled "EventListPage-Mozilla Firefox". The URL is <http://localhost:8084/Biletal/wicket/?wicket:interface=:18:::>. The page displays a table titled "EventList" with columns: Select, Title, Date, Description, Category, Venue, Price, Quato, and Reached. There are five rows of event data. Below the table is a "Delete" button and a search input field containing "Benimle Delir misin?". A "Show" link is also present. At the bottom right of the page is a "WICKET AJAX DEBUG" button.

Select	Title	Date	Description	Category	Venue	Price	Quato	Reached
<input type="checkbox"/>	ITU ÖĞRENCİ ŞENLİĞİ	03-05-2011	25. Geleneksel ITU öğrenci Şenliği Bedri Karafaklıoğlu anısına	college festival	Seref Bey Stadi	0	10000	0
<input type="checkbox"/>	Şebnem Ferah Konseri	14-05-2011	Yüzeylik Yahuuzzik!	alternative music	Ghetto	200	525	0
<input type="checkbox"/>	Ankaragücü AS - BJK	14-05-2011	Maç değil savaş sanki!	football	19 Mayıs Stadyumu	90	30000	0
<input type="checkbox"/>	Neset Ertaş Konseri	11-12-2011	Ahu Gözlerini Sevdilim Diber	show	CRR Konser Salonu	120	600	1
<input type="checkbox"/>	Benimle Delir misin?	08-03-2012	"Her Yöne 90 dk" Komedy Oyununun Yapımcisından Yine İddialı Bir Komedy: Benimle Delir misin? Komedi 2 Perde	theatre	ODTU KKM	30	120	0

Search event via name:
Benimle Delir misin?
Show
WICKET AJAX DEBUG

Figure 3.23: Arama formunda etkinlik show linki

The screenshot shows a Firefox browser window titled "BiletAl-Mozilla Firefox". The URL is <http://localhost:8084/Biletal/wicket/?wicket:interface=:24:::>. The page features three main panels: "Hello, burak" with links to "ProfilPage" and "MyCardsPage", "Admin Panel" (Only admin can see the content) with "UserGroupManage" and "Add User Group" buttons, and "Organizer Panel" (Admin & organizer can see the content) with "Add Event", "Add Category of Event", "Add Group of Category", "Add Venue", "Add City", "Add Group Venue", "Add Announcement", "Add Type News", "Add News", "Add InHistory", and "Add Type InHistory" buttons. Below these panels is a form for adding an event with fields for Title, Date, Time, Price, Quato, Category, Venue, and Description, along with a "Save" button. At the bottom right of the page is a "WICKET AJAX DEBUG" button.

Hello, burak ProfilPage MyCardsPage
MyTicketsPage Sign out

Admin Panel
(Only admin can see the content)
UserGroupManage Add User Group
UserManage Add User

Organizer Panel
(Admin & organizer can see the content)
Add Event Add Category of Event Add Group of Category
Add Venue Add City Add Group Venue
Add Announcement
Add Type News Add News Add InHistory Add Type InHistory

Home Events Announcement Venues News InHistory

Title:
Date:
Time:
Price:
Quato:
Category: Choose One
Venue: Choose One
Description:
Save
WICKET AJAX DEBUG

Figure 3.24: Etkinlik ekleme butonu (add Event)

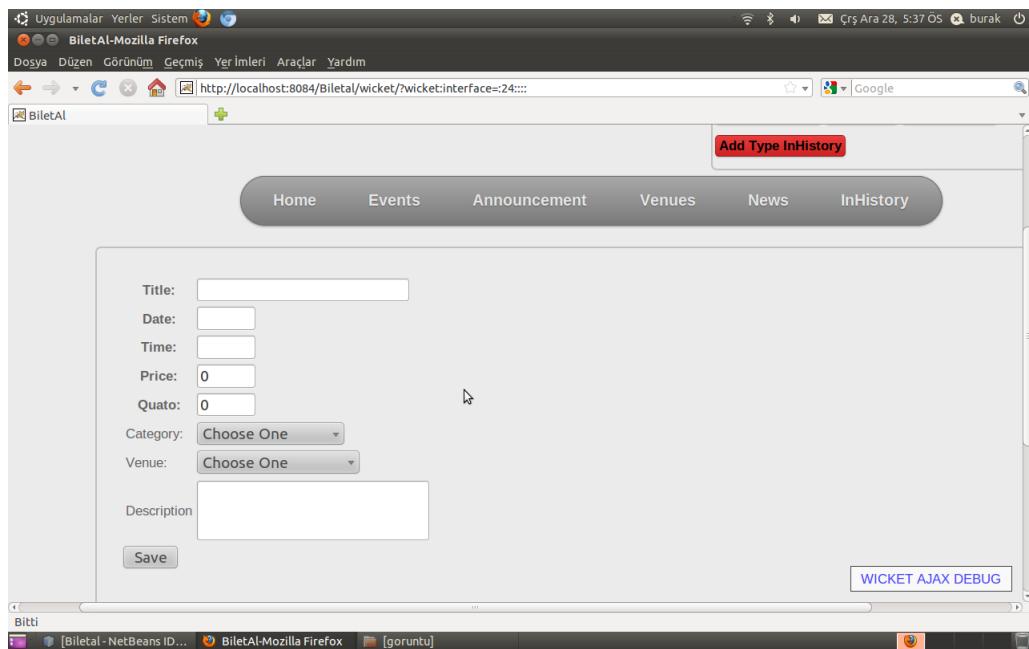


Figure 3.25: Etkinlik ekleme sayfası

Kategoriler: HeaderPanel'de "Categories of Event" adlı linkle kategorileri görüntüleyebiliriz. Tüm kullanıcılar için bu sayfa görüntülenebilmektedir. Açılan sayfada, kategoriler bilgileri ile beraber düzenli bir biçimde görüntülenebilir. Ayrıca yönetici giriş yapmışsa bu kategorilerden "checkbox" ile seçilenler silinebilir. Yönetici giriş yapılmamışsa silinmez. Ayrıca kategoriler için "name" üzerinde olan linkle beraber "CategoryDisplayPage" sayfasına yönlendirilir. Bu sayfada kategori detayları yer alır. Kullanıcı tipi yönetici ise kategoriler için "edit" linki de açılır. Böylelikle kategoriler güncellenebilir. Organizatör ve yönetici tipinden kullanıcılar kategori eklemek için *Organizer Panel*' de bulunan "add category of event" butonunu kullanırlar. Açılan sayfa Ajaxla oluşturulmuş bir Modal Window sayfasıdır. Bu pencere içerisinde kategori eklenir.

Kategoriler için ekran görüntüleri aşağıdaki gibidir;

