مشروع CMS

-اللغة المستخدمة في المشروع:

C#, ASP.NET WebApi Project

في حلي لهذا المشروع حاولت التبسيط قدر الامكان، استخدمت N-Tier Architecture مكونة من 4 طبقات

1-CMS:

في هذة الطبقة يوجد مشروع ASP.NET Web Api وهو يحتوى على كل Controller و Configurations الخاصة بالمشروع

2-Business Layer:

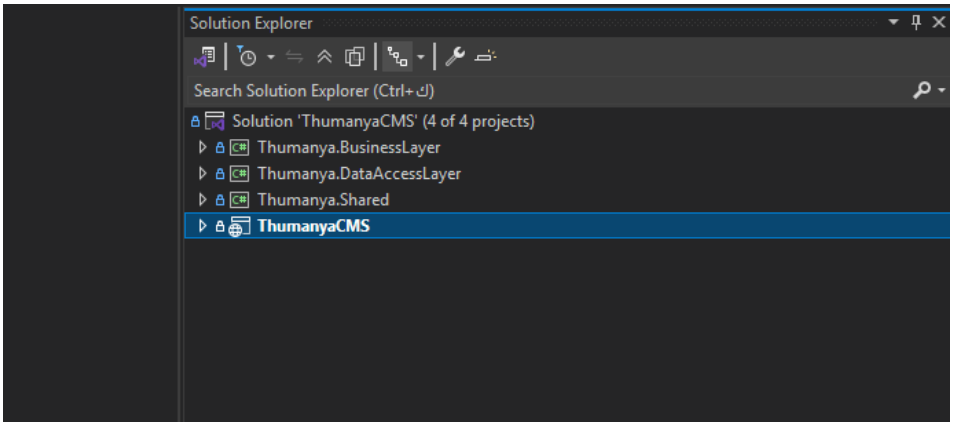
وهي الطبقة الوسطى في المشروع ويوجد فيها كل Logicالخاص بالبيزنس و اي Service ، مفصولا عن ال Controllers

3-DataAccessLayer:

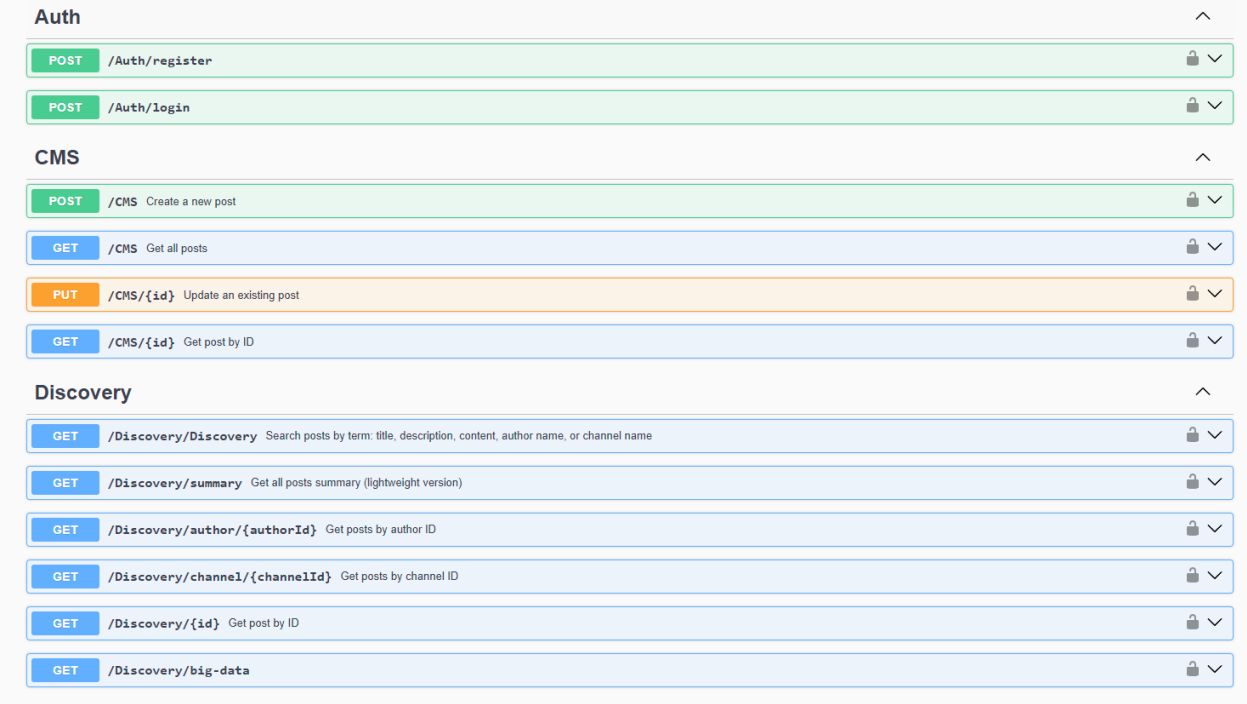
مهمة هذة الطبقة هي جلب البيانات من قاعدة البيانات فقط ، تحتوي على كل Database Entitiesو Migrationsو ال Repositoriesالخاصة بكل Entity

