Бэкенд - Авторизация

Чтобы к методам контроллера был доступ только у аутентифицированных пользователей, для контроллера нужно указать атрибут `[Authorize]`. Для `GenericController` этот атрибут уже добавлен, поэтому для контроллеров-наследников его указывать не нужно. Чтобы отключить требование аутентификации для отдельного метода, нужно добавить для этого метода атрибут `[AllowAnonymous]`.

Виды авторизации
- 1 Авторизация по ролям
- 2 Авторизация по политикам
- 3 Авторизация по функциям приложения
- 4 Авторизация по функциям организаций
  - 4.1 Фильтрация выборки
  - 4.2 Точечная проверка доступа

Виды авторизации

1 Авторизация по ролям

Атрибуту `Authorize` можно передать в качестве аргумента список ролей, которым доступен контроллер или метод:

```
[Authorize(Roles = "Supervisor, AdminTN")]
public class SomeController : ControllerBase
{
    // ...
}
```

2 Авторизация по политикам

Задаются политики в файле `Startup.cs` в методе `ConfigureServices`.

Пример определения политики:

```
opts.AddPolicy(AuthPolicy.Admins,
              policy => policy.RequireClaim(ClaimTypes.Role,
                                    new[] { RoleName.Admin, RoleName.AdminTN }));
```

Пример применения политики:

```
[Authorize(Policy = AuthPolicy.Admins)]
public class AppFunctionsController : GenericController<AppFunctionVM, AppFunction>
{
    // ...
}
```

3 Авторизация по функциям приложения

Для контроллера целиком или для отдельного метода можно указать атрибуты, требующие у пользователя наличия определенной функции приложения с правом чтения или записи:

- `AuthorizeRead(params AppFunc[] appFunctions)` - необходимо право на **чтение** хотя бы для одной из указанных функций;
- `AuthorizeWrite(params AppFunc[] appFunctions)` - необходимо право на **запись** хотя бы для одной из указанных функций.

```
// Пример
[AuthorizeWrite(AppFunc.TripsPlane, AppFunc.TripsHelicopter, AppFunc.TripsBus)]
[HttpPut]
public async Task<ActionResult> UpdateTripStage(int fromDestId, TripStageUpdateVM stage)
{
    // ...
}
```

Чтобы требовать обязательного одновременного наличия нескольких функций приложения, нужно добавить несколько атрибутов:

```
// Пример
[AuthorizeRead(AppFunc.TripsPlane)]
[AuthorizeRead(AppFunc.TripsHelicopter)]
[HttpGet]
public async Task<ActionResult> GetAirStats(DateTime date)
{
    // ...
}
```

4 Авторизация по функциям организаций

4.1 Фильтрация выборки

Чтобы применить фильтр по организациям, можно воспользоваться методом `ApplyOrgFilter` из класса `UserOrgService`:

```
Task<IQueryable<T>> ApplyOrgFilter<T>(IQueryable<T> query,
                                      OrgFunc[] requiredOrgFuncs,
                                      AppFunc masterAppFunc,
                                      CancellationToken cancellationToken = default,
                                      bool writeRequired = false)
```

Данный метод фильтрует запрос `query` так, чтобы в выборке остались данные только для тех организаций, для которых у пользователя есть одна из функций `requiredOrgFuncs` (с правом на запись при `writeRequired` == `true`).

Фильтр по организацим не будет применен, если у пользователя есть мастер-функция приложения `masterAppFunc`, предоставляющая доступ ко всем строкам выборки.

Чтобы воспользоваться данным методом, в конструктор нужного сервиса необходимо добавить `UserOrgService`.

Пример:

```
public class OrderFactService : GenericService<OrderFactVM, OrderFact>
{
    private readonly UserOrgService _userOrgService;

    public OrderFactService(IUnitOfWork unitOfWork, IMapper mapper, UserOrgService userOrgService) : base(unitOfWork, mapper)
    {
        _userOrgService = userOrgService;
    }

    // ...

    public async Task<IQueryable<OrderFactVM>> GetAll(CancellationToken cancellationToken = default)
    {
        IQueryable<OrderFact> orders = _unitOfWork.OrderFactRepo.GetAll();
```

```
        // Вернуть только заявки доступных пользователю организаций
        orders = await _userOrgService.ApplyOrgFilter(
                                        orders,
                                        new[] { OrgFunc.OrderFact },
                                        AppFunc.OrderFactIgnoreOrgFilter,
                                        cancellationToken).ConfigureAwait(false);

        return _mapper.ProjectTo<OrderFactVM>(orders);
    }
}
```

4.2 Точечная проверка доступа

Чтобы точечно проверить, доступна ли пользователю указанная организация, можно воспользоваться методом `IsOrgAccessible` из класса `UserOrgService`:

```
Task<bool> IsOrgAccessible(int orgId,
                           OrgFunc[] requiredOrgFuncs,
                           AppFunc masterAppFunc,
                           bool writeRequired = false)
```

Данный метод возвращает `true`, если у пользователя есть одна из функций `requiredOrgFuncs` (с правом на запись при `writeRequired` == `true`) или если у пользователя есть мастер-функция приложения `masterAppFunc`, предоставляющая доступ ко всем организациям.

Пример:

```
public override async Task<OrderFactVM> Create(OrderFactVM viewModel)
{
    if (!await _userOrgService.IsOrgAccessible(viewModel.OrganizationId,
                                               new[] { OrgFunc.OrderFact },
                                               AppFunc.OrderFactIgnoreOrgFilter,
                                               true).ConfigureAwait(false))
    {
        throw new UnauthorizedAccessException("Нет доступа к заявкам данной организации");
    }

    viewModel.DateRecord = DateTime.Now;
    return await base.Create(viewModel).ConfigureAwait(false);
}
```