Code Academy

Lecturer

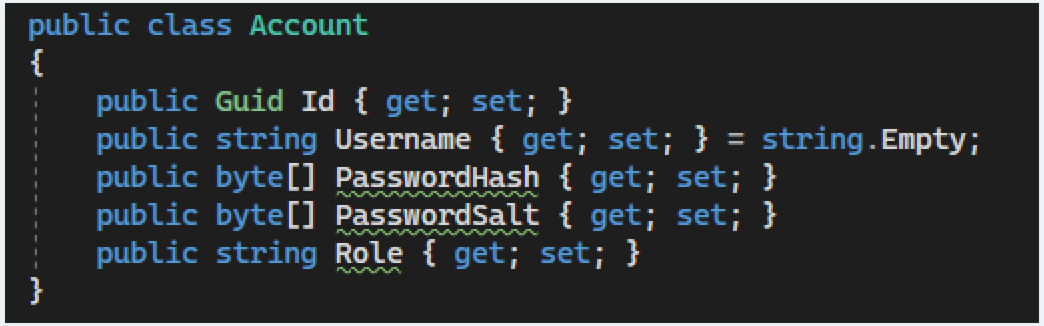
Rokas Slaboševičius

JWT roles

Data

Let's add the Role property

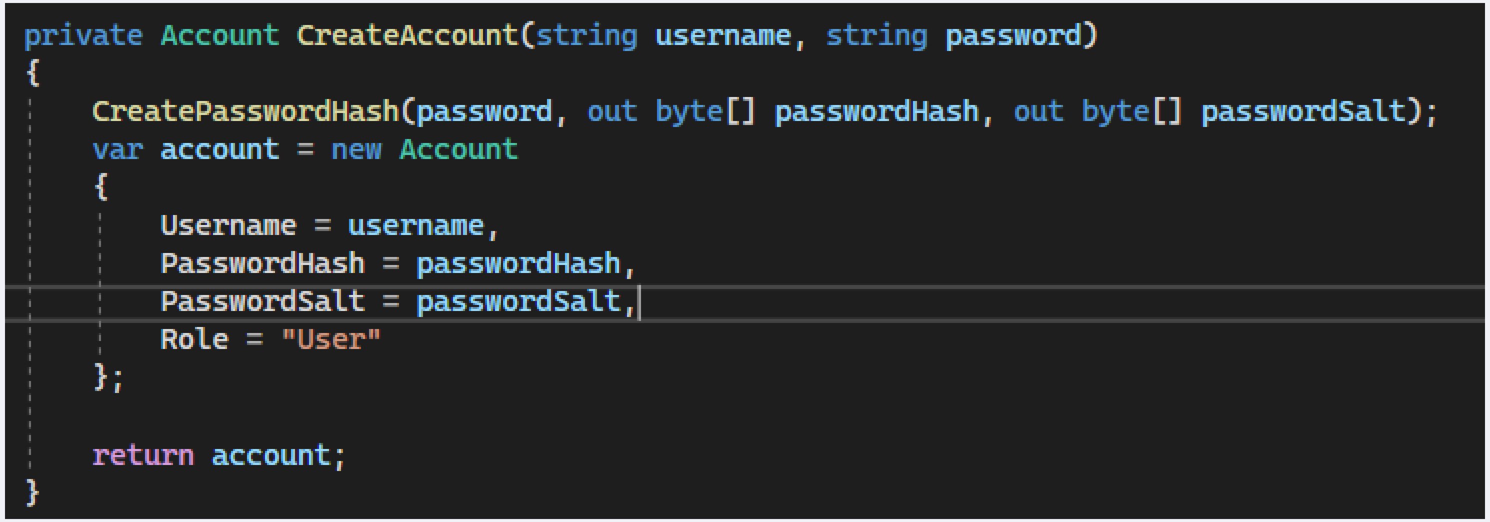
Not forgetting migration



```
public class Account
{
    public Guid Id { get; set; }
    public string Username { get; set; } = string.Empty;
    public byte[] PasswordHash { get; set; }
    public byte[] PasswordSalt { get; set; }
    public string Role { get; set; }
}
```