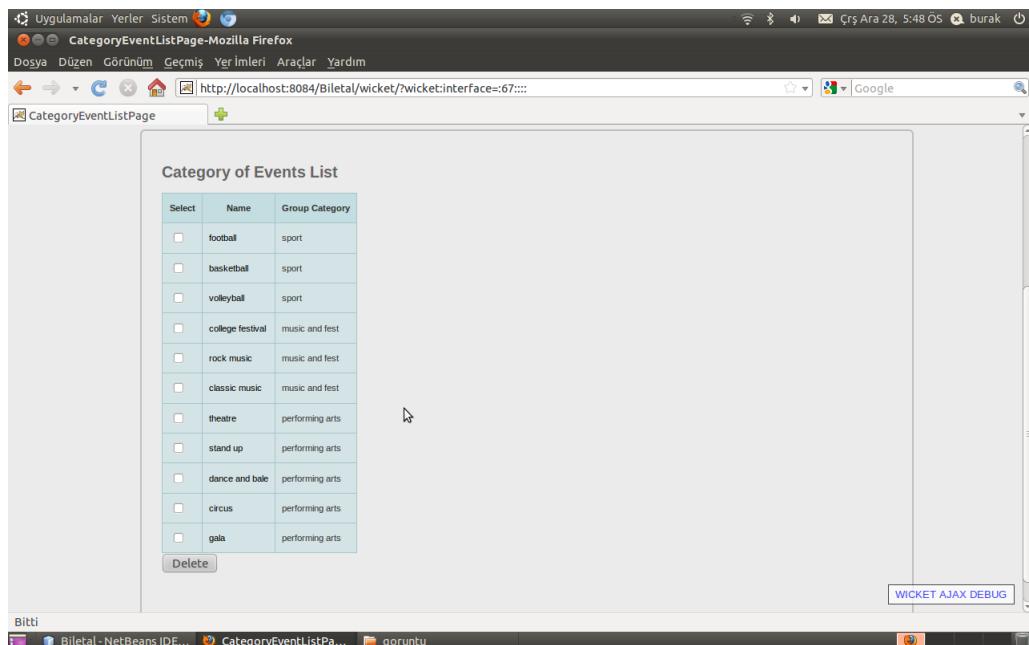


Figure 3.26: Kategori listeleme sayfası

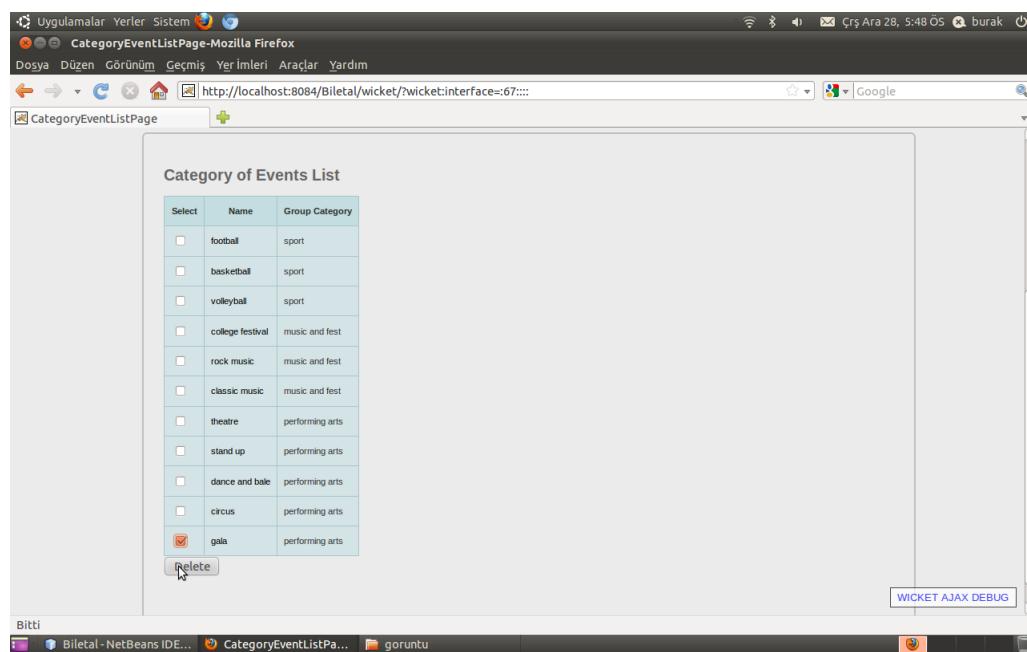


Figure 3.27: Kategori listeleme sayfasında yönetici için silme işlemi

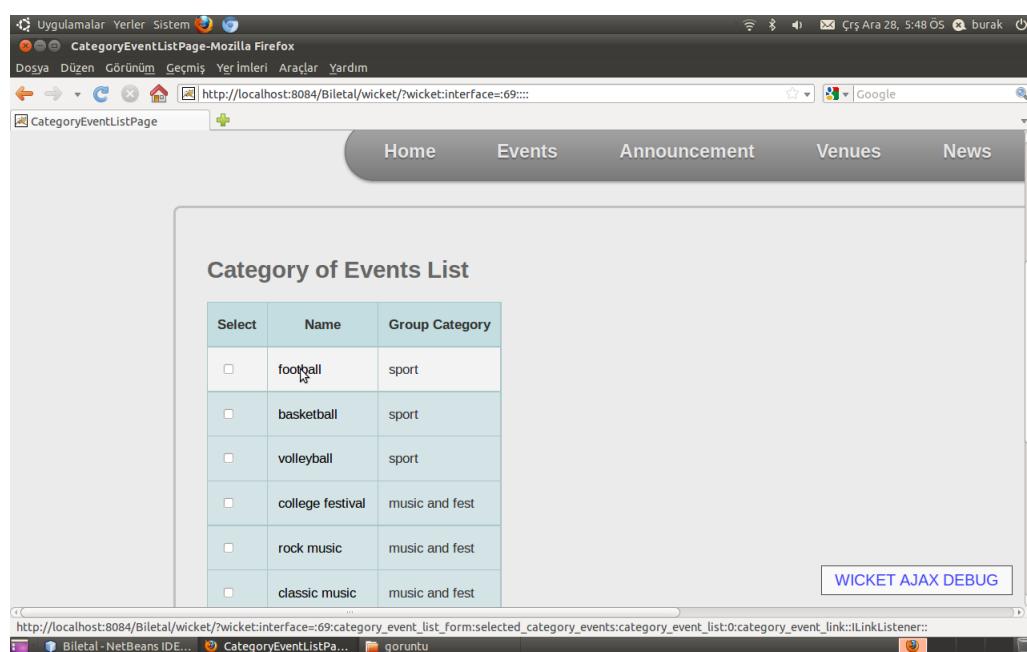


Figure 3.28: Kategori listeleme sayfasında CategoryDisplayPage için link

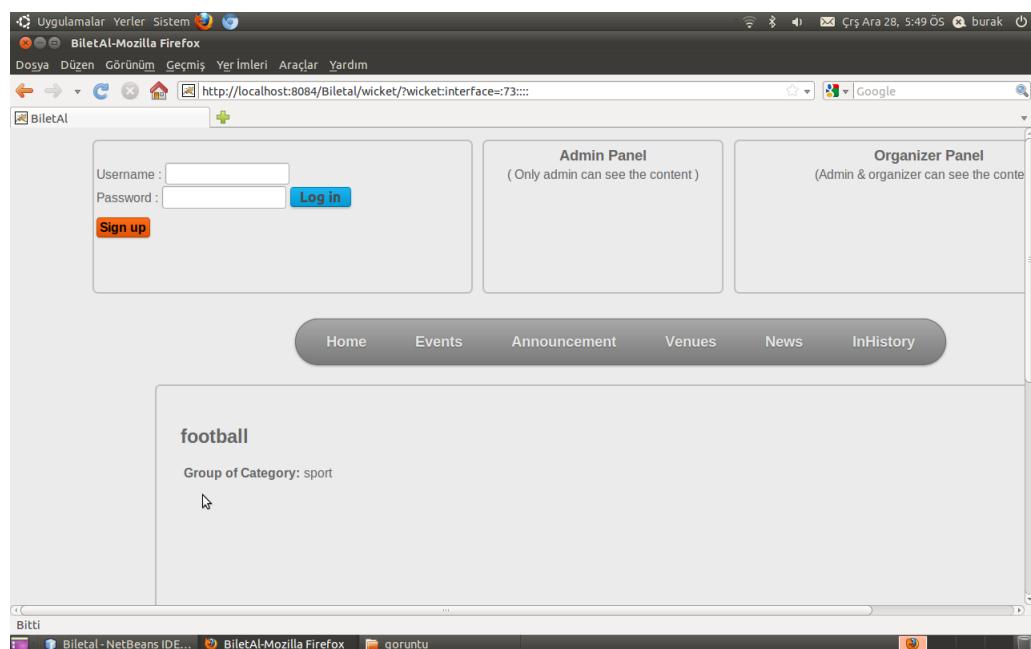


Figure 3.29: Ziyaretçi için CategoryDisplayPage sayfası

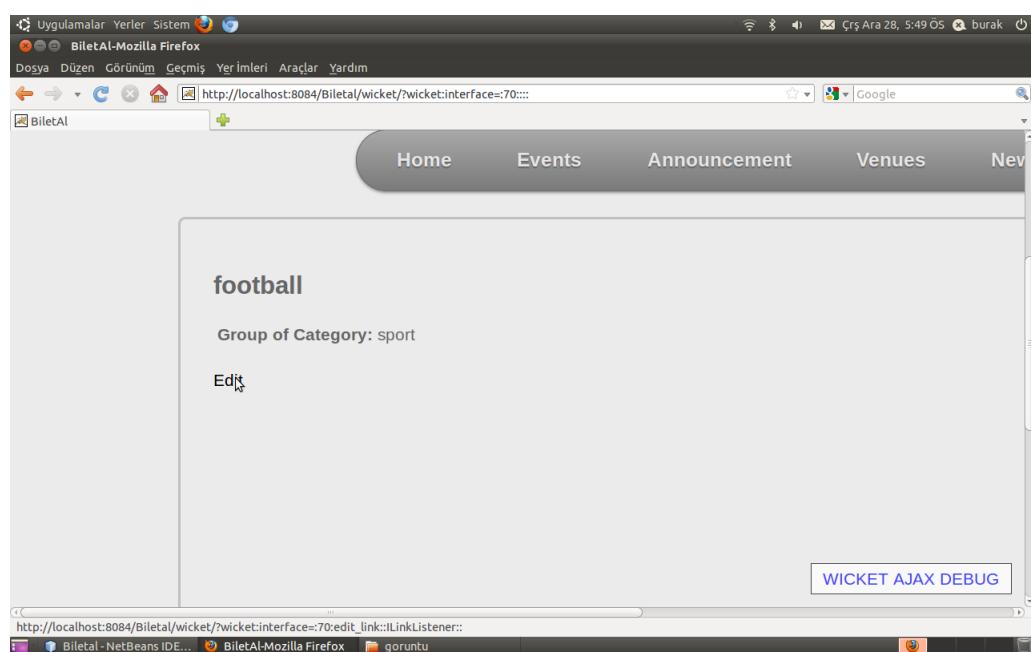


Figure 3.30: Yönetici için CategoryDisplayPage sayfasında güncelleme linki

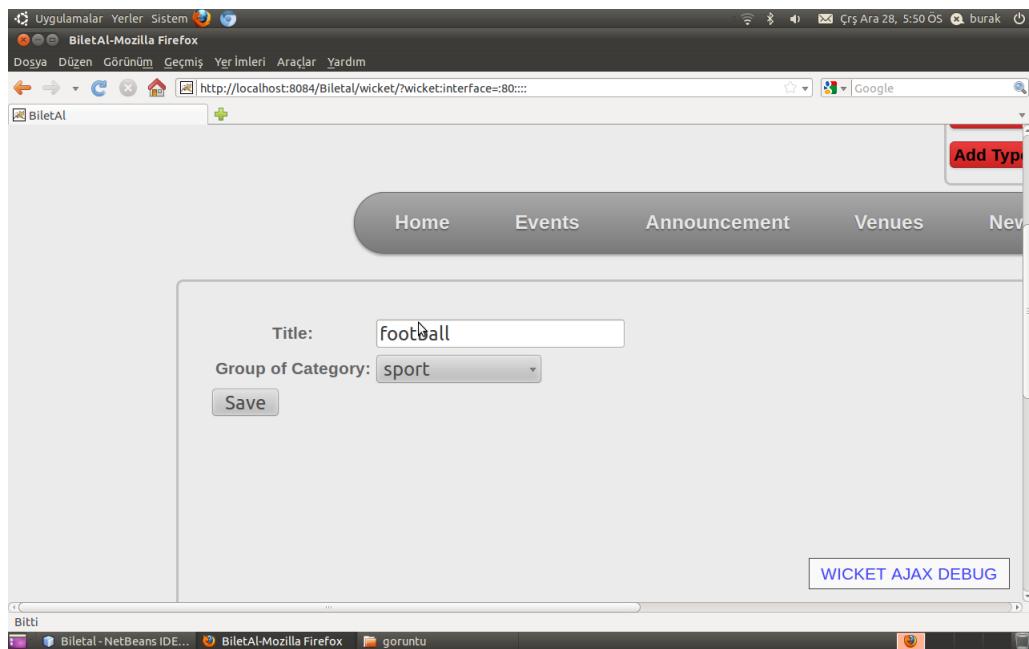


Figure 3.31: Yönetici için CategoryEditPage sayfası

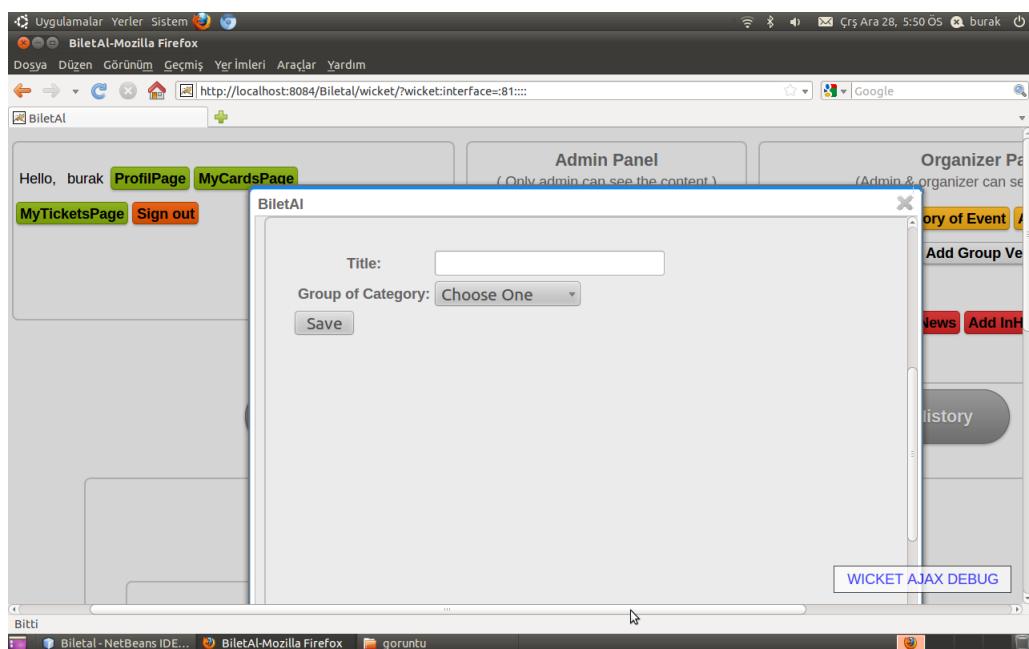


Figure 3.32: Yönetici ve Organizatör için kategori ekleme penceresi

Üst Kategoriler: Üst kategorinin anlaşılması için bir örnek verelim. Örneğin; Fenerbahçe ile Beşiktaş Futbol Kulüpleri arasında oynanan bir karşılaşma etkinliğimiz olsun. Bu etkinliğin kategorisi futboldur. Futbol, basketbol ve voleybol gibi kategorilerin üst kategorisi spordur.

Üst Kategori listesi HeaderPanel'de "Groups of Category" adlı linkle görüntülenebilir. Tüm kullanıcılar için bu sayfa görüntülenebilmektedir. Açılan sayfada, üst kategoriler düzenli bir biçimde görüntülenebilir. Ayrıca yönetici girişi yapılmışsa bu üst kategorilerden "checkbox" ile seçilenler silinebilir. Yönetici girişi yapılmamışsa silinemez. Ayrıca üst kategoriler için "name" üzerinde olan linkle beraber "GroupDisplayPage" sayfasına yönlendirilir. Bu sayfada üst kategori detayları yer alır. Kullanıcı tipi

yönetici ise üst kategoriler için "edit" linki de açılır. Böylelikle üst kategoriler güncellenebilir. Organizatör ve yönetici tipinden kullanıcılar üst kategori eklemek için *Organizer Panel*' de bulunan "add group of category" butonunu kullanırlar. Açılan sayfa Ajaxla oluşturulmuş bir Modal Window sayfasıdır. Bu pencere içerisinde üst kategori eklenir.

Üst Kategoriler için ekran görüntüleri aşağıdaki gibidir;

The screenshot shows a Firefox browser window with the URL <http://localhost:8084/Bileta/wicket/?wicket:interface=:87:::>. The page title is 'GroupCategoryListPage'. The interface is divided into three main panels: 'Hello, burak' (User Profile), 'Admin Panel' (only admin can see the content), and 'Organizer Panel' (admin & organizer can see the content). The 'Group of Category List' section contains a table with columns 'Select' and 'Name'. The 'sport' category is selected, indicated by a checked checkbox. Other categories listed are 'music and fest', 'performing arts', and 'visual art'. A 'Delete' button is at the bottom of the table. A 'WICKET AJAX DEBUG' box is visible on the right side of the page.

Figure 3.33: Üst kategori listeleme sayfası

This screenshot is similar to Figure 3.33, showing the 'GroupCategoryListPage' in a Firefox browser. The 'visual art' category is now selected, indicated by a checked checkbox. The other categories ('sport', 'music and fest', 'performing arts') are not selected. The 'Delete' button is at the bottom of the table. A 'WICKET AJAX DEBUG' box is visible on the right side of the page.

