

1. Tablolarda id gorunmemeli.

Sadece view tarafında degisiklik yaparak cozuldu.

2. Url kisminda idler gorunmemeli. Bunun icin modal yapisi kullanilabilir.

Controller ve view tarafında yapılan degisiklikler ile cozuldu. 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 donecek** biçimde düzenleni 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 deger girilebiliyor. Ornegin -10 yuazip iskonto uygulanmaya calisinca 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 eklenecek, istemci tarafı atlanmış olsa bile IskontoOran < 0 veya > 100 durumlarında istek **BadRequest** ile reddedilecek şekilde garanti altına alındı.

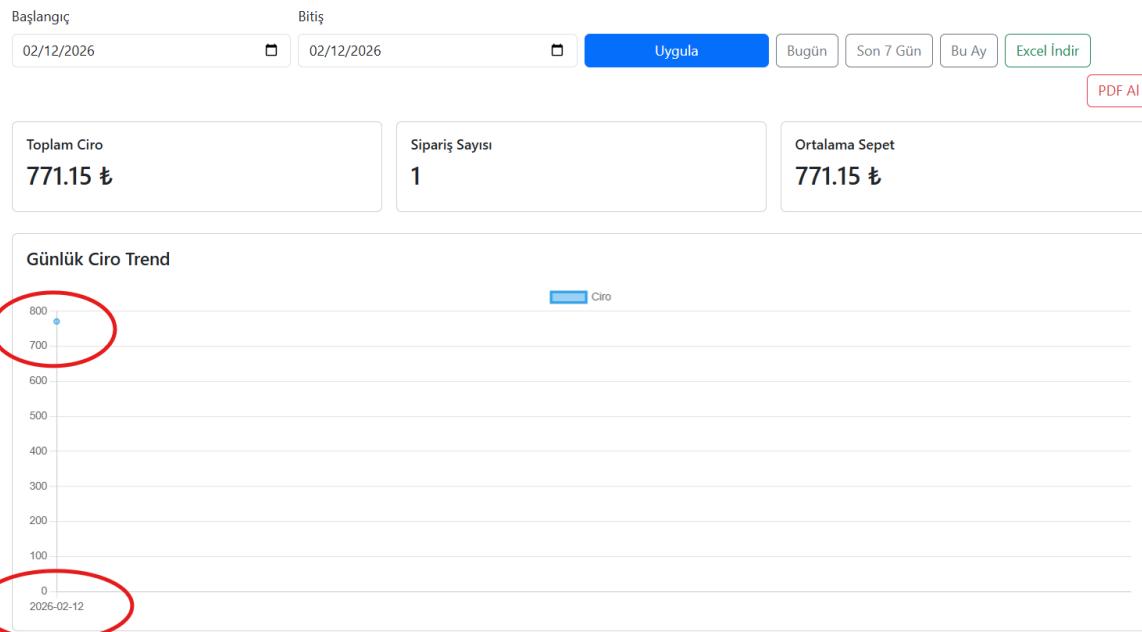
SiparisService.UpdateDiscountRate içinde de aynı aralık doğrulaması tekrarlanarak iş kuralı **servis katmanında** güvenceye alındı (controller bypass edilse dahi geçersiz iskonto uygulanamaz).

5. Admin kendi hesabının rollerinden admini kaldırılamamalı.

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 **disabled** edilerek UI tarafında da görsel kısıt uygulandı.

6. Rapor kısmında bugun kısmı gün bitmediği için mi gelmiyor?

Kazanç Özeti



Geliyor. Sadece tabloda x ve y degeri olarak 1 er deger oldugu icin grafik olusmuyor. Tek bir nokta olusuyor. Bu da tabloda veri yansimadi yanilgisi olusturabiliyor.