Let's add the Role property

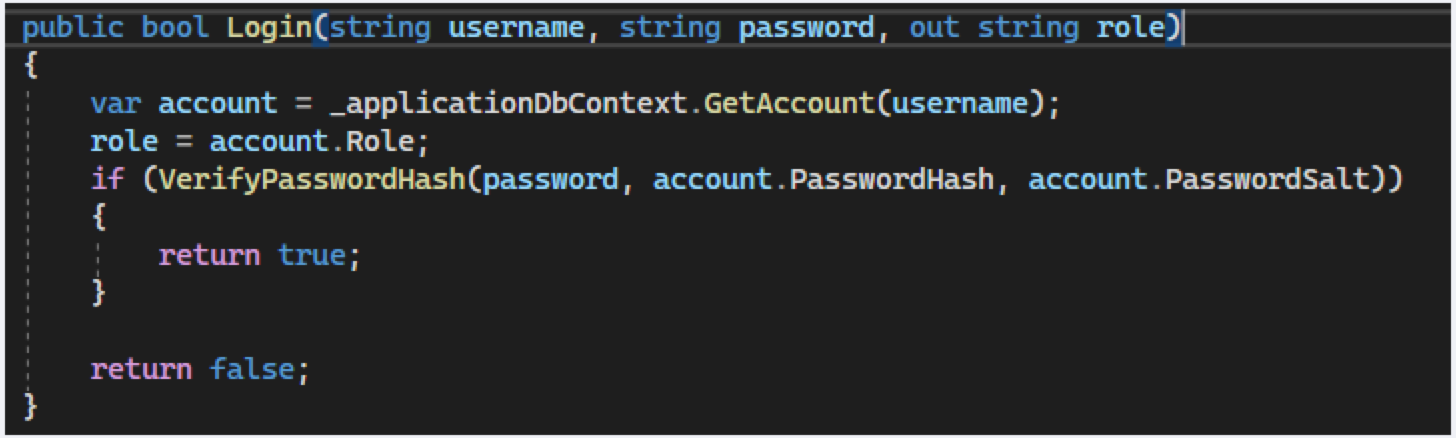
Assign the default role "User"



```
private Account CreateAccount(string username, string password)
{
    CreatePasswordHash(password, out byte[] passwordHash, out byte[] passwordSalt);
    var account = new Account
    {
        Username = username,
        PasswordHash = passwordHash,
        PasswordSalt = passwordSalt,
        Role = "User"
    };

    return account;
}
```

Let's add the Role property

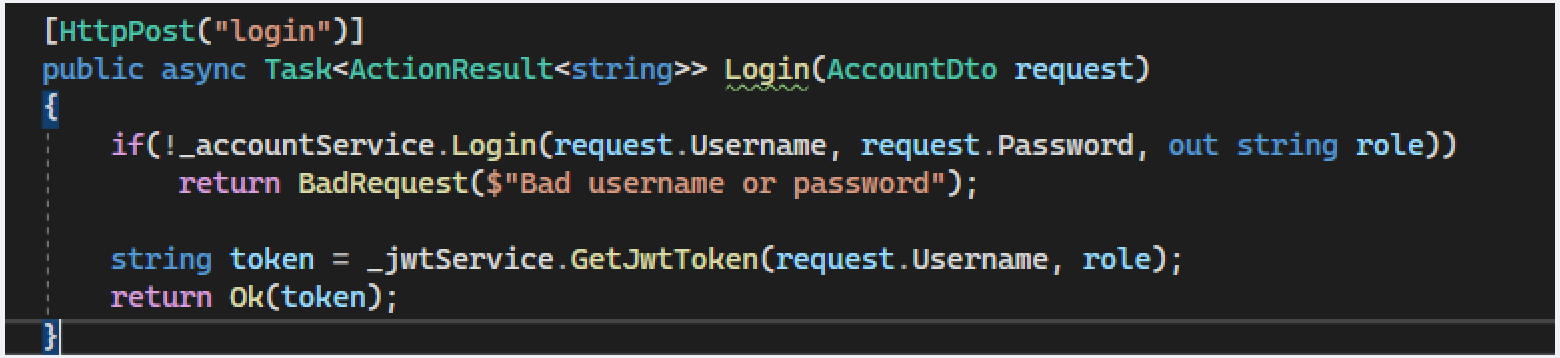
We create a role receipt on login



```
public bool Login(string username, string password, out string role)
{
    var account = _applicationDbContext.GetAccount(username);
    role = account.Role;
    if (VerifyPasswordHash(password, account.PasswordHash, account.PasswordSalt))
    {
        return true;
    }

    return false;
}
```