Figure 3.34: Üst kategori listeleme sayfasında yönetici için silme işlemi

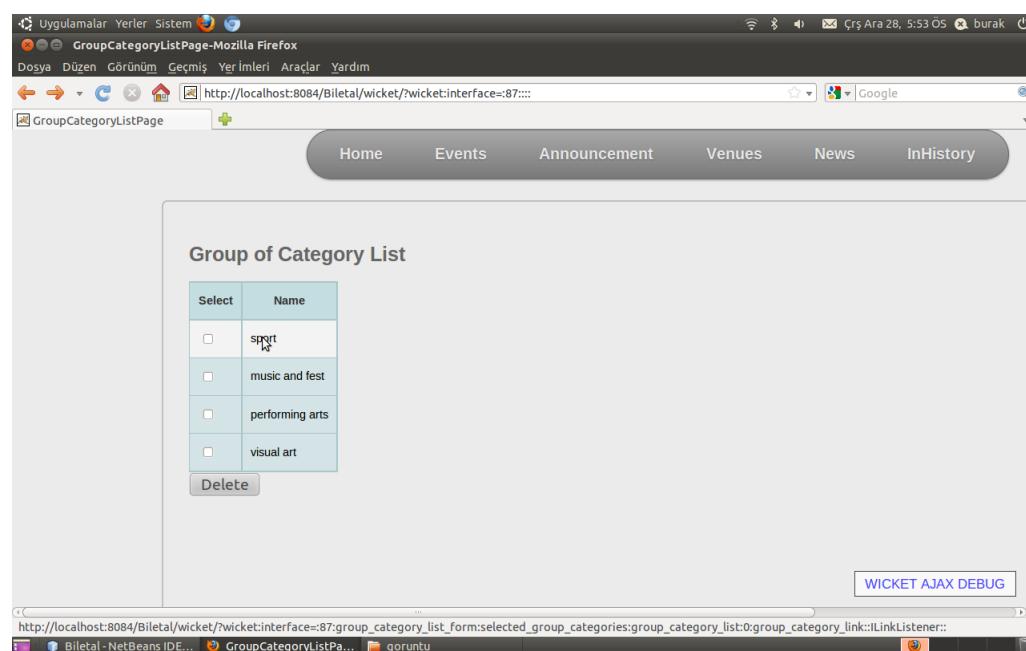


Figure 3.35: Üst kategori listeleme sayfasında GroupDisplayePage için link

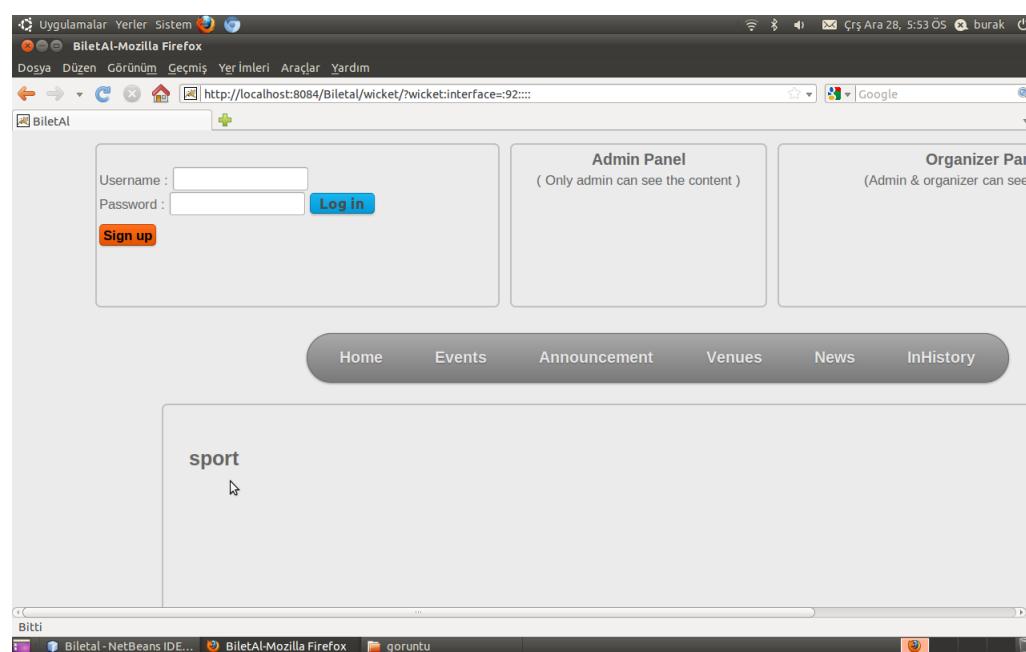


Figure 3.36: Ziyaretçi için GroupDisplayePage sayfası

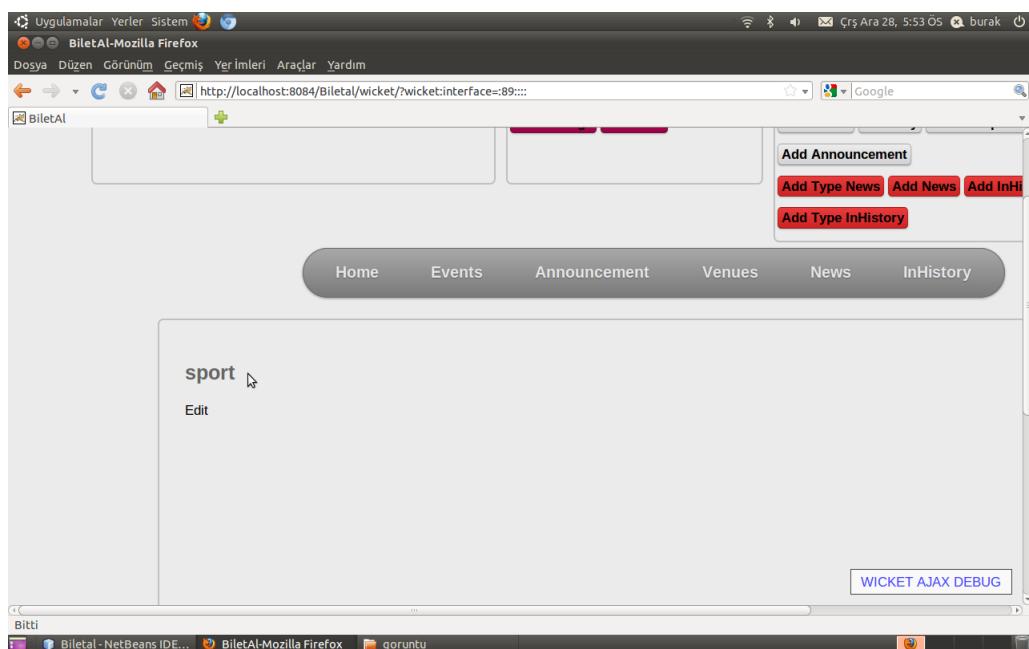


Figure 3.37: Yönetici için GroupDisplayPage sayfasında güncelleme linki

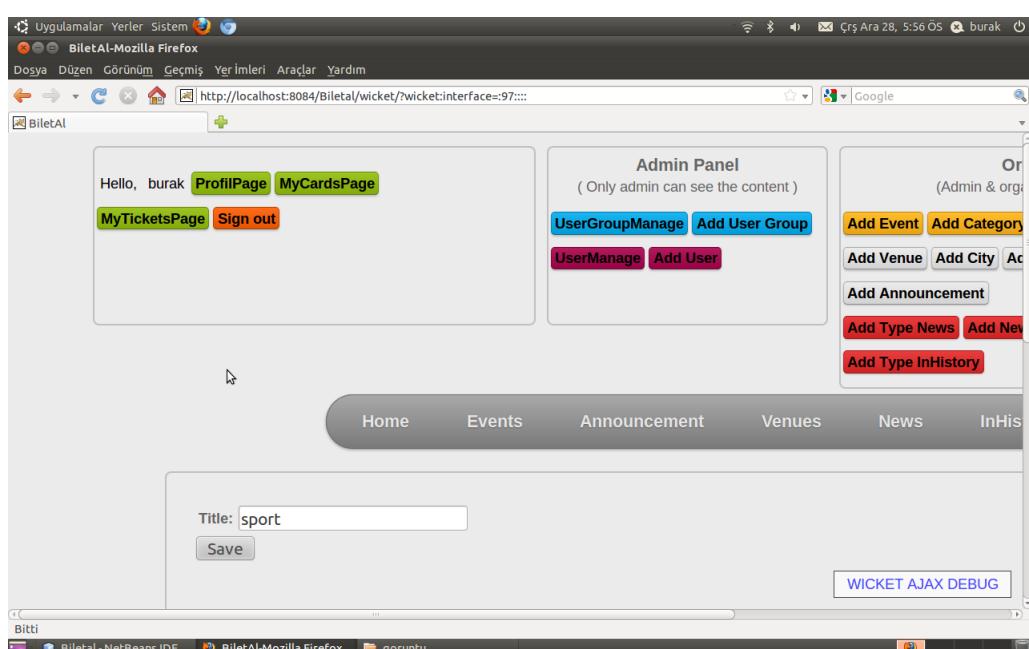


Figure 3.38: Yönetici için GroupEditPage sayfası

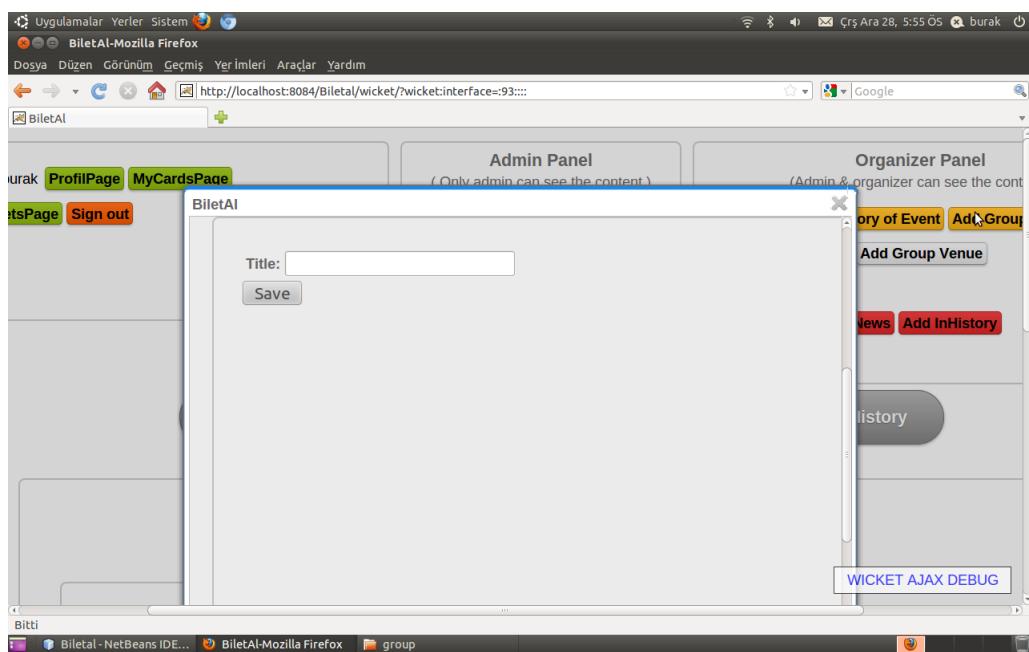


Figure 3.39: Yönetici ve Organizatör için üst kategori ekleme penceresi

Kredi kartları: Kredi kartı 3 üye tipi için hazırlanmıştır. Bunlar bilet satın alabilen üye, organizatör, yöneticidir. Bu kullanıcılar kendilerine ait kredi kartları ekleyebilir, güncelleyebilir ve silebilirler.

Kişiye ait kredi kartları listesi Sign Up-Log In formda "MyCardsPage" adlı linkle görüntülenebilir. Açılan sayfada, kredi kartları düzenli bir biçimde görüntülenebilir. Ayrıca kredi kartlarından "checkbox" ile seçilenler silinebilir. Ayrıca üst kategoriler için "number" üzerinde olan linkle beraber "CardDisplayPage" sayfasına yönlendirilir. Bu sayfada kredi kartlarının detayları yer alır. Kredi kartları için "edit" linki de açılır. Böylelikle kredi kartları güncellenebilir. Kredi kartı eklemek için *MyCardsPage*' de bulunan "add card" butonunu kullanılır. Açılan sayfa Ajaxla oluşturulmuş bir Modal Window sayfasıdır. Bu pencere içerisinde kredi kartı eklenir.