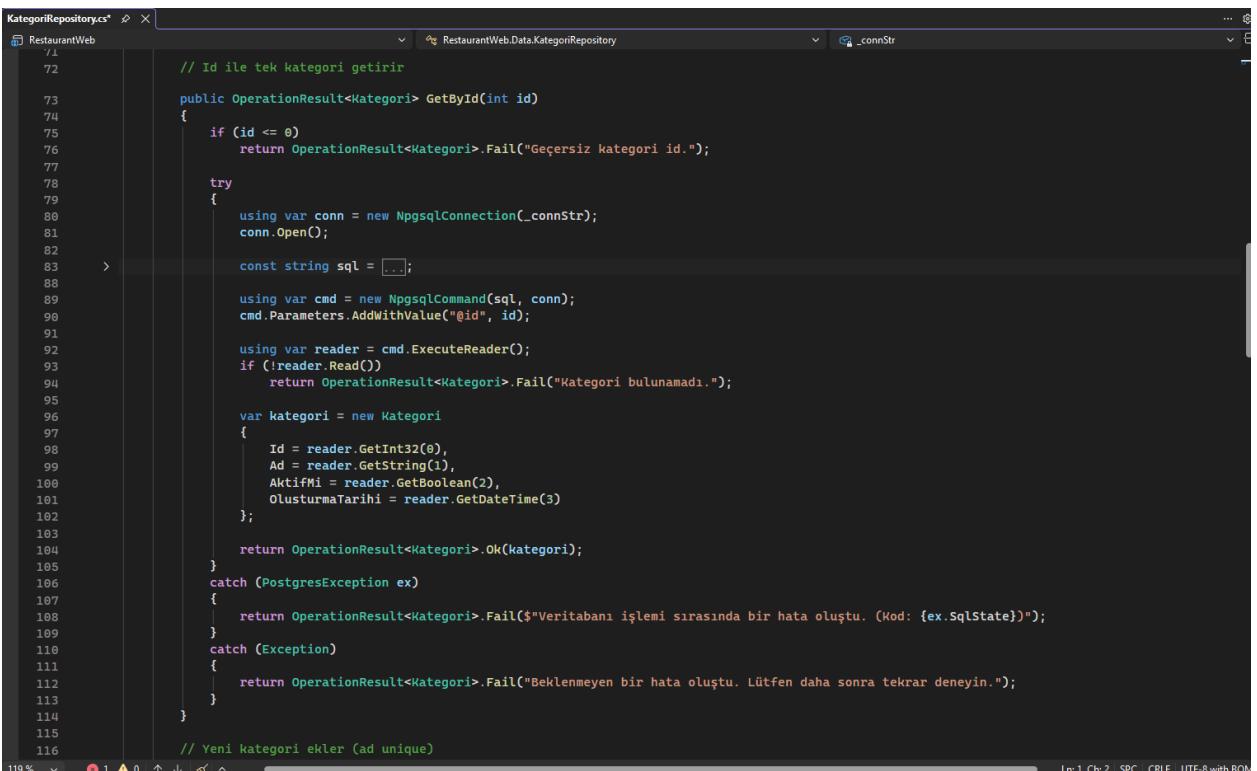
7. Sepetteki ürünler adisyon'a eklenmeden sonra ve siparişi kapatmadan sonra başarı mesajı dönmelii.

Siparisler/Take ekranında **Kaydet (Submit)** işlemi başarılı olduğunda, **siparis-take.js** içinde sunucudan dönen message kullanılarak sayfa yenilenmeden Bootstrap toast ile “Sepetteki ürünler adisyon'a 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 tarafından 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 okunurlugunun düşmesini önlemek.



```
71
72     // Id ile tek kategori getirir
73
74     public OperationResult<Kategori> GetById(int id)
75     {
76         if (id <= 0)
77             return OperationResult<Kategori>.Fail("Geçersiz kategori id.");
78
79         try
80         {
81             using var conn = new NpgsqlConnection(_connStr);
82             conn.Open();
83
84             const string sql = [...];
85
86             using var cmd = new NpgsqlCommand(sql, conn);
87             cmd.Parameters.AddWithValue("gid", id);
88
89             using var reader = cmd.ExecuteReader();
90             if (!reader.Read())
91                 return OperationResult<Kategori>.Fail("Kategori bulunamadı.");
92
93             var kategori = new Kategori
94             {
95                 Id = reader.GetInt32(0),
96                 Ad = reader.GetString(1),
97                 AktifMi = reader.GetBoolean(2),
98                 OlusturmaTarihi = reader.GetDateTime(3)
99             };
100
101             return OperationResult<Kategori>.Ok(kategori);
102         }
103         catch (PostgresException ex)
104         {
105             return OperationResult<Kategori>.Fail($"Veritabanı işlemi sırasında bir hata oluştu. (Kod: {ex.SqlState})");
106         }
107         catch (Exception)
108         {
109             return OperationResult<Kategori>.Fail("Beklenmeyen bir hata oluştu. Lütfen daha sonra tekrar deneyin.");
110         }
111     }
112
113     // Yeni Kategori ekle (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 yapılrsa 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:

The screenshot shows a database interface with two main sections. On the left, under 'Tables (10)', there is a list of tables: kategoriler, masalar, odemeler, personel_loglari, personeller, rezervasyonlar, siparis_kalemleri, siparis_log, siparisler, and urunler. On the right, under 'siparis_log', the 'Columns (7)' are listed with their data types: id (integer), siparis_id (integer), action (character varying(50)), old_value (character varying(100)), new_value (character varying(100)), actor_username (character varying(100)), and created_at (timestamp without time zone).

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 SiparisLogunu bulabilir.

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

The screenshot shows the 'Sipariş Detayı - #315 (Masa 3)' page. At the top, there are details about the order: Durum: Kapalı, Açıldı: 2026-02-12 11:16, Kapatıldı: 2026-02-12 11:17. To the right, payment information is shown: Ödeme: Nakit, Tutar: 1,759.50 ₺, Tarih: 2026-02-12 11:17. Below these, the 'Sipariş Kalemleri' table lists items with their quantities, unit prices, and total amounts. At the bottom, the 'İşlem Geçiği (SiparisLog)' table is highlighted with a red border. It shows a history of actions: Sipariş Kapatma (Status Change), Odeme (Payment), Iskonto (Discount), and another Iskonto entry. The 'Yeni' column indicates the new status after each action, and the 'Kullanıcı' column shows the user 'Yonetici' (Administrator) for all entries.

