İbrahim Faruk Doğan

Rast Academy Task

09/02/2025

Projeden anlaşılanlar

Kanban board yapılması, yani proje iş akışını görselleştirmeye ve görevleri yönetme tablosu yapılmam istenmektedir.

Bu panonun yapılması için Backend kurulup veriler depolanmalı ve frontend'e bu veriler çekilip kullanıcıya gösterilmelidir.

Frontend tarafında kullanıcılar kendi panolarını açabilmeli fakat panolara ulaşmak istiyorlarsa, açtıkları panoların isimlerini bilmelidirler.

Kullanıcılar kendi panolarında gelen 4 tane varsayılan listeyi değiştiremezler. Fakat içlerinde kart ekleyip, çıkartma, güncelleme(renk,isim, etiket) veya listeler arası taşıma ya da aynı liste içerisinde sıralarını değiştirme yapabilirler.

Backend'de hem Boards, hem Lists, hem de Cards tabloları verileri depolamak için tutulmalıdır. Ayrıca bu tablolar birbirleriyle one to many bağlantıları bulundurmalıdırlar.

Frontend'de Url Board'lara ulaşmak için dinamik olmalı. Board oluşturulduğunda verilen isim/numara vesayre ile Board'ın bulunduğu sayfaya ulaşabilmeli.

Birde Frontend'de localstorage'da son gezilen sayfalar tutulmalı, bunun için her frontend sayfasına gelindiğinde localstorage'a istek gönderen bir fonksiyon yapmalı.

Backend Tarafında Yapılanlar

İlk olarak backend ile başladım. Backend'i .net mvc olarak seçtim, önce appsettings.json ve program.cs üzerinden bağlantılarımı gerçekleştirdim ve 3 model oluşturdum.

İlki Boards modeliydi. Boards modeli için, 4 alan belirledim. Id, Başlığı tutması için Title, Yapanı tutması için CreatorName ve Foreign key ile bağlanacak listeleri göstermek için bir ICollection tanımladım.

İkinci olarak Lists modeline geçtim. Lists modeli için, 5 alan belirledim. Id, Başlığı tutması için Title, Many to one bağlantısı için Foreign key tanımladım yani int BoardId alanı açtım ve bu alanın tablosunu göstermek için Board'ı çağırdım. Son olarak da bağlanacak Cards'ları göstermek için bir ICollection tanımladım.

Üçüncü olarak Cards modeline geçtim. Cards modeli için, 7 alan belirledim. Id, Başlığı tutması için Title, Arkaplan rengini tutması için Color, Etiket vermek için Tag, Sıralamalarını tutmak için Order, Many to one bağlantısı için Foreign key tanımladım yani int ListId alanı açtım ve bu alanın tablosunu göstermek için Lists'ı çağırdım.

Boards tablosundaki verileri çekecek olan dinamik sayfanın Boards'ın "Title" alanına bağlı olması gerektiğine karar verdim ve "Title" alanına "unique" özelliği oluşturmam gerektiğine karar verdim. Ardından bunun için, Validators'a geçtim ve url'in dinamik olduğunda sorun yaşamaması için Boards tablosuna Title alanının unique olmasını kontrol eden bir validatör ekledim. Bu validator'u program.cs'de tanımladım:

```
builder.Services.AddValidatorsFromAssemblyContaining<BoardsValidator>();
builder.Services.AddFluentValidationAutoValidation();
```

Ardından Dbcontext oluşturdum ve tablolarımı tanımladım. Ardından bu tablolar arasındaki parent'ı olan tabloları tanımladım(cards ve lists), böylece parent silinince kendi verileride silinebilsin:

```
modelBuilder.Entity<Lists>()
      .HasOne(l => l.Board)
      .WithMany(b => b.Lists)
      .HasForeignKey(l => l.BoardId)
      .OnDelete(DeleteBehavior.Cascade);
```

Ondan sonra da Boards controller'a geçtim ve crud fonksiyonlarını tanımlayıp validator kontrol ettim.

```
if (!ModelState.IsValid)
    return BadRequest(ModelState);
```

Crud fonksiyonları olarak, id yerine title'a göre boards verisi çekme fonksiyonu tanımladım ve boards'ı çekerken list vs card'ları da çekmesini ekledim. Ardından create ederkenki kısımda, 4 default list(Backlog, To do, In progress, Done) ekleme isteğini ekledim.

