

### 1. Tablolarda id görünmemeli.

Sadece view tarafında değişiklik yaparak çözüldü.

### 2. Url kısmında idler görünmemeli. Bunun için modal yapısı kullanılabilir.

Controller ve view tarafında yapılan değişiklikler ile çözüldü. Eski edit.cshtml ve delete.cshtml dosyaları kaldırıldı. Yerlerine `_EditModalPartial.cshtml` ve `_SetPasswordModalPartial.cshtml` geldi. Ayrıca Controller tarafında edit ve delete methodları `EditModal` ve `DeleteModal` ile değiştirildi.

### 3. Aktif pasif durumu toggle değil.

Toggle butonları klasik form submit yerine **AJAX (fetch)** ile çalışacak şekilde revize edildi. Controller metodları **JSON dönecek** biçimde düzenlendi ve başarılı işlem sonrası ilgili satırın aktif durumu ile buton metni dinamik olarak güncellendi. Böylece sayfa yenileme kaldırılarak kullanıcı deneyimi iyileştirildi.

### 4. Iskontoda klavye ile negative değer girilebiliyor. Örneğin -10 yazıp iskonto uygulanmaya çalışınca indirim yapılmıyor.

Views/Siparisler/Take.cshtml içinde iskonto input'u, tarayıcıların `type="number"` alanlarında - / e / E gibi karakterlere izin verebilmesi nedeniyle daha kontrollü bir yapıya alındı ve **yalnızca sayısal girişe** uygun hale getirildi.

`wwwroot/js/siparis-take.js` tarafında iskonto alanına anlık sanitize (input sırasında temizleme) ve 0–100 aralığına **clamp** uygulanarak kullanıcı klavye ile negatif veya sayı dışı karakter giremez duruma getirildi.

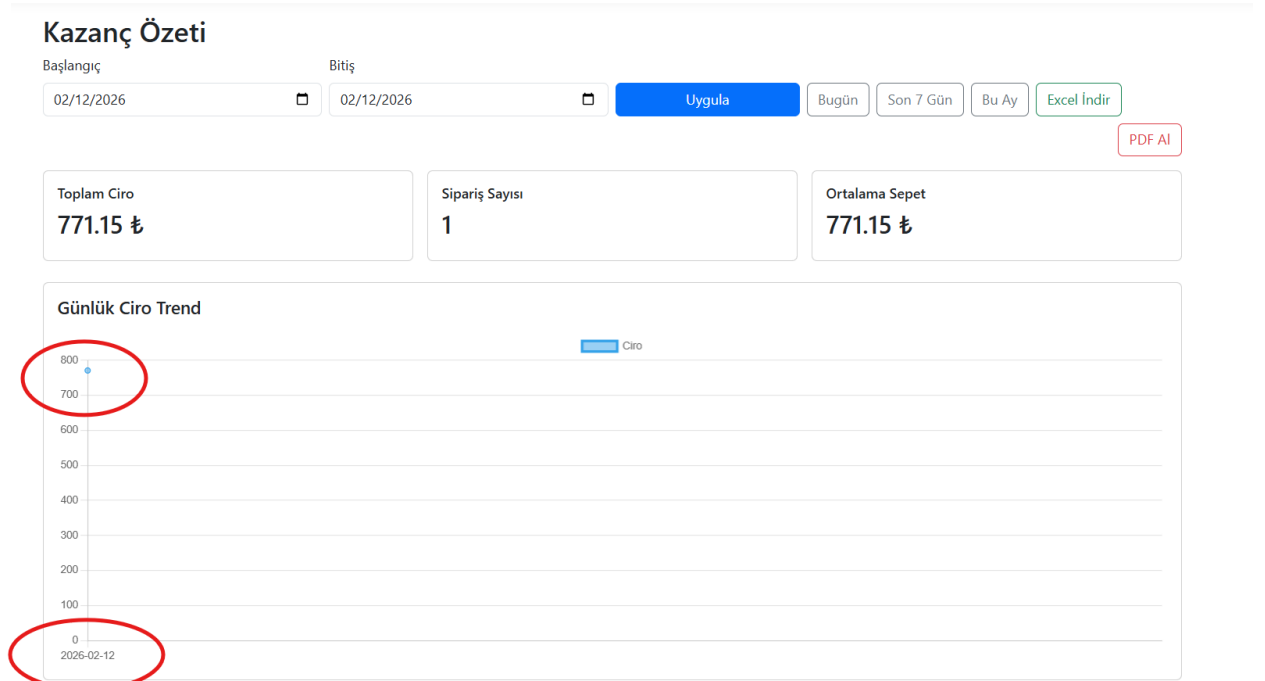
`SiparislerController.UpdateDiscount` içinde `SiparisId` ve `IskontoOran` için kontrol eklenerek, istemci tarafı atlanmış olsa bile `IskontoOran < 0` veya `> 100` durumlarında istek **BadRequest** ile reddedilecek şekilde garanti altına alındı.

