سفارشی سازی اتریبیوت authorize

برای چک کردن permission های کاربر در تمام کنترلر های api، یک BaseApiController ایجاد کردیم که در آن یک فیلد UserId موجود است و با HttpContext.Current.User.Identity.Name پر می شود.

به دلیل اینکه JWT سمت سرور logout ندارد، مجبور به ذخیره ی BlackList توکن هایی هستیم که معتبر هستند ولی چون کاربر گزینه ی logoutرا زده، دیگر نباید مورد استفاده قرار گیرند. توکن هایی که تاریخ انقضاء آنها نگذشته باشد و معتبر باشند.

برای افزایش performance از Redis استفاده کردیم. فیلد ExpireDatetime را جهت انجام task های زمانبندی شده توسط سرور و پاک کردن توکن های منقضی شده از لیست سیاه گذاشتیم.

برای این کار یک AuthorizeAttribute سفارشی میسازیم و ابتدا چک میکنیم که اگر توکن معتبر نبود، دیگر دیتابیس BlackListرا چک نکند.

```
using System;
using System.IdentityModel.Tokens.Jwt;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web;
using System.Web.Http.Controllers;
using AuthorizeAttribute = System.Web.Http.AuthorizeAttribute;
using ServiceStack.Redis;

namespace JWT_Pharmacy.Models
{
    public class JwtAuthorize : AuthorizeAttribute
    {
        JwtValidationEntities _context = new JwtValidationEntities();

        public bool IsInBlackList(string token)
        {
            var connection = new RedisClient();
            var tokenSave = connection.As<InvalidTokenViewModel>();
            var invalids = tokenSave.GetAll().FirstOrDefault(w => w.Token == token);
            return invalids != null;
        }
        public override void OnAuthorization(HttpActionContext actionContext)
        {
            var auth =
System.Web.HttpContext.Current.Request.Headers.GetValues("Authorization");
```

```
            if (auth != null)
            {
                string jwtToken = auth.First().Substring(7);
                var token = new JwtSecurityToken(jwtEncodedString: jwtToken);
                var exp = Convert.ToInt64(token.Claims.First(c => c.Type ==
"exp").Value);
                var exp_datetime = new DateTime(1970, 1, 1, 0, 0, 0).AddSeconds(exp);

                if (exp_datetime < DateTime.Now)
                {
                    HttpContext.Current.User = null;
                    actionContext.Response = new
HttpResponseMessage(HttpStatusCode.Unauthorized);
                }
                else if(this.IsInBlackList(jwtToken))
                {
                    HttpContext.Current.User = null;
                }
            }
            base.OnAuthorization(actionContext);
        }
    }
                                                                                    }
```