Ardından Cards'ın controller'ını oluşturdum ve Crud işlemlerini tanımladım. Cards'ta yeni kart eklerken otomatik sıra sayısı yapması için düzenleme yaptım bunun için, önce listId'sine göre listeyi çektim ardından listedeki verilerin order alanındaki max sayısını aldım ve ona 1 ekleyerek sırasını düzenledim, eğer listedeki tek üyeyse otomatik olarak order 1 olacaktır.

```
var highestOrder = await _context.Cards
     .Where(c => c.ListId == cards.ListId)
     .MaxAsync(c => (int?)c.Order) ?? 0; // If no cards exist,
default to 0

 var newCard = new Cards
 {
     Title = cards.Title,
     Color = cards.Color,
     Tag = cards.Tag,
     Order = highestOrder + 1, // Auto increment the order
     ListId = cards.ListId,
     List = list
 };
```

Liste controller'ı eklemedim. Hem boards listeleri döndürüyor hem de liste oluşturma/silme istenmediği için

Frontend Tarafında Yapılanlar

İlk olarak Angular projesini kurdum ardından backend'i bağlamak için service ve modelleri oluşturdum. Öncelikle Cards,Boards,Lists tablolarının modellerini interface'lere aktardım. Ardından service kurulumuna geçtim. İlk olarak Board ekleme yapmam gerektiğine karar verdim ve service'e createBoard metodunu ekledim. Ayrıca app.config.ts içerisinde provideHttpClient'ı ekledim ki http işlemlerini yapabileyim.

Ardından Home component'ını oluşturup, içine talimatlar yazıp, board oluşturma buton'u ekleyip router'da bağladım.

BoardCreate component'ını yaptım ve basit bir arayüz ekleyip, Formbuilder aracılığı ile formumu kurdum ve boards.service ile birleştirip submit fonksiyonumu kurdum.

Fakat test ettiğimde 500 hatası alıyordum ve bu yüzden boardscontroller'a try and catch kurdum ve program.cs içerisine

`builder.Services.AddLogging();` ekledim.

Backend terminalime düşen hata benden iç içe döngü durumundan dolayı referencehandler.preserve kullanmamı istediğini araştırmam sonucunda anladım. Program.cs'ye ekleyince sorun çözeldi.

```
System.Text.Json.JsonException: A possible object cycle was detected.
This can either be due to a cycle or if the object depth is larger
than the maximum allowed depth of 32. Consider using
ReferenceHandler.Preserve on JsonSerializerOptions to support cycles.
```

Ardından BoardsDetail component'ı ekledim. Router'da ":title" diyip component olarak BoardsDetail'i işaret ettim ki doğru title girince o title'ın sayfasına ulaşalım. Ondan sonra BoardsDetail'i doldurmaya başladım. İlk olarak service'i çekip title'a göre get isteği yapan fonksiyonumu doldurup BoardsDetail'e import ettim ve ngOnInit implementasyonuyla yani render başladığında, ActivatedRoute kullanarak url'deki parametreyi çektim ve service api'sinden gelen veriyi board değişkenine aktardım. Ardından bu veriyi html'de yazdırdım. Fakat gelen verinin içindeki liste ve card'lar object verileri olarak geldiklerini gördüm ve ayrıca id'ler de gelen response'un içindeydiler. Bunu çözmek için backend'e geri döndüm.

Önce backend'de bunu çözmek için DTO'lar yani veri transfer yaparken özel objeler yapılması gerektiğine karar verdim ve boards/lists/cards için dto'lar oluşturdum id'yi döndürmeyen şekilde modelledim. Card Controller'i Dto'ya göre tekrar düzenledim

Ardından title'a göre get döndüren boards controller'ımdaki fonksiyona geri döndüm ve boardDTO temelinde bir response döndürmesini sağladım. Ayrıca "select" query isteği ile içeri eklenen list ve card'ların dizi olarak döndürülmesini sağladım:

```
var boardDTO = new BoardDTO
{
    Title = board.Title,
    CreatorName = board.CreatorName,
    Lists = board.Lists.Select(l => new ListDTO
    {
        Title = l.Title,
        Cards = l.Cards.Select(c => new CardDTO
        {
            Title = c.Title,
            Tag = c.Tag,
            Color = c.Color
        }).ToList()
    }).ToList()
  };
```