`SiparisService.UpdateDiscountRate` içinde de aynı aralık doğrulaması tekrarlanarak iş kuralı **servis katmanında** güvenceye alındı (controller bypass edilse dahi geçersiz iskonto uygulanamaz).

##### 5. Admin kendi hesabının rollerinden admini kaldıramamalı.

**PersonelService.Update** metoduna, kullanıcının kendi kaydını güncellerken mevcut Admin rolünü kaldırmasını engelleyen iş kuralı eklendi. Böylece **server-side** güvence sağlandı. Ayrıca `_EditModalPartial.cshtml` içinde kendi hesabını düzenlerken Admin checkbox'ı **disable** edilerek UI tarafında da görsel kısıt uygulandı.

##### 6. Rapor kısmında bugün kısmi gün bitmediği için mi gelmiyor?



Geliyor. Sadece tabloda x ve y değeri olarak 1'er değeri olduğu için grafik oluşmuyor. Tek bir nokta oluşuyor. Bu da tabloda veri yansımada yanlışlığı oluşturabiliyor.

##### 7. Sepetteki ürünleri adisyona eklemekten sonra ve siparişi kapatmadan sonra başarı mesajı dönmeli.

Siparişler/Take ekranında **Kaydet (Submit)** işlemi başarılı olduğunda, siparis-take.js içinde sunucudan dönen message kullanılarak sayfa yenilenmeden Bootstrap toast ile "Sepetteki ürünler adisyona eklendi" bilgilendirmesi gösterilecek şekilde güncellendi.

**Ödeme Al / Siparişi Kapat (Close)** işleminde ise controller tarafında TempData["Success"] set edilerek kullanıcı Masalar/Board ekranına yönlendirildiğinde sağ üstte “Ödeme başarıyla alındı” toast mesajının görünmesi sağlandı.

## 8. Kategori Yönetimi kısmında Try-Catch yok

Sadece kategori değil herhangi bi bölümün Controller tarafında özellikle herhangi bi hata yakalama işlemi yapılmadı. Hata yakalama ya service katmanına ya da repository katmanına bırakıldı. (basit hatalar if ile yakalandı.) Burada temel amaç, örneğin database kaynaklı bir hatanın Controller katmanında tespit edilmesinin imkanı olmaması ve de controller kısmını try catch ile uzatılarak kod okunurluğunun düşmesini önlemek.

```
KategoriRepository.cs*
RestaurantWeb
72 // Id ile tek kategori getirir
73 public ActionResult<Kategori> GetById(int id)
74 {
75     if (id <= 0)
76         return ActionResult<Kategori>.Fail("Geçersiz kategori id.");
77
78     try
79     {
80         using var conn = new NpgsqlConnection(_connStr);
81         conn.Open();
82
83         const string sql = "...";
84
85         using var cmd = new NpgsqlCommand(sql, conn);
86         cmd.Parameters.AddWithValue("@id", id);
87
88         using var reader = cmd.ExecuteReader();
89         if (!reader.Read())
90             return ActionResult<Kategori>.Fail("Kategori bulunamadı.");
91
92         var kategori = new Kategori
93         {
94             Id = reader.GetInt32(0),
95             Ad = reader.GetString(1),
96             AktifMi = reader.GetBoolean(2),
97             OlusturmaTarihi = reader.GetDateTime(3)
98         };
99
100         return ActionResult<Kategori>.Ok(kategori);
101     }
102     catch (PostgresException ex)
103     {
104         return ActionResult<Kategori>.Fail($"Veritabanı işlemi sırasında bir hata oluştu. (Kod: {ex.SqlState})");
105     }
106     catch (Exception)
107     {
108         return ActionResult<Kategori>.Fail("Beklenmeyen bir hata oluştu. Lütfen daha sonra tekrar deneyin.");
109     }
110 }
111 // Yeni kategori ekler (ad unique)
```