Kredi kartları için ekran görüntüleri aşağıdaki gibidir;

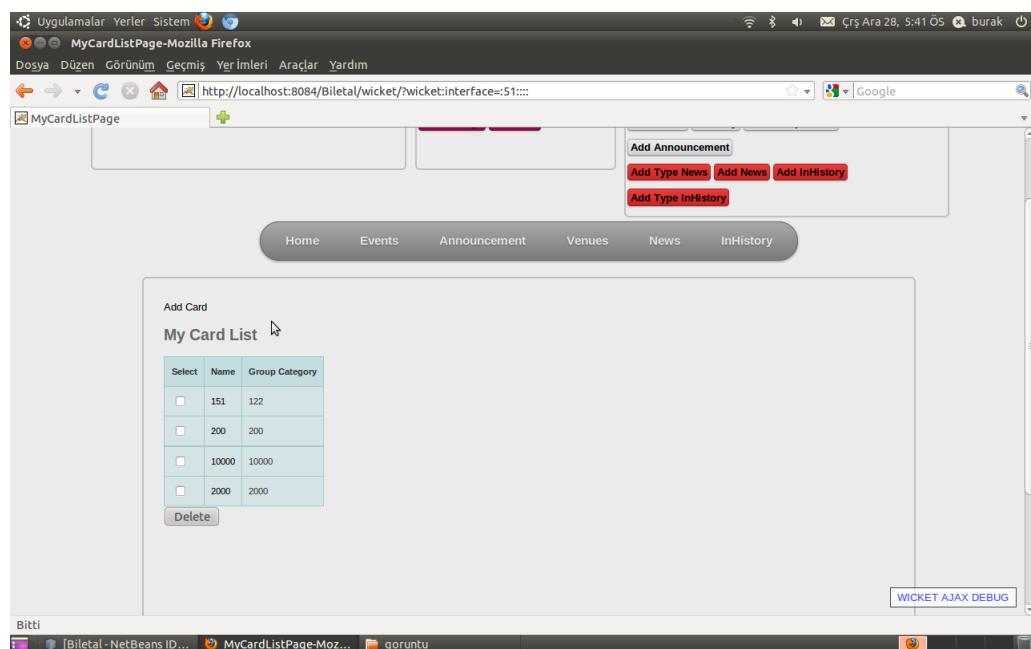


Figure 3.40: Kişiye ait kredi kartlarının listelendiği sayfa

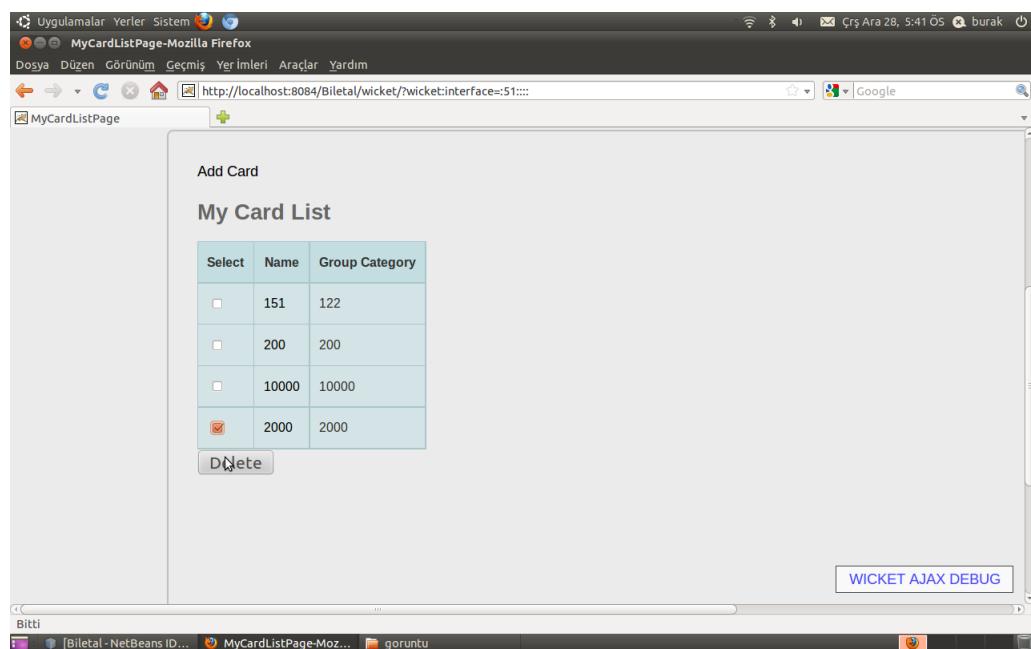


Figure 3.41: Kişiye ait kredi kartlarını silme işlemi

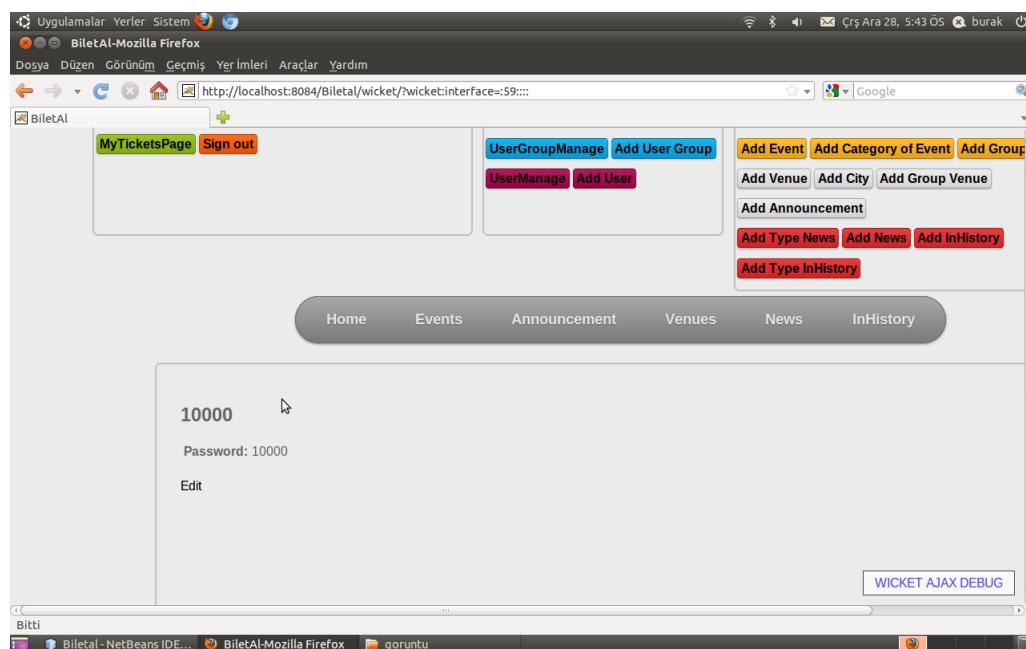


Figure 3.42: Kredi Kartı bilgilerini içeren sayfa

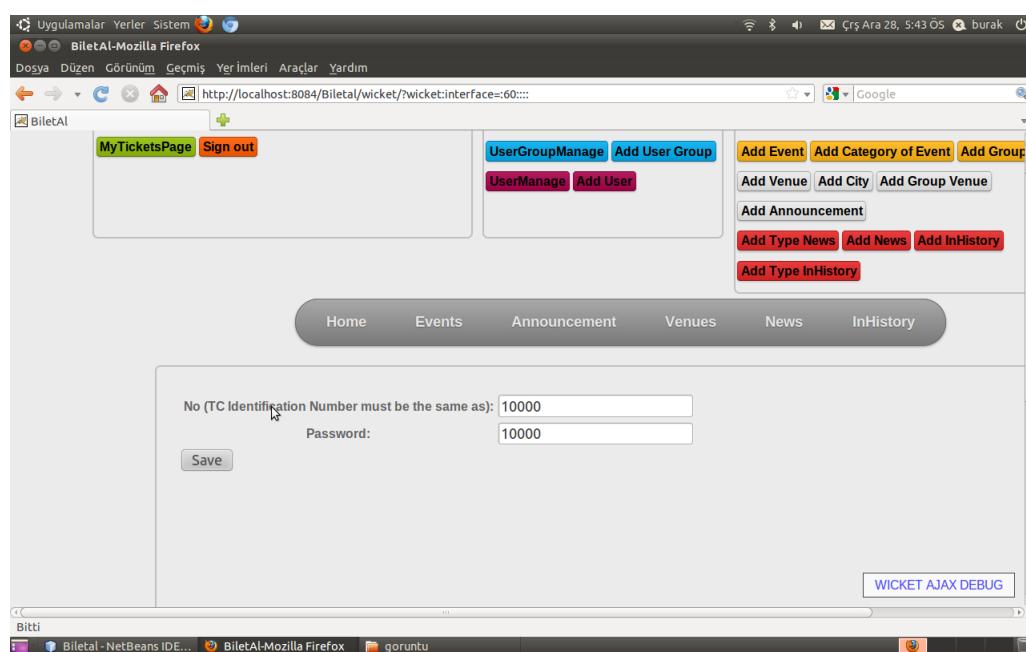


Figure 3.43: Kredi kartı güncelleme sayfası

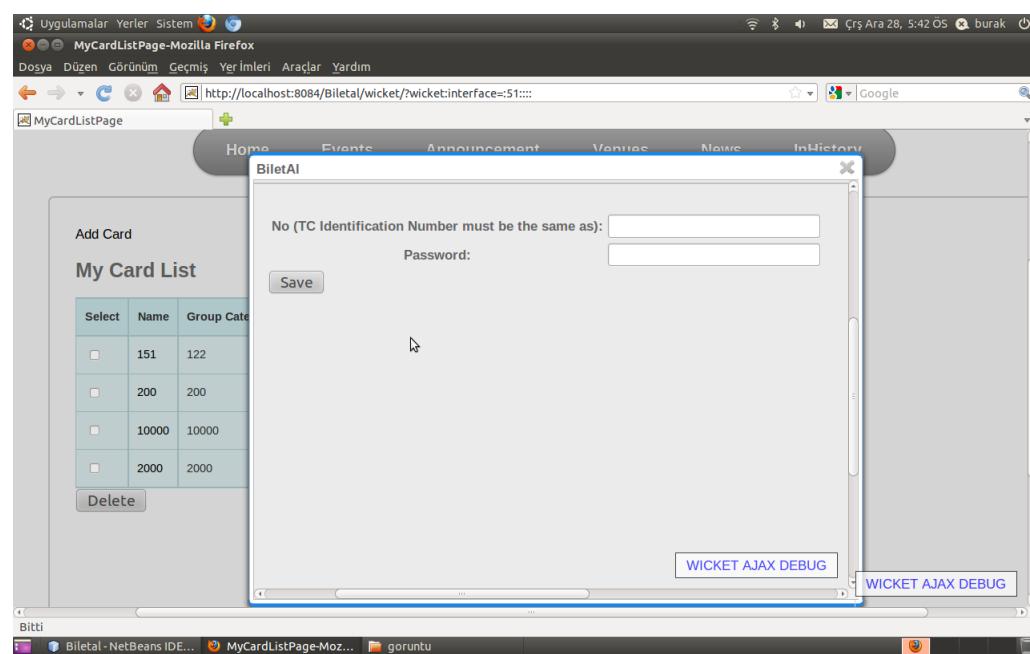


Figure 3.44: Kredi kartı ekleme penceresi

Chapter 4

Teknik Kılavuz

4.1 Veritabanı Tasarımı

Not: Veritabanı tasarımda sadece benim sorumlu olduğum tablolar anlatılacaktır. Bunlar "Event, CategoryEvent, GroupCategory ve Card" tablolarıdır.