Let's add the Role property

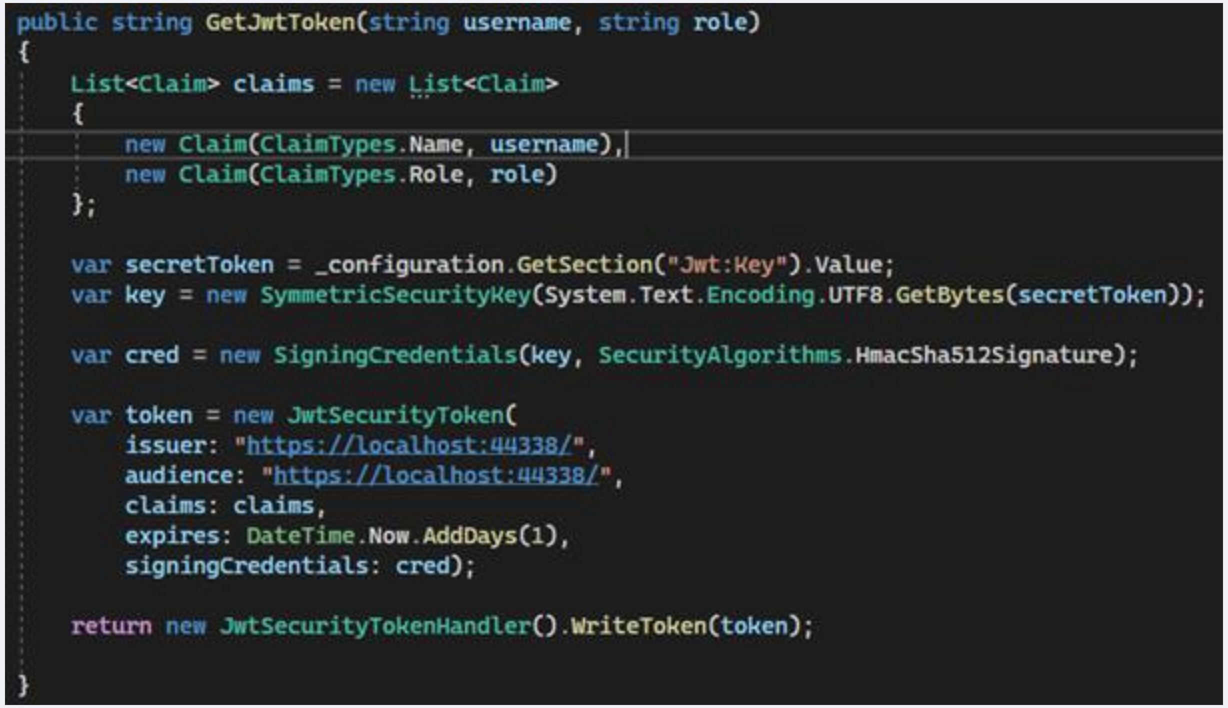
Transferring the role to the Jwt service



```
[HttpPost("login")]
public async Task<ActionResult<string>> Login(AccountDto request)
{
    if(!_accountService.Login(request.Username, request.Password, out string role))
        return BadRequest($"Bad username or password");

    string token = _jwtService.GetJwtToken(request.Username, role);
    return Ok(token);
}
```

