-REFAKTORING –

-AMA COSMETICS-

1. Usklađivanje ERD dijagrama sa dijagramom klasa

Pošto konterjnerksi tipovi podataka kao što su liste nisu pogodni za bazu podataka došlo je do promene u modelima klasa. Sve liste su uklonjene kako bi se omogucila perzistencija podataka.
Kako bi se realizovale veze više na više bilo je potrebno uvesti dodatne međuklase i iste smestiti sa ostalim u bazu podataka.
Klase koje su naknadno uvedene zbog ovog problema su: Korpa, Naruceniproizvodi i DodeljeniPopusti.
U nastavku ćemo videti da svaka klasa sadrzi dva strana ključa kako bi se omogućila veza više na više.

- Korpa:

```
namespace AMA_cosmetics.Models
{
    25 references
    public class Korpa
    {
        [Key]
        11 references
        public int Id { get; set; }
        [ForeignKey("Proizvod")]
        10 references
        public int IDProizvod { get; set; }
        17 references
        public Proizvod Proizvod { get; set; }
        15 references
        public string UserName { get; set; }
        [ForeignKey("Korisnik")]
        10 references
        public string IDKorisnik { get; set; }
        0 references
        public Korisnik Korisnik { get; set; }
        11 references
        public int Cijena { get; set; }
        [DisplayName("Kolicina")]
        14 references
        public int kolicina { get; set; }
        1 reference
        public Korpa() { }
    }
}
```

- DodeljeniPopusti:

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace AMA_cosmetics.Models
{
    2 references
    public class DodeljeniPopusti
    {
        [Key]
        0 references
        public int ID { get; set; }

        [ForeignKey("PremiumKorisnik")]
        0 references
        public int IDKorisnika { get; set; }

        [ForeignKey("Popust")]
        0 references
        public int IDPopust { get; set; }

        0 references
        public PremiumKorisnik PremiumKorisnik { get; set; }
        0 references
        public Popust Popust { get; set; }
    }
}
```

- NaruceniPoroizvodi:

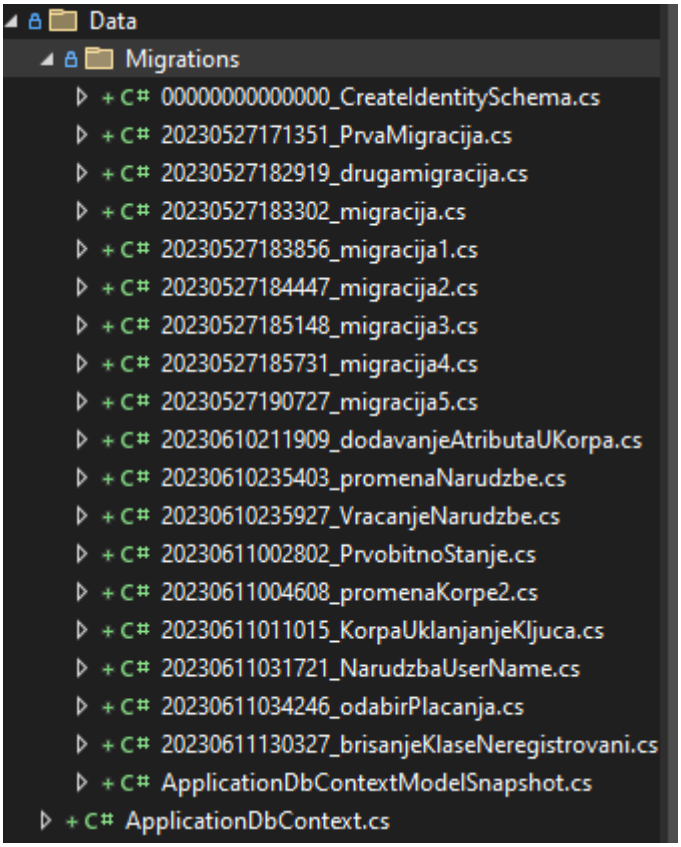
```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Data;

namespace AMA_cosmetics.Models
{
    3 references
    public class NaruceniProizvodi
    {
        [Key]
        0 references
        public int ID {get; set;}
        [ForeignKey ("Proizvod")]
        0 references
        public int IDProizvod { get; set; }
        0 references
        public Proizvod Proizvod { get; set; }
        0 references
        public int  Kolicina { get; set; }
        [ForeignKey("Narudzba")]
        0 references
        public int IDNarudzba { get; set; }
        0 references
        public Narudzba Narudzba { get; set; }
        0 references
        public NaruceniProizvodi() { }