Fakat yine aynı hatayı aldım. Console log'ları kullanarak öğrendim ki diziler $values altında oluyorlarmış. Bunu çözmek için html'de ngFor sonuna $values ekledim:

```
<div *ngFor="let list of board.lists.$values">
<div *ngFor="let card of list.cards.$values">
```

Böylece gelen veriyi yazdırmayı başarabildim. Buradan itibaren kart ekleme yapmaya odaklandım ve bunun için CardCreate html'i oluşturdum. Bu html Title, Color ve Tag alanlarını içerecek şekilde yaptım ve typescript dosyasını buna göre düzenledim.

CardCreate'i BoardsDetail'de listelerin üzerine "hover" yaptığımda çıkan buton aracılığı ile navigate yapmasını sağladım. Ardından card oluşturma testlerimi yaptım ve sonucun başarılı olduğunu gördüm.

Ardından Card-Update'e geçtim. Card Update için backend'deki card controller'inde yeni bir DTO ve PUT api'si oluşturdum. Ardından BoardsDetail'de cards'ların üzerine "hover" yaptığımda çıkan buton aracılığı ile navigate yapmasını sağladım. Birde cards üzerine çarpı butonu ekledim ve backend'deki card controller'inde yeni bir Delete api'si oluşturdum. Silince hem listeden hem de kart tablosundan silmesi için:

```
var list = await _context.Lists.FirstOrDefaultAsync(l => l.Id == listId);

list.Cards.Remove(card);
```

Son olarak card drag and drop işlemine geçtim. Bunun için DragDropModule import ettim. Ardından html'de hem listeler arası drag event, hem de card'lar arası event'ler kurdum. Son olarak card'ın taşınabilmesi için cdkDrag ekledim. Event tetiklediğinde çağırılması için onCardDrop fonksiyonunu kurdum. Bu fonksiyonda tutup bırakılan kartın, önce bırakıldıkları listeleri çektim. Ardından aynı liste de olup olmadığını kontrol ettim. Eğer değilse diziden çıkarıp diğer listeye eklemesini yazdım. Birde kartın sırasını güncelledim.

Fakat testlerimde fark ettim ki sadece tutup bıraktığım kartın sırası değişiyor, sildiğimde de sıralarda değişiklik olmuyor. Bunun için hem sildiğim yerde hemde tutup bırakma fonksiyonunda listeleri çektim ve tekrar sıralamalarını yaptırdım:

```
const cardOrderDTOCurrent: CardOrderDTO[] = list.cards.$values.map((c:
Cards, index: number) => {

            return {

              id: c.id,

              order: index + 1

            };

          });
```

1'den başlayıp sona kadar giden yeni bir sıralama oluşturdum ve card'ları buna göre güncelledim. Backend'de de güncelleme olması için Cards controllerın'da patch api'si açtım ve liste şeklinde card'ları gönderdim:

```
public async Task<IActionResult> PatchCardOrder([FromBody] List<CardOrderDTO>
cards)
{

    foreach (var card in cards)
    {
        var dbCard = await _context.Cards.FindAsync(card.Id);
        if (dbCard != null)
        {
            dbCard.Order = card.Order;
        }
    }

 }
```

Ardından geçmiş sayfaları gösteren component'ı yapmaya başladım. Bunun için yeni bir component açtım RecentPages adında, ardından yeni bir service açtım localstorage'a ekleme yapacak ve localstorage'dan çekecek:

```
addPage(url: string): void {
    let pages = this.getPages();
    pages = pages.filter(page => page !== url);
    pages.unshift(url);
    if (pages.length > 5) {
      pages.pop();
    }
    localStorage.setItem(this.storageKey,
JSON.stringify(pages));
  }
  getPages(): string[] {
    const pages = localStorage.getItem(this.storageKey);
    return pages ? JSON.parse(pages) : [];
  }
```

Bundan sonra RecentPages'in html vs css'lerini yaptım. Ardından her sayfaya:

```
this.recentPagesService.addPage(window.location.href);
```

Ekledim. Böylece o sayfayı açtıklarında sayfa localstorage'a eklenecekti.

Son olarak testlerimi yaptım ve github'a Readme dosyasıyla beraber yükledim.