## 9. Session Management (giriş yapan kullanıcı bilgisi)

Mevcut projede bulunanlar:

### ASP.NET Core Cookie Authentication

- Login olunca cookie oluşturuluyor
- Sliding expiration var
- 8 saat expire var
- Logout var

Teknik olarak basit seviyede session management var.

**Fakat geliştirilebilir.**

Mevcut Cookie ayarları:

```
// Cookie tabanlı kimlik doğrulama
builder.Services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)
    .AddCookie(options =>
    {
        options.LoginPath = "/Auth/Login";
        options.LogoutPath = "/Auth/Logout";
        options.AccessDeniedPath = "/Auth/Denied";
        options.Cookie.Name = "RestaurantWeb.Auth";
        options.SlidingExpiration = true;
        options.ExpireTimeSpan = TimeSpan.FromHours(8);
    });
```

Yeni hali:

```
// Cookie tabanlı kimlik doğrulama
builder.Services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)
    .AddCookie(options =>
    {
        options.LoginPath = "/Auth/Login";
        options.LogoutPath = "/Auth/Logout";
        options.AccessDeniedPath = "/Auth/Denied";
        options.Cookie.Name = "RestaurantWeb.Auth";

        options.Cookie.HttpOnly = true;
        options.Cookie.SecurePolicy = CookieSecurePolicy.Always;
        options.Cookie.SameSite = SameSiteMode.Lax;

        options.SlidingExpiration = true;
        options.ExpireTimeSpan = TimeSpan.FromHours(8);

        options.Events = new CookieAuthenticationEvents
        {
            OnValidatePrincipal = async ctx =>
            {
                var userIdStr = ctx.Principal?.FindFirstValue(ClaimTypes.NameIdentifier);
                if (!int.TryParse(userIdStr, out var userId))
                {
                    ctx.RejectPrincipal();
                    await ctx.HttpContext.SignOutAsync();
                    return;
                }

                using var scope = ctx.HttpContext.RequestServices.CreateScope();
                var repo = scope.ServiceProvider.GetRequiredService<PersonelRepository>();

                var activeRes = repo.IsActiveById(userId);

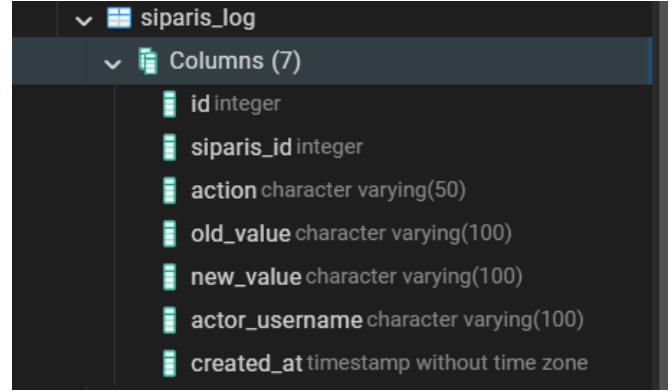
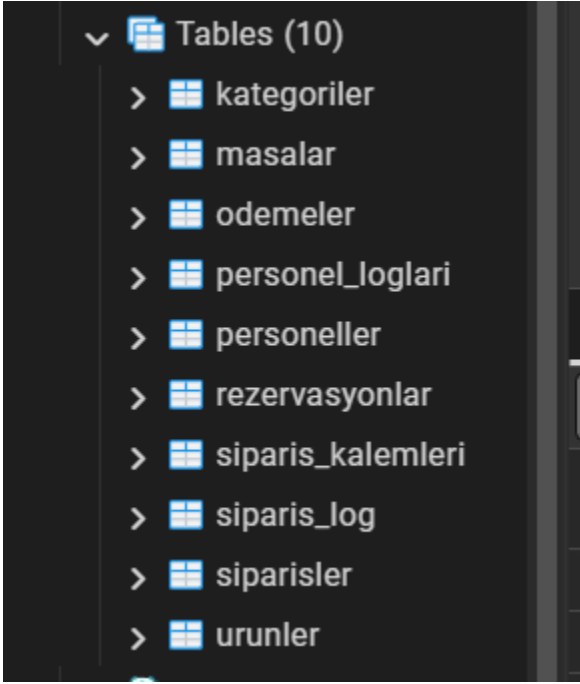
                // bulunamadı / hata / pasif -> oturumu düşür
                if (!activeRes.Success || activeRes.Data == false)
                {
                    ctx.RejectPrincipal();
                    await ctx.HttpContext.SignOutAsync();
                }
            }
        }
    });
```