4-Shared:

هي طبقة منفصلة تحتوي على كل ما هو مشترك بين باقي الطبقات مثل Dtos



شرح مبسط لل Endpoints



عند تشغيل المشروع يظهر **Swagger Endpoint** مقسمة الى

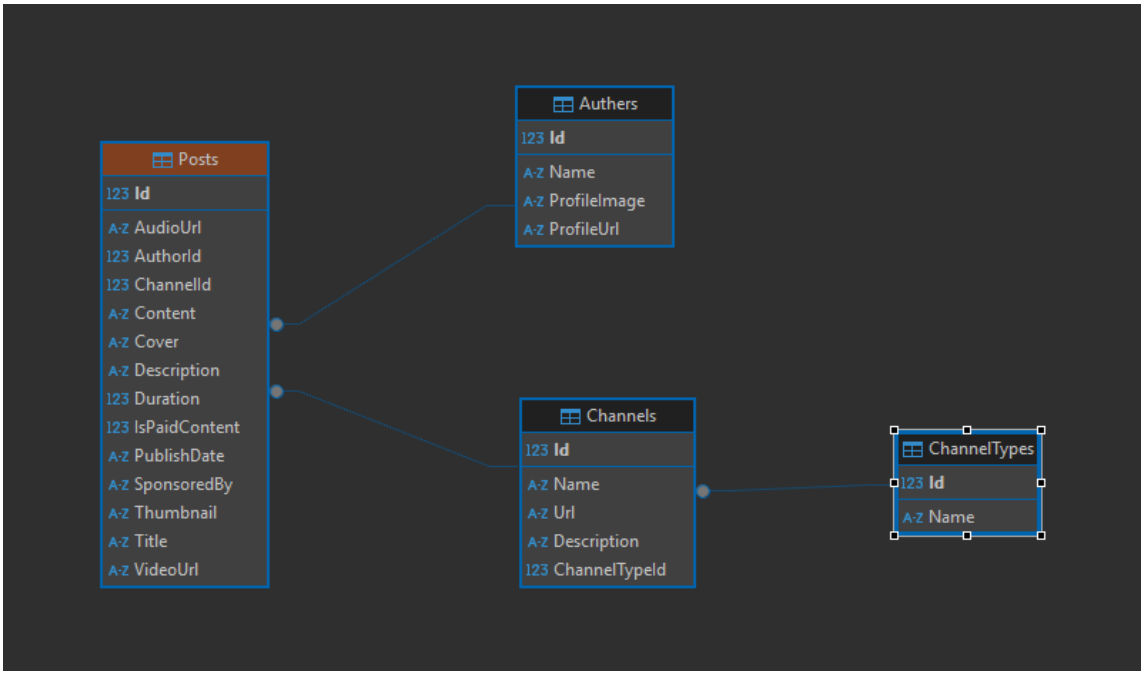
Auth وهي المسؤوله عن Login باستخدام JWT

CMS وهي المسؤوله عن انشاء المقالات وتعديلها

Discovery وهي المسؤولة عن جلب البيانات للمستخدم

قاعدة البيانات

للتسهيل استخدمت قاعدة بيانات SQLite وارفقتها مع المشروع لتسهيل تشغيل المشروع ، **مع العلم انه يجب استخدام قاعدة بيانات اخرى في البيئة الفعلية Production لتتحمل العدد الكبير من المستخدمين**



جدول **Post** يعبر عن المحتوى الرئيسي سواء كان مقالا او بودكاست

جدول **Authers** لكتاب المحتوى ، هنا استخدمت One to many relationship بافتراض ان كل Post سيتم كتابته او تقديمة بواسطة شخص واحد فقط، **في حالة تعدد الكتاب يجب استخدام Many To Many Relationship**

جدول **Channels** يعبر عن القناه التي ستقدم المحتوى ، وجدول **Channel Types** يعبر عن نوعها سواء كانت Podcast او Video

تطبيق ال Scalabilityو تحسين الاداء

قمت بتطبيق بعض المبادئ البسيطة لتحسين الاداء وللمساعدة على تحسين Scalability المشروع

