تغییرات جدید

1 – استفاده از OpenTelemntry و متریک های دیفالت

`Microsoft.AspNetCore.Hosting`

`Microsoft.AspNetCore.Server.Kestre`

و ایجاد متریک های custom با نام :

1 – `CustomExceptionMeter`

2- `SuccessfulApiMeter`

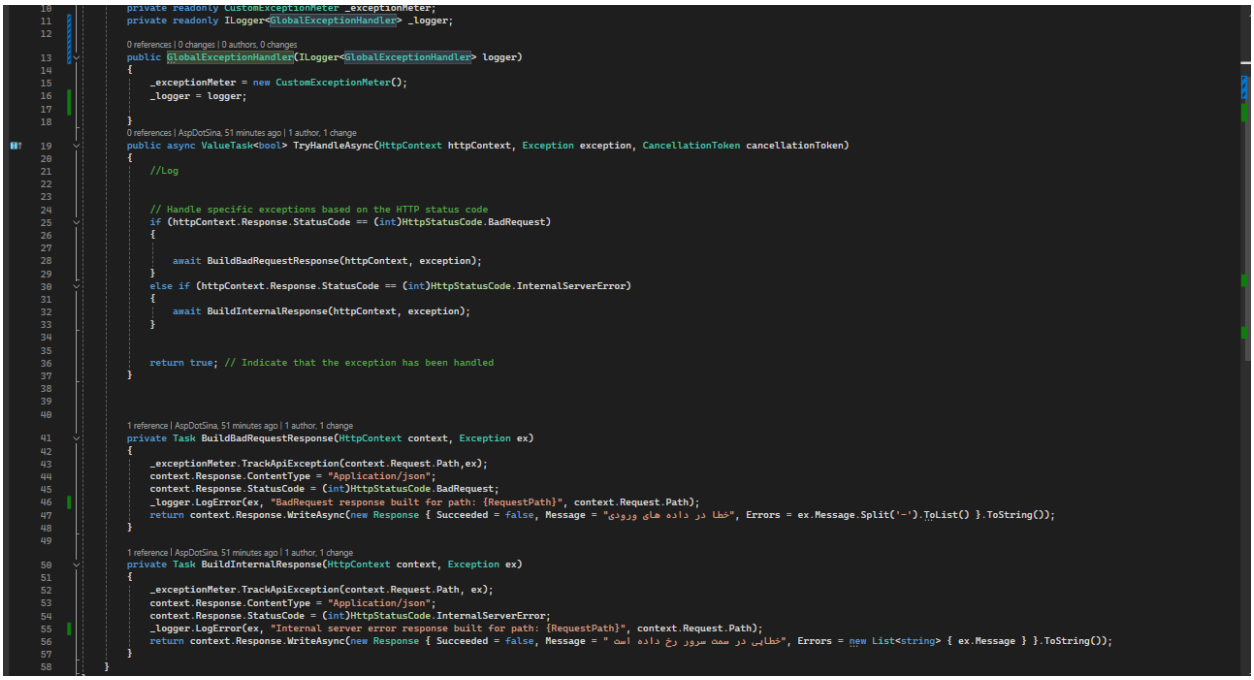
---

2 – استفاده از InMemory Cache برای Query Request

با استفاده از Mediatar PiplineBehavior

```
7
8   namespace EShop.Application.Abstractions.Behaviors
9   {
        1 reference | AspDotSina, 44 minutes ago | 1 author, 1 change
10      internal sealed class QueryCachingPipelineBehavior<TRequest, TResponse>(ICacheService cacheService) : IPipelineBehavior<TRequest, TResponse>
11          where TRequest : ICachedQuery
12      {
13          private readonly ICacheService _cacheService = cacheService;
14
            0 references | AspDotSina, 44 minutes ago | 1 author, 1 change
15          public async Task<TResponse> Handle(TRequest request, RequestHandlerDelegate<TResponse> next, CancellationToken cancellationToken)
16          {
17              return await _cacheService.GetorCreateAsync(request.Key, _ => next(),cancellationToken);
18          }
19      }
20  }
```

3 – پیاده سازی `GlobalExceptionHandler` جای کاستوم میدلویر



```
    private readonly CustomExceptionMeter _exceptionMeter;
    private readonly ILogger<GlobalExceptionHandler> _logger;

    0 references | 0 changes | 0 authors, 0 changes
    public GlobalExceptionHandler(ILogger<GlobalExceptionHandler> logger)
    {
        _exceptionMeter = new CustomExceptionMeter();
        _logger = logger;

    }
    0 references | AspDotSina, 51 minutes ago | 1 author, 1 change
    public async ValueTask<bool> TryHandleAsync(HttpContext httpContext, Exception exception, CancellationToken cancellationToken)
    {
        //Log


        // Handle specific exceptions based on the HTTP status code
        if (httpContext.Response.StatusCode == (int)HttpStatusCode.BadRequest)
        {

            await BuildBadRequestResponse(httpContext, exception);
        }
        else if (httpContext.Response.StatusCode == (int)HttpStatusCode.InternalServerError)
        {
            await BuildInternalResponse(httpContext, exception);
        }


        return true; // Indicate that the exception has been handled
    }



    1 reference | AspDotSina, 51 minutes ago | 1 author, 1 change
    private Task BuildBadRequestResponse(HttpContext context, Exception ex)
    {
        _exceptionMeter.TrackApiException(context.Request.Path,ex);
        context.Response.ContentType = "Application/json";
        context.Response.StatusCode = (int)HttpStatusCode.BadRequest;
        _logger.LogError(ex, "BadRequest response built for path: {RequestPath}", context.Request.Path);
        return context.Response.WriteAsync(new Response { Succeeded = false, Message = "خطا در داده های ورودی", Errors = ex.Message.Split('-').ToList() }.ToString());
    }

    1 reference | AspDotSina, 51 minutes ago | 1 author, 1 change
    private Task BuildInternalResponse(HttpContext context, Exception ex)
    {
        _exceptionMeter.TrackApiException(context.Request.Path, ex);
        context.Response.ContentType = "Application/json";
        context.Response.StatusCode = (int)HttpStatusCode.InternalServerError;
        _logger.LogError(ex, "Internal server error response built for path: {RequestPath}", context.Request.Path);
        return context.Response.WriteAsync(new Response { Succeeded = false, Message = "خطایی در سمت سرور رخ داده است", Errors = new List<string> { ex.Message } }.ToString());
    }
}
```

---

4 – پیاده سازی integration Test و تست سرویس های نرم افزار در لایه اپلیکیشن .