Identity

(Demo prep)

- Delete Identity collection from Postman
- Ensure there's at least one registered user

Adding the Users REST API

So far we can register and sign in users into our Identity microservice via the ASP.NET Core Identity default UI pages. However, we should also provide a REST API that other parts of our Play Economy system can later use to interact with our users database.

So let's add a Users REST API.

1. Add **Dtos.cs** at the root:

```
namespace Play.Identity.Service.Dtos
{
    public record UserDto(
            Guid Id,
            string UserName,
            string Email,
            decimal Gil,
            DateTimeOffset CreatedDate);

    public record UpdateUserDto(
            [Required][EmailAddress] string Email,
            [Range(0, 1000000)] decimal Gil);
}
```

2. Add **Extensions.cs** at the root:

```
namespace Play.Identity.Service
{
    public static class Extensions
    {
        public static UserDto AsDto(this ApplicationUser user)
        {
            return new UserDto(user.Id, user.UserName, user.Email, user.Gil, user.CreatedOn);
        }
    }
}
```

3. Add **UsersController.cs (and collapse explorer)**:

```
namespace Play.Identity.Service.Controllers
{
    [ApiController]
    [Route("users")]
    public class UsersController : ControllerBase
    {
        private readonly UserManager<ApplicationUser> userManager;

        public UsersController(UserManager<ApplicationUser> userManager)
        {
            this.userManager = userManager;
        }

        [HttpGet]
        public ActionResult<IEnumerable<UserDto>> Get()
        {
            var users = userManager.Users
                .ToList()
                .Select(user => user.AsDto());
            return Ok(users);
        }


        [HttpGet("{id}")]
        public async Task<ActionResult<UserDto>> GetByIdAsync(Guid id)
        {
            var user = await userManager.FindByIdAsync(id.ToString());

            if (user == null)
            {
                return NotFound();
            }

            return user.AsDto();
        }

        [HttpPut("{id}")]
        public async Task<IActionResult> PutAsync(Guid id, UpdateUserDto userDto)
        {
            var user = await userManager.FindByIdAsync(id.ToString());

            if (user == null)
            {
                return NotFound();
            }

            user.UserName = userDto.Email;
            user.Email = userDto.Email;
            user.Gil = userDto.Gil;
```

```
            await userManager.UpdateAsync(user);

            return NoContent();
        }

        [HttpDelete("{id}")]
        public async Task<IActionResult> DeleteAsync(Guid id)
        {
            var user = await userManager.FindByIdAsync(id.ToString());

            if (user == null)
            {
                return NotFound();
            }

            await userManager.DeleteAsync(user);

            return NoContent();
        }
    }
}
```

4. Start web server

5. Import to Postman:
   https://localhost:5003/swagger/v1/swagger.json

6. Set base url for Identity collection to https://localhost:5003

7. Test all APIs

With that, our integration with ASP.NET Core Identity is mostly complete.

In the next module we will learn about the multiple patterns and techniques you can use today to secure your microservices.