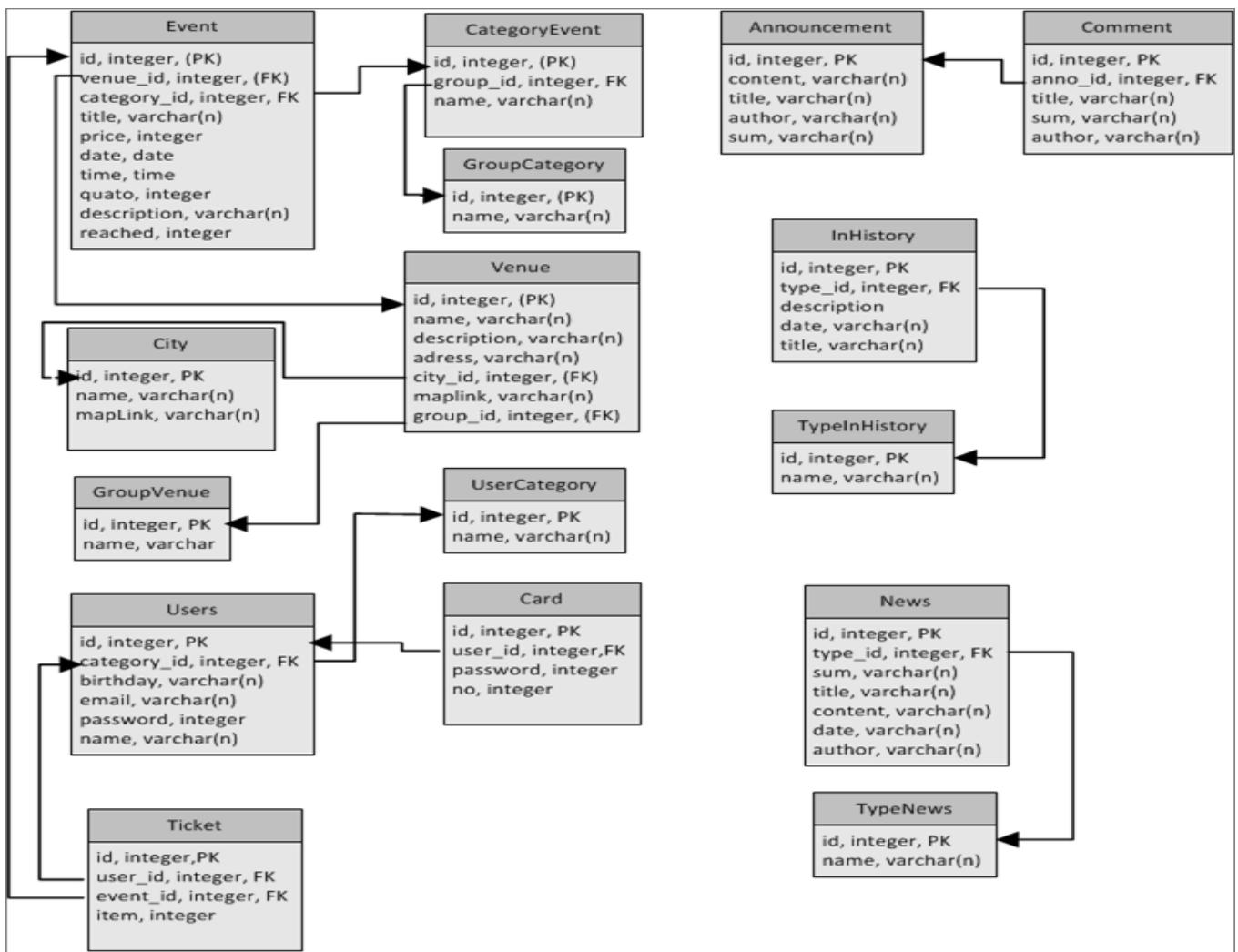


Figure 4.1: Veritabanı varlık-ilişki diyagramı

Event tablosu: Event tablosunu oluşturan SQL kodu;

```
CREATE TABLE event(
    id INTEGER NOT NULL PRIMARY KEY,
    title VARCHAR(100),
    date VARCHAR(20),
    time TIME,
    description VARCHAR(255),
    category_id INTEGER NOT NULL,
    venue_id INTEGER NOT NULL,
    price INTEGER,
    quato INTEGER,
    image BLOB,
    reached INTEGER DEFAULT 0,
    FOREIGN KEY (category_id) REFERENCES event_category(id),
    FOREIGN KEY (venue_id) REFERENCES venue(id)
)
```

Event tablosu sistemimizde yer alan etkinlikleri belirlemek için tasarlanmıştır. Birincil anahtarı id'dir. İsmi title, tarihi date, saatı time, içeriği description, kategori türü category_id, mekanı venue_id, bilet fiyatı price, kontenjan quato, satın alınmış bilet sayısı reached ile tanımlanmıştır. "category_id" CategoryEvent tablosuna; "venue_id" ise Venue tablosuna dış anahtarla bağlanmıştır. Böylelikle mekani ya da kategorisi silinen etkinlik tabloda tutulmaya devam edilecektir.

CategoryEvent tablosu: CategoryEvent tablosunu oluşturan SQL kodu;

```
CREATE TABLE event_category (
    id INTEGER NOT NULL PRIMARY KEY,
    name VARCHAR(100),
    group_id INTEGER NOT NULL,
    FOREIGN KEY (group_id) REFERENCES group_category(id)
)
```

CategoryEvent tablosu sistemimizde yer alan etkinlik kategorilerini belirlemek için tasarlanmıştır. Birincil anahtarı id'dir. İsmi name, üst kategori türü group_id ile tanımlanmıştır. "group_id" GroupCategory tablosuna dış anahtarla bağlanmıştır. Böylelikle üst kategorisi silinen kategori tabloda tutulmaya devam edilecektir.

GroupCategory tablosu: GroupCategory tablosunu oluşturan SQL kodu;

```
CREATE TABLE group_category (
    id INTEGER NOT NULL PRIMARY KEY,
    name VARCHAR(100)
)
```

GroupCategory tablosu sistemimizde yer alan üst kategorileri belirlemek için tasarlanmıştır. Birincil anahtarı id'dir. İsmi name ile tanımlanmıştır.

Card tablosu: Card tablosunu oluşturan SQL kodu;

```
CREATE TABLE card (
    id INTEGER NOT NULL PRIMARY KEY,
    no VARCHAR(100),
    password VARCHAR(100),
    user_id INTEGER NOT NULL,
    FOREIGN KEY (user_id) REFERENCES users(id)
)
```

Card tablosu sistemimizde yer alan kredi kartlarını belirlemek için tasarlanmıştır. Birincil anahtarı id'dir. Kart numarası no, şifresi password, kime ait olduğu user_id ile tanımlanmıştır. "user_id" Users tablosuna dış anahtarla bağlanmıştır. Böylelikle MyCardsPage sayfasında kişiye ait kredi kartları görüntülenebilmektedir.

Veritabanı tablolarını varlık-ilişki bakımında incelediğimizde ; Event ve CategoryEvent tablolarında n'den 1'e ilişki görülür. CategoryEvent ve GroupCategory tablolarında n'den 1'e ilişki görülür. User ve Card tablolarında 1'den n'e ilişki görülür.

4.2 Yazılım Tasarımı

Projede yazdığım kodlar:

1. Application.java
2. BasePage.java
3. HomePage.java
4. HomePage.html
5. HeaderPanel.java
6. HeaderPanel.html
7. Cards, Events, CategoryEvents, GroupCategories, Search package içerisindeki java ve html dosyaları

Yazılan Kodların İçerikleri

1. Application.java

Bu kod en temelinde SQLite sunucusuna ve veritabanı dosyasına bağlanma işlevi görünüyor. Buna Kurulum kısmında detaylı olarak değindiğimiz için tekrar girmiyorum.

Bu dosya içerisinde sistemimizde içerik olarak yer alan etkinlik, kullanıcı, kredi kartı gibi verileri veritabanından gerekli SQL sorguları ile çeken "Collection" nesneleri tanımlanmıştır. Böylelikle sistemimizde yer alan içerikleri herhangi bir işlem için kullanmak istediğimizde Application.java dosyasına gerekli kurucu fonksiyon ile çağrı yaparız. Ayrıca "session" ve "HomePage" çalıştırılabilmesi içinde gerekli metodlar bu dosya eklenmiştir.

```
public class Application extends WebApplication {  
    ...  
    private EventCollection eventCollection;  
    private NewsCollection newsCollection;  
    ...  
    private Connection db;  
    private String pathh;  
  
    public Application() throws SQLException {  
        ...  
        this.eventCollection = new EventCollection(db);  
        this.cardCollection = new CardCollection(db);  
        ...  
    }  
  
    @Override  
    public Session newSession(Request request, Response response) {  
        return new UserSession(request);  
    }  
  
    @Override  
    public Class getHomePage() {  
        return HomePage.class;  
    }  
    public EventCollection getEventCollection(){  
        return this.eventCollection;  
    }  
    ...  
}
```

2. BasePage.java

Projemizdeki birçok sayfa BasePage'den türetilmiştir.

Proje için hazırlanan CSS dosyası (style.css) ve HeaderPanel sayfalara BasePage'den kalıtım özelliği ile iletilmektedir. Ancak unutmamak gereklidir ki BasePage'den türetilen sayfaların html kısmına gerekli wicket componenti eklenmeli. Aksi takdirde sayfalarımız çalışmamayacaktır.

```
public class BasePage extends WebPage {  
    public BasePage() {  
        this(null);  
    }  
    public BasePage(IModel model) {  
        super(model);  
        this.add(new StyleSheetReference("stylesheet", BasePage.class,  
            "style.css"));  
        this.add(new HeaderPanel("mainNavigation"));  
    }  
}
```

3. HomePage.java

Kullanım Kılavuzunda AnaSayfada yer alan içerikler detaylıca anlatılmıştır. Bu yüzden bunlara tekrar girilmeyecek ve kodlar açıklanacaktır. Ayrıca birbirine çok benzeyen mekan (venue), kategori (categoryEvent) ve şehir (city) için hazırlanan kod parçaları birbirinin aynıdır bu yüzden sadece venue gösterilecektir. HomePage BasePage'den türetilmiş bir sayfadır.

AnaSayfada kategori, şehir ve mekana göre etkinlik arama formu benim sorumlu olduğum bölümdür.

Burada önemli iki nokta vardır. Birincisi dış anahtarın değerini değil içeriğini ekranda göstermek için DropDownChoice kullanılmıştır. İkincisi veritabanından verileri çekebilmek için Application türünden nesne yaratılmış ve gerekli metodları çağrılmıştır.

```
public class HomePage extends BasePage {
    private DropDownChoice venueDropDownChoice;
    private Venue venue;
    private Form selectVenueForm;
    private List <Venue> venues;
    public HomePage() {
        Date now = new Date();
        Label labelDateTime = new Label("datetimestamp", now.toString());
        this.add(labelDateTime);

        Application app = (Application) this.getApplication();
        final VenueCollection venueCollection = app.getVenueCollection();
        venues = venueCollection.getVenues();

        ...

        selectVenueForm = new Form("select_venue");

        venueDropDownChoice = new DropDownChoice("for_venue",
            new Model(""), venues, new IChoiceRenderer() {

                @Override
                public Object getDisplayValue(Object object) {
                    return ((Venue) object).getName();
                }
                @Override
                public String getIdValue(Object object, int index) {
                    return Integer.toString(index);
                }
            });
        venueDropDownChoice.setRequired(true);
        selectVenueForm.add(venueDropDownChoice);
        selectVenueForm.add(new Button("submit") {

            @Override
            public void onSubmit() {
                Venue venue = (Venue) venueDropDownChoice.getModelObject();

                this.setResponsePage(new byVenueListPage(venue.getName()));
            }
        });
        this.add(selectVenueForm);
    }
}
```

4. HomePage.html

BasePage'den türetilen HomePage için html kodudur. İlk olarak gerekli CSS dosyası ve HeaderPanel'den gelen "navigation" için componentler eklenmiştir.

Daha sonra HomePage.java dosyasında bulunan Wicket Componentleri için doğru hiyerarşi ile uygun HTML table ve formları eklenerek oluşturulmuştur.

5. HeaderPanel.java

Kullanım Kılavuzunda HeaderPanel'de yer alan içerikler detaylıca anlatılmıştır. Bu yüzden bunlara tekrar girilmeyecek ve kodlar açıklanacaktır. Ayrıca birbirine çok benzeyen kod parçaları tekrarlanmayacaktır.

HeaderPanel'de sistem içindeki etkinlik, mekan, haber gibi içeriklere linkler vardır. Ayrıca Sign Up-Log In Form, Admin ve Organizer Panel vardır. Benim sorumlu olduğum ve yaptığım kısım genel itibariyle "Event, CategoryEvent, GroupCategory ve Card" nesneleri için gerekli linkler ve gerekli panellerin "session, session.getUserId ve session.getUserGroupId" sorgularıyla gösterilip gösterilmemesinin ayarlanmasıdır.