    }
}
```

2. Brisanje nepotrebnih klasa

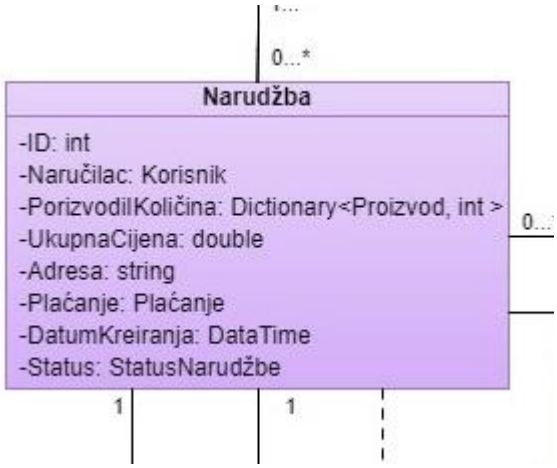
Nakon upoznavanja sa bazom podataka i kontrolom pristupa klasa NeregistrovaniKorisnik koja je označavala gosta u sistemu postaje nepotrebna. S obrzirom na postojanje AspNetUsers i ostale klase koje se odnose na korisnike postaju nepotrebne. Iste nisu uklonjene zbog toga što sadrže specifične attribute te mogu biti potrebne kod daljeg razvoja aplikacije.
Nakon toga je izvršena nova migracija u bazu podataka.



3. Dodavanje novih atributa

Uklasu Narudzba dodani su novi atributi zbog omogućavanja korisnicima da umesto svog imena, prezimena, broja telefona, grada i adrese unesu neku drugu u slučaju da to žele. Dakle, neće se koristit njihovi podaci uneti prilikom registracije već uvek novi podaci koji će se unositi prilikom kreiranja svake narudžbe.

Atributi prvobitne klase:



Atributi nove verzije klase:

```
namespace AMA_cosmetics.Models
{
    22 references
    public class Narudzba
    {
        [Key]
        11 references
        public int ID { get; set; }
        9 references
        public string UserName { get; set; }
        12 references
        public string Ime { get; set; }
        12 references
        public string Prezime { get; set; }
        12 references
        public string Grad { get; set; }
        12 references
        public string Adresa { get; set; }
        [DisplayName("Broj telefona")]
        12 references
        public string BrojTeleona { get; set; }
        [DisplayName("Ukupna cijena")]
        9 references
        public double UkupnaCijena { get; set; }
        [ForeignKey("Placanje")]
        6 references
        public int PlacanjeID { get; set; }
        9 references
        public Placanje Placanje { get; set; }
        [DisplayName("Odabir načina plaćanja")]
        3 references
        public VstaPlacanja vrsta { get; set; }
        [DisplayName("Datum kreiranja narudžbe")]
        12 references
        public  DateTime DatumKreiranja { get; set; }
        9 references
        public  StatusNarudzbe status { get; set; }
        0 references
        public Narudzba() { }

    }
}
```

4. Dodavanje uloga u tabelu AspNetRoles

Kako bi se izvršila kontrola pristupa bilo je neophodno dodati uloge u sistemu u tabelu AspNetRoles . Potom su iste dodeljene korisnicima iz tabele AspNetUsers pomoću međutabele AspNetUseRoles.

Izgled tabele AspNetRoles koja definiše uloge korisnika u sistemu je prikazana ispod.

Id	Name
1	Administrator
2	RegistrovaniKorisnik
3	PremiumKorisnik

3 record(s) affected.

5. Kontrola pristupa

Bilo je neophodno dodati kontrolu prostupa u pogledima i kontrolerima kako bi se sprečilo da neki korisnici pristupaju nekim funkcionalnostima koje nisu predviđene za njih.
Kako bismo samo za administratora omogućili kreiranje, izmenu i brisanje proizvoda (Create, Edit i Delete) bilo je neophodno izvršiti kontrolu pristupa u ProizvodController i u IndexView koji je vezan za model Proizvodi.

```
// GET: Proizvod/Create
[Authorize(Roles = "Administrator")]
0 references
public IActionResult Create()
{
    return View();
}

// POST: Proizvod/Create
// To protect from overposting attacks, enable the specific properties you want to bind to.
// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
[Authorize(Roles = "Administrator")]
0 references
public async Task<IActionResult> Create([Bind("ID,Naziv,Opis,Cijena,Kolicina,Kategorija,SlikaP
{
    if (ModelState.IsValid)
    {
        _context.Add(proizvod);
        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }
    return View(proizvod);
}

// GET: Proizvod/Edit/5
[Authorize(Roles = "Administrator")]
0 references
public async Task<IActionResult> Edit(int? id)
{
    if (id == null)
    {
        return NotFound();
    }

    var proizvod = await _context.Prizvod.FindAsync(id);
    if (proizvod == null)
    {
        return NotFound();
    }
    return View(proizvod);
}
```

```
<td>
        @if (User.IsInRole("Administrator"))
        {
            <a asp-action="Edit" asp-route-id="@item.ID">Edit</a>
            <nobr>|</nobr>
            <a asp-action="Delete" asp-route-id="@item.ID">Delete</a>
            <nobr>|</nobr>

        }

        <a asp-action="Details" asp-route-id="@item.ID">Details</a>
        @if (User.IsInRole("RegistrovaniKorisnik") || User.IsInRole("PremiumKorisnik"))
        {
             <nobr>|</nobr>
            <a asp-controller="Korpa" asp-action="Add" asp-route-id="@item.ID"><i class="fa-solid fa-bag-shopping"></i></a>
        }


</td>
```