1-استخدام async,await

سيؤدي ذلك الى تحسين الاداء لانة سيستخدم ال Threading حيث سيقوم الكود باستخدام Threadمعين لاداء العملية واعادة هذا ال Thread الى ال Thread Pool حتى تنتهي العملية ، مما يحسن الاداء

```
/// <summary>
/// Get post by ID
/// </summary>
/// <param name="id">Post ID</param>
[HttpGet(template: "{id}")]
1 reference | Mohamed Dabbour, 1 hour ago | 1 author, 1 change
public async Task<ActionResult<PostDto>> GetById(int id)
{
    try
    {
        var post = await _postService.GetByIdAsync(id);
        return Ok(post);
    }
    catch (KeyNotFoundException ex)
    {
        return NotFound(ex.Message);
    }
}
```

2-استخدام Response Compression

وهي مدمجة في ال .NET. سيستخدم gzipلضغط ال Response ، مما يؤدي الى تقليل حجم ال Payload

```
1 reference | Mohamed Dabbour, 1 hour ago | 1 author, 1 change
public static IServiceCollection AddCompressionMiddleware(this IServiceCollection services)
{
    services.AddResponseCompression(options =>
    {
        options.EnableForHttps = true;
        options.Providers.Add<GzipCompressionProvider>();
        // options.Providers.Add<BrotliCompressionProvider>();

        // Add specific MIME types
        options.MimeTypes = ResponseCompressionDefaults.MimeTypes.Concat(
            new[] {
    "application/json",
    "text/json",
    //"text/plain",
    //"text/html",
    //"text/css",
    //"application/javascript",
    //"image/svg+xml"
            });
    });

    services.Configure<GzipCompressionProviderOptions>(options =>
    {
        options.Level = CompressionLevel.Optimal; // Or Optimal for better compression
    });

    return services;
```

3-استخدام ال Caching

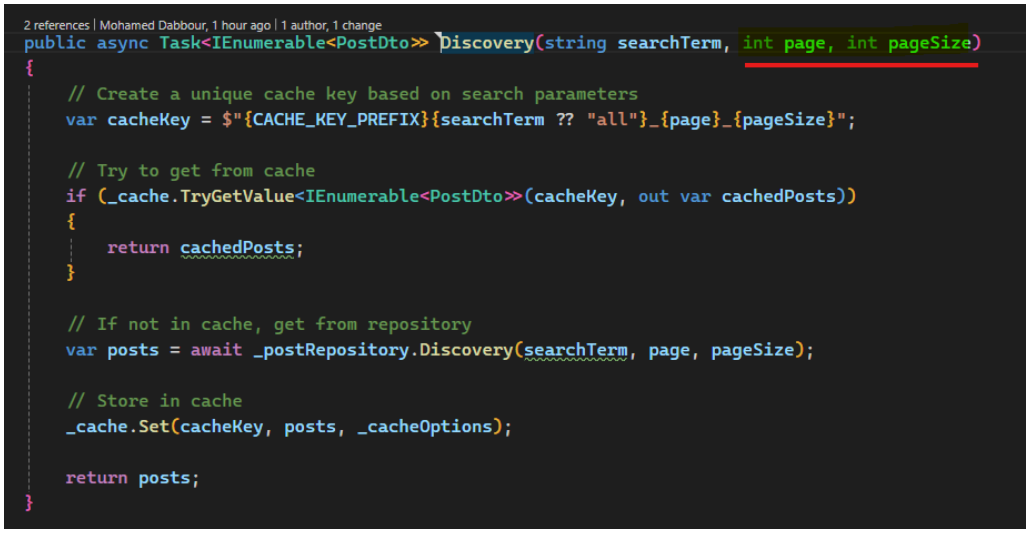
قمت باستخدام Memory Caching المدمجة في .NET ، وهي طريقة بسيطة لل Cachingلكن يجب استخدام طريقة افضل في ال Productionمثل Redis

اي Endpointبها بحث يتم حفظ نتيجة البحث في ال Cacheو استخراجها عند التكرار مما يؤدي الى تخفيف الضغط عن قاعدة البيانات

```
2 references | Mohamed Dabbour, 1 hour ago | 1 author, 1 change
public async Task<IEnumerable<PostDto>> Discovery(string searchTerm, int page, int pageSize)
{
    // Create a unique cache key based on search parameters
    var cacheKey = $"{CACHE_KEY_PREFIX}{searchTerm ?? "all"}_{page}_{pageSize}";

    // Try to get from cache
    if (_cache.TryGetValue<IEnumerable<PostDto>>(cacheKey, out var cachedPosts))
    {
        return cachedPosts;
    }

    // If not in cache, get from repository
    var posts = await _postRepository.Discovery(searchTerm, page, pageSize);

    // Store in cache
    _cache.Set(cacheKey, posts, _cacheOptions);

    return posts;
}
```

4-استخدام Pagination

ببساطة اي نتيجة بحث بها Pagination ، لا نريد ان نجلب جميع المقالات للمستخدم كل مرة



الامان

قمت باستخدام JWT لتأمين كل ال Endpoints لكن للتسهيل قمت بفتح Discovery Endpoint بدون الحاجة الى Login