Bu bölümde yazmış olduğum kod blogu aşağıdadır. Ayrıca kullandığım Ajax kodları da bu blok içerisinde gösterilmektedir.

```
public final class HeaderPanel extends Panel {
    public HeaderPanel(String id) {
        super(id);
        UserSession session = (UserSession)getSession();
        ...
        Link EventListLink = new Link("list_events"){

            @Override
            public void onClick(){
                this.setResponsePage (new EventListPage());
            }
        };
        this.add(EventListLink);

        ...

        if (session.isSignedIn() && session.getuserGroup()==1) {

            final ModalWindow m = new ModalWindow("modal");
            final ModalWindow m2 = new ModalWindow("modal2");
            ...
            this.add(new WebMarkupContainer("UserEntry").setVisible(false));
            ...
            Link EventAddLink = new Link("add_event"){

                @Override
                public void onClick(){
                    Event event= new Event();
                    this.setResponsePage (new EventEditPage(event));
                }
            };
            this.add(EventAddLink);

            this.add(new AjaxLink("add_category_events") {

                @Override
                public void onClick(AjaxRequestTarget target) {
                    m.show(target);
                }
            });

            this.add(m);
            m.setPageCreator (new ModalWindow.PageCreator() {

                @Override
                public Page createPage() {
                    CategoryEvent categoryEvent= new CategoryEvent();
                    return new CategoryEventEditPage(categoryEvent,m);
                }
            });
        };

        this.add(new AjaxLink("add_group_categories") {

            @Override
            public void onClick(AjaxRequestTarget target) {
                m2.show(target);
            }
        });
    }
}
```

6. HeaderPanel.html

HeaderPanel için html kodudur.

HeaderPanel.java dosyasında bulunan Wicket Componentleri için doğru hiyerarşi ile uygun HTML table ve formları eklenerek oluşturulmuştur.

7.a Cards Package

Card.java

Card sınıfına ait nesnenin id, no, şifre ve kime ait olduğunu bilgisini içeren user_id'yi tutan java kodlarıdır.

CardCollection.java

CardCollection.java Card nesnelerinin içeriğini, özelliklerini veritabanına ekleyen, nesneleri veritabanından silen, veritabanında güncelleşten, ve çeşitli listeleme sorgulamaları gerçekleştiren java dosyasıdır. En önemli sorgularından biri aşağıdadır. Burada bilet satın alacak kişinin kendine ait kredi kartı ile mi giriş yaptığı sorguluyor:

```
public boolean byCard(int userId, String no, String password) {  
    int count = 0;  
    String query = "SELECT COUNT(*) AS count FROM card WHERE "  
        + "(no = '" + no + "' AND password = '" + password + "'"  
        + "AND user_id = '" + userId + "')";  
    try {  
        Statement statement = db.createStatement();  
        ResultSet result = statement.executeQuery(query);  
        count = result.getInt("count");  
    } finally {  
        result.close();  
    }  
    return count > 0;  
}
```

```
if (count != 0) {
    return true;
} else {
    return false;
}
} catch (SQLException ex) {
    throw new UnsupportedOperationException(ex.getMessage());
}
}
```

CardDisplayPage (Wicket)

CardDisplayPage.java Kredi kartlarının no, password içerikleriyle gösterilmesini sağlayan java kodlarıdır. Buradaki edit linki tıklanarak güncelleme yapılabilir. CardDisplayPage.html dosyasında ise gerekli Wicket Componentleri ve HTML table ve formları vardır. Örnek java kodu aşağıdadır:

```
public final class CardDisplayPage extends BasePage {
    private Card card;
    private int userId;
    public CardDisplayPage(Card card) {
        this.card = card;
        userId = card.getUserId();
        this.add(new Label("no", card.getNo()));
        this.add(new Label("password", card.getPassword()));
        Link editLink = new Link("edit_link") {
            @Override
            public void onClick() {
                CardDisplayPage parent = (CardDisplayPage) this.getParent();
                this.setResponsePage(new CardEditPage(parent.card, false));
            }
        };
        this.add(editLink);
    }
}
```

CardDisplayPageLink.java

CardDisplayPage sınıfının kurucu fonksiyonunu çağırınan sınıfın java kodlarıdır.

CardEditForm.java

CardEditForm.java CardEditPage içerisinde oluşturulan formdur. Kişiye ait kredi kartı eklemek ya da güncellemek için çağrılr. Sınıf içerisinde tanımlanan boolean tipinden bayrak ile kontrol edilerek formun ekleme mi güncelleme mi yapılacağına karar verilir. Ayrıca "session.getUserId" metodu ile kime ait olduğu eklenir. Sınıfa ait java kodları:

```
public class CardEditForm extends Form {
    private boolean newCardFlag;
    UserSession session = (UserSession) getSession();

    public CardEditForm(String id, Card card, boolean newCardFlag) {
        super(id);

        CompoundPropertyModel model = new CompoundPropertyModel(card);
        this.setModel(model);
        this.add(new TextField("no"));
        this.add(new TextField("password"));
        this.newCardFlag = newCardFlag;
    }

    @Override
    public void onSubmit() {
        Card card = (Card) this.getModelObject();
```

```
Application app = (Application) this.getApplication();
CardCollection cardCollection = app.getCardCollection();
if (this.newCardFlag) {
    cardCollection.addCard(card.getNo(), card.getPassword(),
        session.getUserId());
}
else {
    cardCollection.updateCard(card.getId(), card.getNo(),
        card.getPassword(), session.getUserId());
}
this.setResponsePage(new CardDisplayPage(card));
}
```

CardEditPage (Wicket)

CardEditPage.java Kullanıcının kendine ait kredi kartı eklemek için çağrırdığı sınıfın java kodlarıdır. CardEditPage.html dosyasında ise gerekli Wicket Componentleri ve HTML table ve formları vardır. Örnek java kodu aşağıdadır:

```
public final class CardEditPage extends BasePage {

    public CardEditPage(Card card, final ModalWindow m) {

        this.add(new CardEditForm("card_edit", card, true));
    }

    public CardEditPage(Card card, boolean newCardFlag) {

        this.add(new CardEditForm("card_edit", card, newCardFlag));
    }
}
```

MyCardsForm.java

MyCardsForm.java MyCardsPage içerisinde oluşturulan formdur. Kişiye ait kredi kartlarını listelemek için çağrılır. Sınıfa ait java kodları:

```
public class MyCardsForm extends Form {
    private ... selectedCards;
    public MyCardsForm(String id) {
        super(id);
        UserSession session = (UserSession) getSession();
        ...
        List... cards = cardCollection.getMyCards(session.getUserId());
        PropertyListView cardListView =
            new PropertyListView("card_list", cards) {
                @Override
                protected void populateItem(ListItem item) {
                    ...
                }
            };
        cardCheckGroup.add(cardListView);
    }
    @Override
    public void onSubmit() {
        ...
        for (Card card : this.selectedCards) {
            cardCollection.deleteCard(card);
        }
        ...
    }
}
```

MyCardsPage (Wicket)

MyCardsPage.java Kullanıcının kendine ait kredi kartlarını listelemek için çağrırdığı sınıfın java kodlarıdır. Bu sınıf Ajax kodu ile yazılmıştır. Sayfa Modal Window penceresi açılarak görüntülenir. MyCardsPage.html dosyasında ise gerekli Wicket Componentleri ve HTML table ve formları vardır. Örnek java kodu aşağıdadır:

```
public final class MyCardsPage extends BasePage {
    public MyCardsPage() {
        final ModalWindow m = new ModalWindow("modal");
        this.add(new AjaxLink("add_card") {
            @Override
            public void onClick(AjaxRequestTarget target) {
                m.show(target);
            }
        });
        this.add(m);
        m.setPageCreator(new ModalWindow.PageCreator() {
            @Override
            public Page createPage() {
                Card card= new Card();
                return new CardEditPage(card,m);
            }
        });
        MyCardsForm cardListForm = new MyCardsForm("card_list_form");
        this.add(cardListForm);
    }
}
```

MyCardsPageLink.java

MyCardsPage sınıfının kurucu fonksiyonunu çağıran sınıfın java kodlarıdır.

*7.b Events Package**Event.java*

Event sınıfına ait nesnenin niteliklerini tutan, çağrıran getter,setter ve kurucu fonksiyonlarına sahip olan java kodlarıdır.

EventCollection.java

EventCollection.java Event nesnelerinin içeriğini, özelliklerini veritabanına ekleyen, nesneleri veritabanından silen, veritabanında güncelleyen, ve çeşitli listeleme sorgulamaları gerçekleştiren java dosyasıdır. En önemli sorgularından biri aşağıdadır. Burada seçilen kategoriye göre etkinlikler listelenmektedir:

```
public List... getEventsbyCategory(String categoryName) {
    List ... events = new LinkedList...();
    try {
        String query = "SELECT id, title, date, time, description, "
                    + "category_id, venue_id, price, quato FROM event WHERE"
                    + " category_id = (Select id from event_category "
                    + "where(name=' " + categoryName + "' )) ";
        Statement statement = this.db.createStatement();
        ResultSet result = statement.executeQuery(query);
        while (result.next()) {
            int id = result.getInt("id");
            String title = result.getString("title");
            String time = result.getString("time");
            String description = result.getString("description");
            String date = result.getString("date");
            int category = result.getInt("category_id");
            int venue = result.getInt("venue_id");
            int price = result.getInt("price");
            int quato = result.getInt("quato");
            events.add(new Event(id, title, date, time, description,
                category, price, quato, venue));
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

```

        }
    } catch (SQLException ex) {
        throw new UnsupportedOperationException(ex.getMessage());
    }
    return events;
}

```

EventDisplayPage (Wicket)

EventDisplayPage.java Etkinlikleri içerikleriyle gösterilmesini sağlayan java kodlarıdır. Eğer yönetici tipi kullanıcı bu sayfayı açtıysa buradaki edit linki görünür olacak ve güncelleme yapılabilecektir. Ziyaretçi dışında herhangi bir tip kullacı da "buy" linkini görecek ve satın alabilecektir. EventDisplayPage.html dosyasında ise gerekli Wicket Componentleri ve HTML table ve formları vardır. Örnek java kodu aşağıdadır:

```

public final class EventDisplayPage extends BasePage {
    private Event event;
    public EventDisplayPage(final Event event) {
        this.event = event;
        UserSession session = (UserSession)getSession();
        final ModalWindow m = new ModalWindow("modal");
        Application app = (Application) this.getApplication();
        EventCollection eventCollection = app.getEventCollection();
        CategoryEventCollection categoryEventCollection = app.getCategoryEventCollection();
        Event new_event = eventCollection.getEvent(event.getId());
        CategoryEvent category = categoryEventCollection.getCategoryEvent(event.getCategoryid ←
            ());
        ...
        if (session.getUserGroup() == 1) {
        ...
        }
        else{
            this.add(new WebMarkupContainer("edit_link").setVisible(false));
        }
        if(session.isSignedIn()){
            this.add(new AjaxLink("buy_link") {
                ...
            });
        }
        else
        {
            this.add(new WebMarkupContainer("buy_link").setVisible(false));
            this.add(new WebMarkupContainer("modal").setVisible(false));
        }
    }
}

```

EventDisplayPageLink.java

EventDisplayPage sınıfının kurucu fonksiyonunu çağırınan sınıfın java kodlarıdır.

EventEditForm.java

EventEditForm.java EventEditPage içerisinde oluşturulan formdur. Etkinlik eklemek ya da güncellemek için çağrılr. Sınıf içerisinde tanımlanan boolean tipinden bayrak ile kontrol edilerek formun ekleme mi güncelleme mi yapılacağına karar veriliir. Application türünden bir nesne oluşturulur kendi nitelikleri olan mekan ve katgoriler için collection çağrıları yapılır. DropDownChoice kullanılarak mekan ve kategori için tutulan id'ler yerine isimleri getirilir. Ayrıca TextArea componentleri de kullanılır. Sınıfa ait java kodları:

```

public class EventEditForm extends Form {
    ...
    public EventEditForm(String id, Event event, boolean newEventFlag) {
        super(id);
        Application app = (Application) this.getApplication();
    }
}

```