Bu değişiklikler sonrası:

- Kullanıcı pasif yapılırsa aktif oturum düşüyor.
- Admin bir kullanıcıyı pasif edince o anki session invalidate oluyor.

## 10. Sipariş log tablosu ve loglama

DB’de Sipariş Log tablosu bulunuyor:



Loglama işlemi ise, kullanıcı siparişte bir değişiklik yaptığında siparişi açtığında ya da kapattığında ya da iskonto tutarını değiştirdiğinde gerçekleşiyor.

Kullanıcılar, siparişler sekmesindeki sipariş geçmişinden ilgili siparişin detay kısmına girdiğinde sayfanın alt kısmında İşlem Geçmişini yani SiparisLog unu bulabilir.

Burada işlemin tarihi, işlemin türü, işlem öncesi durum, sonrası durum ve de tüm bu işlemlerin kim tarafından yapıldığı bilgisi tutulmaktadır.

RestaurantWeb Kategoriler Ürünler Masalar Rezervasyonlar Personeller Siparişler Raporlar Mutfak Personel Logları Backup Yönetici İÇİ

### Sipariş Detayı - #315 (Masa 3)

[Geri](#)

**Durum:** Kapalı  
**Açılış:** 2026-02-12 11:16  
**Kapanış:** 2026-02-12 11:17

**Ara Toplam:** 2.070,00 ₺  
**İskonto Oranı:** 15,00 %  
**İskonto Tutarı:** 310,50 ₺  
**Toplam:** 1.759,50 ₺

**Ödeme:** Nakit  
**Tutar:** 1.759,50 ₺  
**Tarih:** 2026-02-12 11:17

Ürün	Adet	Birim	Tutar
Cappuccino	2	95,00 ₺	190,00 ₺
Et Sote	3	320,00 ₺	960,00 ₺
Izgara Köfte	1	295,00 ₺	295,00 ₺
Izgara Tavuk Pirzola	1	285,00 ₺	285,00 ₺
Kola	2	60,00 ₺	120,00 ₺
Soğan Halkası	2	110,00 ₺	220,00 ₺
<b>Toplam:</b>			<b>2.070,00 ₺</b>

**İşlem Geçmişi (SiparisLog)** [Yenile](#)

Tarih	İşlem	Eski	Yeni	Kullanıcı
2026-02-12 11:17:03	Sipariş Kapatma	Açık	Kapalı	Yönetici
2026-02-12 11:17:03	Ödeme	-	Yöntem=0;Tutar=1759,50	Yönetici
2026-02-12 11:16:53	İskonto	5	15	Yönetici
2026-02-12 11:16:38	İskonto	0	5	Yönetici