```

final EventCollection eventCollection = app.getEventCollection();
final CategoryEventCollection categoryCollection = app.getCategoryEventCollection();
categoriesEvent = categoryCollection.getCategoryEvents();
...
categoryDropDownChoice = new DropDownChoice("category_event", new Model(""), ←
    categoriesEvent, new IChoiceRenderer() {
    ...
});
categoryDropDownChoice.setRequired(true);
descriptionArea = new TextArea ... ("description", new Model...(""));
descriptionArea.setRequired(true);
CompoundPropertyModel model = new CompoundPropertyModel(event);
this.setModel(model);
...
}
@Override
public void onSubmit() {
    ...
    if (this.newEvent) {
        eventCollection.addEvent(event);
    }
    else {
        eventCollection.updateEvent(event);
    }
    this.setResponsePage(new EventDisplayPage(event));
}
}
}

```

EventEditPage (Wicket)

EventEditPage.java Kullanıcının etkinlik eklemek için çağrıdığı sınıfın java kodlarıdır. EventEditPage.html dosyasında ise gerekli Wicket Componentleri ve HTML table ve formları vardır. Örnek java kodu aşağıdadır:

```

public final class EventEditPage extends BasePage {
    public EventEditPage(Event event) {
        this.add(new EventEditForm("event_edit",event, true));
    }
    public EventEditPage(Event event, boolean newEventFlag) {
        this.add(new EventEditForm("event_edit",event, newEventFlag));
    }
}

```

EventListForm.java

EventListForm.java EventListPage içerisinde oluşturulan formdur. Etkinlikleri listelemek için çağrılr. Application sınıfından nesne ve getcollection metodları kullanılır. DropDownChoice componentleri ile dış anahtar olan mekan ve kategori id'leri yerine isimleri gösterilir. Ayrıca "session.getuserGroup" sorgusu ile yönetici olanlara silme onayı verilir. Sınıfa ait java kodları:

```

public class EventListForm extends Form {
    private List... selectedEvents;
    UserSession session = (UserSession) getSession();
    public EventListForm(String id) {
        super(id);
        ...
    }
    @Override
    public void onSubmit() {
        Application app = (Application) this.getApplication();
        EventCollection eventCollection = app.getEventCollection();
        if(session.getuserGroup()==1){
            for (Event event : this.selectedEvents) {
                eventCollection.deleteEvent(event);
            }
        }
    }
}

```

```
        }
    }
    this.setResponsePage(new EventListPage());
}
}
```

EventListPage (Wicket)

EventListPage.java Kullanıcının etkinlikleri listelemek için çağrıdığı sınıfın java kodlarıdır. Bu sınıf Ajax Arama Formu eklenmiştir. Forma yazılan katarları içeren etkinlikler listelenir. Ardından "Show" linki ile etkinliğin detaylarını gösteren sayfaya gidilir. EventListPage.html dosyasında ise gerekli Wicket Componentleri ve HTML table ve formları vardır. Örnek java kodu aşağıdadır:

```
public final class EventListPage extends BasePage {
    public EventListPage() {
        ...
        final AutoCompleteTextField searchSpare = new AutoCompleteTextField("key", new Model<String>());
        @Override
        protected Iterator<String> getChoices(String input) {
            Application app = (Application) this.getApplication();
            EventCollection eventCollection = app.getEventCollection();
            if (Strings.isEmpty(input))
                return Collections.EMPTY_LIST.iterator();
            List<String> usernames = eventCollection.findEvent(input);
            List<String> choices = new ArrayList(10);
            for (int i=0; i < usernames.size(); i++) {
                choices.add(usernames.get(i));
                if (choices.size() == 10)
                    break;
            }
            return choices.iterator();
        }
    };
    searchSpare.setOutputMarkupId(true);
    Form<Void> auto = new Form<Void>("autocomplete");
    final Label result = new Label("title", searchSpare.getDefaultModel());
    result.setOutputMarkupId(true);
    result.setVisibilityAllowed(false);
    auto.add(searchSpare);
    auto.add(result);
    searchSpare.add(new AjaxFormSubmitBehavior("onchange") {
        @Override
        protected void onSubmit(AjaxRequestTarget target) {
            target.addComponent(result);
        }
        @Override
        protected void onError(AjaxRequestTarget target) {
        }
    });
    this.add(auto);
    Application app = (Application) this.getApplication();
    EventCollection eventCollection = app.getEventCollection();
    Event event = eventCollection.findEvent2((String)searchSpare.getModelObject());
    EventDisplayPageLink showLink = new EventDisplayPageLink("show", event);
    this.add(showLink);
}
}
```

EventListPageLink.java

EventListPage sınıfının kurucu fonksiyonunu çağırın sınıfın java kodlarıdır.

7.c CategoryEvents Package

CategoryEvents.java

CategoryEvent sınıfına ait nesnenin niteliklerini tutan, çağırın getter,setter ve kurucu fonksiyonlarına sahip olan java kodlarıdır.

CategoryEventCollection.java

CategoryEventCollection.java CategoryEvent nesnelerinin içeriğini, özelliklerini veritabanına ekleyen, nesneleri veritabanından silen, veritabanında güncelleyen ve çeşitli listeleme sorgulamaları gerçekleştiren java dosyasıdır. Aşağıda verilen sorguda seçilen kategorinin tüm detayları listelenmektedir:

```
public CategoryEvent getCategoryEvent(int Id) {
    CategoryEvent categoryEvent = new CategoryEvent();
    String query = "SELECT * FROM event_category WHERE "
        + "(id = '" + Id + "')";
    try {
        Statement statement = db.createStatement();
        ResultSet result = statement.executeQuery(query);
        while (result.next()) {
            int id = result.getInt("id");
            String name = result.getString("name");
            int group_name_id = result.getInt("group_id");
            categoryEvent = (new CategoryEvent(id, name, group_name_id));
        }
    } catch (SQLException ex) {
        throw new UnsupportedOperationException(ex.getMessage());
    }
    return categoryEvent;
}
```

CategoryEventDisplayPage (Wicket)

CategoryEventDisplayPage.java Kategorileri içerikleriyle gösterilmesini sağlayan java kodlarıdır. Eğer yönetici tipi kullanıcı bu sayfayı açtıysa buradaki edit linki görünür olacak ve güncelleme yapabilecektir. CategoryEventDisplayPage.html dosyasında ise gerekli Wicket Componentleri ve HTML table ve formları vardır. Örnek java kodu aşağıdadır:

```
public final class CategoryEventDisplayPage extends BasePage {
    private CategoryEvent categoryEvent;
    public CategoryEventDisplayPage(CategoryEvent categoryEvent) {
        ...
        if (session.getUserGroup() == 1) {
            ...
        } else{
            this.add(new WebMarkupContainer("edit_link").setVisible(false));
        }
    }
}
```

CategoryEventDisplayPageLink.java

CategoryEventDisplayPage sınıfının kurucu fonksiyonunu çağırın sınıfın java kodlarıdır.

CategoryEventEditForm.java

CategoryEventEditForm.java CategoryEventEditPage içerisinde oluşturulan formdur. Kategori eklemek ya da güncellemek için çağrıılır. Sınıf içerisinde tanımlanan boolean tipinden bayrak ile kontrol edilerek formun ekleme mi güncelleme mi yapılacağına karar verilir. Application türünden bir nesne oluşturulur kendi niteliği olan üst kategori için collection çağrıları yapılır. Drop-DownChoice kullanılarak üst kategoriler için tutulan id'ler yerine isimleri getirilir. Ayrıca TextArea componentleri de kullanılır. Sınıfa ait java kodları:

```

public class CategoryEventEditForm extends Form {
    ...
    public CategoryEventEditForm(String id, CategoryEvent categoryEvent, boolean ←
        newCategoryEventFlag) {
        super(id);
        Application app = (Application) this.getApplication();
        final CategoryEventCollection categoryCollection = app.getCategoryEventCollection() ←
            ;
        final GroupCategoryCollection groupCategoryCollection
            = app.getGroupCategoryCollection();
        groupCategories = groupCategoryCollection.getGroupCategories();
        groupCategoryDropDownChoice = new DropDownChoice("group_category",
            new Model(""), groupCategories, new IChoiceRenderer() {
            ...
        });
        groupCategoryDropDownChoice.setRequired(true);
        CompoundPropertyModel model = new CompoundPropertyModel(categoryEvent);
        ...
    }
    @Override
    public void onSubmit() {
        ...
        if (this.newCategoryEvent) {
            categoryCollection.addCategoryEvent(categoryEvent);
        } else {
            categoryCollection.updateCategoryEvent(categoryEvent);
        }
        this.setResponsePage(new CategoryEventDisplayPage(categoryEvent));
    }
}

```

CategoryEventEditPage (Wicket)

CategoryEditPage.java Kullanıcının kategori eklemek için çağrırdığı sınıfın java kodlarıdır. Kurucuya Ajax Modal Window için gerekli parametreler eklenmiştir. CategoryEventEditPage.html dosyasında ise gerekli Wicket Componentleri ve HTML table ve formları vardır. Örnek java kodu aşağıdadır:

```

public final class CategoryEventEditPage extends BasePage {
    public CategoryEventEditPage(CategoryEvent categoryEvent, final ModalWindow m) {
        this.add(new CategoryEventEditForm("category_event_edit",categoryEvent, true));
    }
    public CategoryEventEditPage(CategoryEvent categoryEvent, boolean newCategoryEventFlag) ←
        {
        this.add(new CategoryEventEditForm("category_event_edit",categoryEvent, ←
            newCategoryEventFlag));
    }
}

```

CategoryEventListForm.java

CategoryEventListForm.java EventListPage içerisinde oluşturulan formdur. Kategorileri listelemek için çağrırlır. Application sınıfından nesne ve getcollection metodları kullanılır. DropDownChoice componentleri ile dış anahtar olan üst kategori id'leri yerine isimleri gösterilir. Ayrıca "session.getuserGroup" sorgusu ile yönetici olanlara silme onayı verilir. Sınıfa ait java kodları:

```

public class CategoryEventListForm extends Form {
    ...
    UserSession session = (UserSession) getSession();
    public CategoryEventListForm(String id) {
        super(id);
        ...
        Application app = (Application) this.getApplication();
    }
}

```

```

CategoryEventCollection categoryCollection = app.getCategoryEventCollection();
final GroupCategoryCollection groupCategoryCollection = app. ←
    getGroupCategoryCollection();
PropertyListView categoryEventListView =
    new PropertyListView("category_event_list", categoryEvents) {
...
};

categoryEventCheckGroup.add(categoryEventListView);
}
@Override
public void onSubmit() {
    Application app = (Application) this.getApplication();
    CategoryEventCollection categoryCollection = app.getCategoryEventCollection();
    if(session.getuserGroup()==1) {
        for (CategoryEvent categoryEvent : this.selectedCategoryEvents) {
            categoryCollection.deleteCategoryEvent(categoryEvent);
        }
    }
    this.setResponsePage(new CategoryEventListPage());
}
}
}

```

CategoryEventListPage (Wicket)

CategoryEventListPage.java Kullanıcının kategorileri listelemek için çağrıdığı sınıfın java kodlarıdır. CategoryEventListPage.html dosyasında ise gerekli Wicket Componentleri ve HTML table ve formları vardır. Örnek java kodu aşağıdadır:

```

public final class CategoryEventListPage extends BasePage {
    public CategoryEventListPage() {
        CategoryEventListForm categoryEventListForm = new CategoryEventListForm(" ←
            category_event_list_form");
        this.add(categoryEventListForm);
    }
}

```

CategoryEventListPageLink.java

CategoryEventListPage sınıfının kurucu fonksiyonunu çağıran sınıfın java kodlarıdır.

7.d GroupCategories Package

GroupCategory.java

GroupCategory sınıfına ait nesnenin niteliklerini tutan, çağrıran getter,setter ve kurucu fonksiyonlarına sahip olan java kodlarıdır.

CategoryEventCollection.java

GroupCategoryCollection.java GroupCategory (üst kategori) nesnelerinin içeriğini, özelliklerini veritabanına ekleyen, nesneleri veritabanından silen, veritabanında güncelleyen ve çeşitli listeleme sorgulamaları gerçekleştiren java dosyasıdır. Aşağıda verilen soruda seçilen üst kategorinin güncellenmesi yapılmaktadır:

```

public void updateGroupCategory(int new_id, String new_name) {
    try {
        String query = "UPDATE group_category SET name=? " +
            + "WHERE (id=?)" ;
        PreparedStatement statement = this.db.prepareStatement(query);
        statement.setString(1, new_name);
        statement.setInt(2, new_id);
        statement.executeUpdate();
    } catch (SQLException ex) {
        throw new UnsupportedOperationException(ex.getMessage());
    }
}

```

GroupCategoryDisplayPage (Wicket)

GroupCategoryDisplayPage.java Üst kategorileri içerikleriyle gösterilmesini sağlayan java kodlarıdır. Eğer yönetici tipi kullanıcı bu sayfayı açtıysa buradaki edit linki görünür olacak ve güncelleme yapılabilecektir. GroupCategoryDisplayPage.html dosyasında ise gerekli Wicket Componentleri ve HTML table ve formları vardır. Örnek java kodu aşağıdadır:

```
public final class GroupCategoryDisplayPage extends BasePage {
    private GroupCategory groupCategory;
    public GroupCategoryDisplayPage(GroupCategory groupCategory) {
        this.groupCategory = groupCategory;
        UserSession session = (UserSession)getSession();
        ...
        if (session.getUserGroup() == 1) {
        ...
        }
        else{
            this.add(new WebMarkupContainer("edit_link").setVisible(false));
        }
    }
}
```

GroupCategoryDisplayPageLink.java

GroupCategoryDisplayPage sınıfının kurucu fonksiyonunu çağırınan sınıfın java kodlarıdır.

GroupCategoryEditForm.java

GroupCategoryEditForm.java GroupCategoryEditPage içerisinde oluşturulan formdur. Üst kategori eklemek ya da güncellemek için çağrılr. Sınıf içerisinde tanımlanan boolean tipinden bayrak ile kontrol edilerek formun ekleme mi güncelleme mi yapılacağına karar verilir. Sınıfa ait java kodları:

```
public class GroupCategoryEditForm extends Form {
    private boolean newGroupCategoryFlag;
    public GroupCategoryEditForm(String id, GroupCategory groupCategory, boolean ←
        newGroupCategoryFlag) {
        super(id);
        ...
    }
    @Override
    public void onSubmit() {
        ...
        if (this.newGroupCategoryFlag) {
            groupCategoryCollection.addGroupCategory(groupCategory.getName());
        }
        else {
            groupCategoryCollection.updateGroupCategory(groupCategory.getId(), groupCategory ←
                .getName());
        }
        this.setResponsePage(new GroupCategoryDisplayPage(groupCategory));
    }
}
```

GroupCategoryEditPage (Wicket)

GroupCategoryEditPage.java Kullanıcının üst kategori eklemek için çağrırdığı sınıfın java kodlarıdır. Kurucuya Ajax Modal Window için gerekli parametreler eklenmiştir. GroupCategoryEditPage.html dosyasında ise gerekli Wicket Componentleri ve HTML table ve formları vardır. Örnek java kodu aşağıdadır:

```
public final class GroupCategoryEditPage extends BasePage {
    public GroupCategoryEditPage(GroupCategory groupCategory, final ModalWindow m) {
        this.add(new GroupCategoryEditForm("group_category_edit", groupCategory, true));
    }
    public GroupCategoryEditPage(GroupCategory groupCategory, boolean newGroupCategoryFlag) ←
        {
```

```

        this.add(new GroupCategoryEditForm("group_category_edit", groupCategory, ←
            newGroupCategoryFlag));
    }
}

```

GroupCategoryListForm.java

GroupCategoryListForm.java EventListPage içerisinde oluşturulan formdur. Üst kategorileri listelemek için çağrılr. Ayrıca "session.getuserGroup" sorgusu ile yönetici olanlara silme onayı verilir. Sınıfa ait java kodları:

```

public class GroupCategoryListForm extends Form {
    ...
    UserSession session = (UserSession) getSession();
    public GroupCategoryListForm(String id) {
        super(id);
        ...
    }
    @Override
    public void onSubmit() {
        Application app = (Application) this.getApplication();
        GroupCategoryCollection groupCategoryCollection = app.getGroupCategoryCollection();
        if(session.getuserGroup()==1){
            for (GroupCategory groupCategory : this.selectedGroupCategories) {
                groupCategoryCollection.deleteGroupCategory(groupCategory);
            }
        }
        this.setResponsePage(new GroupCategoryListPage());
    }
}

```

GroupCategoryListPage (Wicket)

GroupCategoryListPage.java Kullanıcının üst kategorileri listelemek için çağrırdığı sınıfın java kodlarıdır. GroupCategoryListPage.html dosyasında ise gerekli Wicket Componentleri ve HTML table ve formları vardır. Örnek java kodu aşağıdadır:

```

public final class GroupCategoryListPage extends BasePage {
    public GroupCategoryListPage() {
        GroupCategoryListForm groupCategoryListForm = new GroupCategoryListForm("←
            group_category_list_form");
        this.add(groupCategoryListForm);
    }
}

```

GroupCategoryListPageLink.java

GroupCategoryListPage sınıfının kurucu fonksiyonunu çağrıran sınıfın java kodlarıdır.

7.e Search Package

Kategori ile arama için EventCollection'da kullanılan sorgu:

```

public List<Event> getEventsbyCategory(String categoryName) {
    List<Event> events = new LinkedList<Event>();
    try {
        String query = "SELECT id, title, date, time, description, "
                      + "category_id, venue_id, price, quato FROM event WHERE"
                      + " category_id = (Select id from event_category "
                      + "where(name='" + categoryName + "')) ";
        Statement statement = this.db.createStatement();
        ResultSet result = statement.executeQuery(query);
        while (result.next()) {
            int id = result.getInt("id");
            String title = result.getString("title");
            String time = result.getString("time");

```

```

        String description = result.getString("description");
        String date = result.getString("date");
        int category = result.getInt("category_id");
        int venue = result.getInt("venue_id");
        int price = result.getInt("price");
        int quato = result.getInt("quato");
        events.add(new Event(id, title, date, time, description,
                             category, price, quato, venue));
    }
} catch (SQLException ex) {
    throw new UnsupportedOperationException(ex.getMessage());
}
return events;
}

```

Mekan ile arama için EventCollection'da kullanılan sorgu:

```

public List ..Event> getEventsbyVenue(String venuename) {
    List..Event> events = new LinkedList ..Event>();
    try {
        String query = "SELECT id, title, date, time, description, "
                      + "category_id, venue_id, price, quato FROM event WHERE"
                      + " venue_id = (Select id from venue "
                      + "where(name=' " + venuename + "' )) ";
        Statement statement = this.db.createStatement();
        ResultSet result = statement.executeQuery(query);
        while (result.next()) {
            int id = result.getInt("id");
            String title = result.getString("title");
            String time = result.getString("time");
            String description = result.getString("description");
            String date = result.getString("date");
            int category = result.getInt("category_id");
            int venue = result.getInt("venue_id");
            int price = result.getInt("price");
            int quato = result.getInt("quato");
            events.add(new Event(id, title, date, time, description,
                                 category, price, quato, venue));
        }
    } catch (SQLException ex) {
        throw new UnsupportedOperationException(ex.getMessage());
    }
    return events;
}

```

Şehir ile arama için EventCollection'da kullanılan sorgu:

```

public List ..Event> getEventsbyCity(String city) {
    List..Event> events = new LinkedList ..Event>();
    try {
        String query = "SELECT id, title, date, time, description, "
                      + "category_id, venue_id, price, quato FROM event WHERE"
                      + " venue_id in (Select venue.id from venue,city "
                      + "where( city.name=' " + city + "' ) and "
                      + "( city.id=venue.city_id ) ";
        Statement statement = this.db.createStatement();
        ResultSet result = statement.executeQuery(query);
        while (result.next()) {
            int id = result.getInt("id");
            String title = result.getString("title");
            String time = result.getString("time");
            String description = result.getString("description");

```

```

        String date = result.getString("date");
        int category = result.getInt("category_id");
        int venue = result.getInt("venue_id");
        int price = result.getInt("price");
        int quato = result.getInt("quato");
        events.add(new Event(id, title, date, time, description,
                             category, price, quato, venue));
    }
} catch (SQLException ex) {
    throw new UnsupportedOperationException(ex.getMessage());
}
return events;
}

```

byCategoryEventListForm.java

byCategoryEventListPage içerisinde oluşturulan formdur. Kategoriye göre aranan Etkinlikleri listelemek için çağrılr. Application sınıfından nesne ve getcollection metodları kullanılır. DropDownChoice componentleri ile dış anahtar olan mekan ve kategori id'leri yerine isimleri gösterilir. Sınıfa ait java kodları:

```

public class byCategoryEventListForm extends Form {
    public byCategoryEventListForm(String id, String categoryName) {
        super(id);
        Application app = (Application) this.getApplication();
        EventCollection eventCollection = app.getEventCollection();
        List ..Event> events = eventCollection.getEventsbyCategory(categoryName);
        final CategoryEventCollection categoryEventCollection = app. ←
            getCategoryEventCollection();
        final VenueCollection venueCollection = app.getVenueCollection();
        PropertyListView eventListView =
            new PropertyListView("event_list", events) {

                @Override
                protected void populateItem(ListItem item) {
                    ...
                }
            };
        this.add(eventListView);
    }
}

```

byCategoryEventListPage (Wicket)

byCategoryEventListPage.java Kullanıcının kategoriye göre aradığı etkinlikleri listelemek için çağrırdı sınıfın java kodlarıdır. byCategoryEventListPage.html dosyasında ise gerekli Wicket Componentleri ve HTML table ve formları vardır. Örnek java kodu aşağıdadır:

```

public final class byCategoryEventListPage extends BasePage {
    public byCategoryEventListPage(String categoryName) {
        byCategoryEventListForm form =
            new byCategoryEventListForm("event_list_form", categoryName);
        this.add(form);
    }
    public byCategoryEventListPage() {
        super();
    }
}

```

byCityListForm.java

byCityListPage içerisinde oluşturulan formdur. Şehire göre aranan Etkinlikleri listelemek için çağrılr. Application sınıfından nesne ve getcollection metodları kullanılır. DropDownChoice componentleri ile dış anahtar olan mekan ve şehir id'leri yerine isimleri gösterilir. Sınıfa ait java kodları:

```

public class byCityListForm extends Form {
    public byCityListForm(String id, String city) {
        super(id);
        Application app = (Application) this.getApplication();
        EventCollection eventCollection = app.getEventCollection();
        List<Event> events = eventCollection.getEventsbyCity(city);
        final CategoryEventCollection categoryEventCollection = app. ←
            getCategoryEventCollection();
        final VenueCollection venueCollection = app.getVenueCollection();
        PropertyListView eventListView =
            new PropertyListView("event_list", events) {
                @Override
                protected void populateItem(ListItem item) {
                    ...
                }
            };
        this.add(eventListView);
    }
}

```

byCityListPage (Wicket)

byCityListPage.java Kullanıcının şehire göre aradığı etkinlikleri listelemek için çağrıdığı sınıfın java kodlarıdır. byCityListPage.html dosyasında ise gerekli Wicket Componentleri ve HTML table ve formları vardır. Örnek java kodu aşağıdadır:

```

public final class byCityListPage extends BasePage {
    public byCityListPage() {
        super();
    }
    public byCityListPage(String city) {
        byCityListForm form = new byCityListForm("event_list_form", city);
        this.add(form);
    }
}

```

byVenueListForm.java

byVenueListPage içerisinde oluşturulan formdur. Mekana göre aranan Etkinlikleri listelemek için çağrıılır. Application sınıfından nesne ve getcollection metodları kullanılır. DropDownListChoice componentleri ile dış anahtar olan mekan ve kategori id'leri yerine isimleri gösterilir. Sınıfa ait java kodları:

```

public class byVenueListForm extends Form {
    public byVenueListForm(String id, String venueName) {
        super(id);
        Application app = (Application) this.getApplication();
        EventCollection eventCollection = app.getEventCollection();
        List<Event> events = eventCollection.getEventsbyVenue(venueName);
        final CategoryEventCollection categoryEventCollection = app. ←
            getCategoryEventCollection();
        final VenueCollection venueCollection = app.getVenueCollection();
        PropertyListView eventListView =
            new PropertyListView("event_list", events) {
                @Override
                protected void populateItem(ListItem item) {
                    ...
                }
            };
        this.add(eventListView);
    }
}

```

byVenueListPage (Wicket)

byVenueListPage.java Kullanıcının mekana göre aradığı etkinlikleri listelemek için çağrırdığı sınıfın java kodlarıdır. byVenueListPage.html dosyasında ise gerekli Wicket Componentleri ve HTML table ve formları vardır. Örnek java kodu aşağıdadır:

```
public final class byVenueListPage extends BasePage {  
    public byVenueListPage() {  
        super();  
    }  
    public byVenueListPage(String venueName) {  
        byVenueListForm form = new byVenueListForm("event_list_form", venueName);  
        this.add(form);  
    }  
}
```