Let's add the Role property

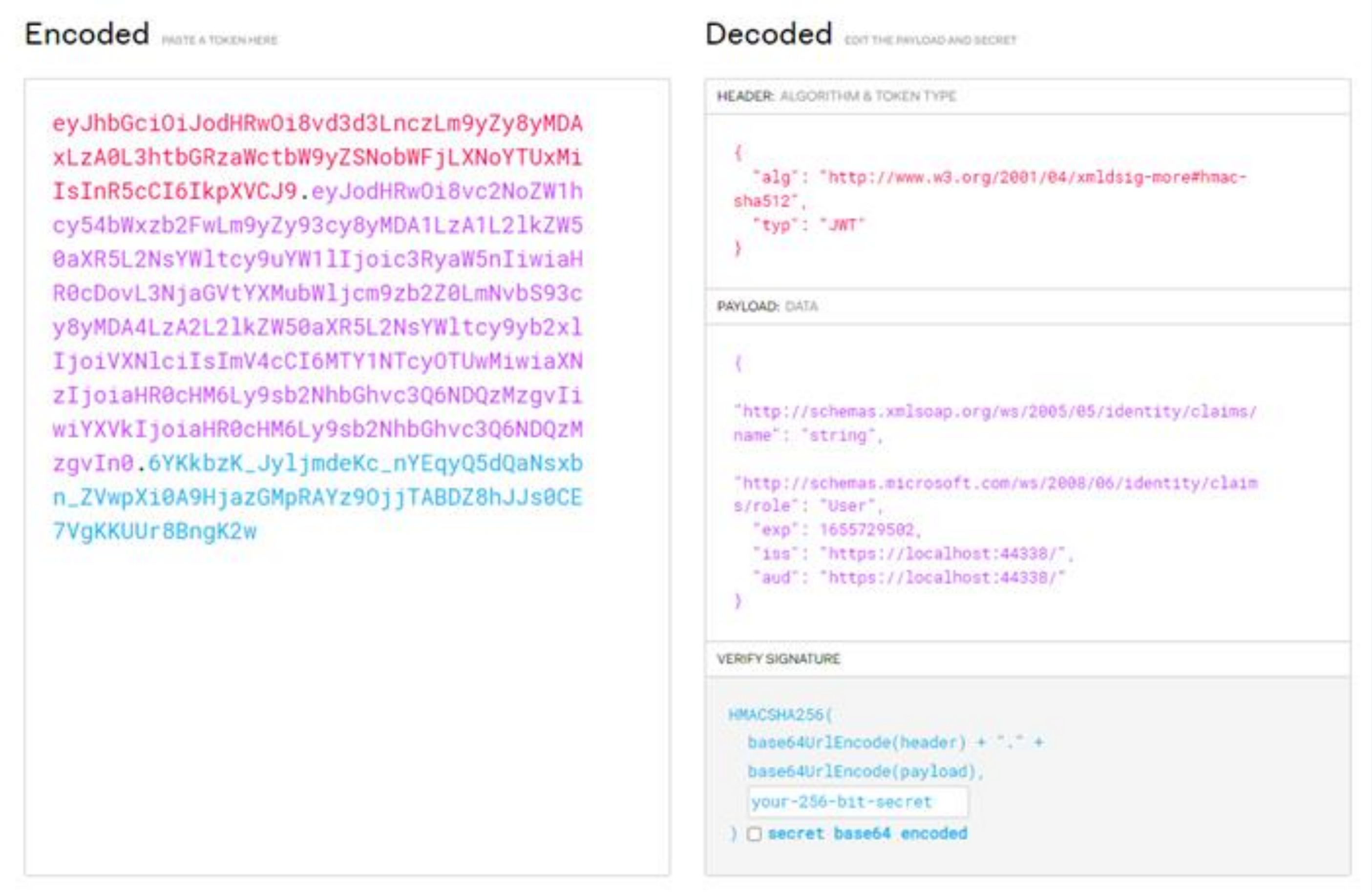
Assign role as Claim



```
public string GetJwtToken(string username, string role)
{
    List<Claim> claims = new List<Claim>
    {
        new Claim(ClaimTypes.Name, username),
        new Claim(ClaimTypes.Role, role)
    };

    var secretToken = _configuration.GetSection("Jwt:Key").Value;
    var key = new SymmetricSecurityKey(System.Text.Encoding.UTF8.GetBytes(secretToken));

    var cred = new SigningCredentials(key, SecurityAlgorithms.HmacSha512Signature);

    var token = new JwtSecurityToken(
        issuer: "https://localhost:44338/",
        audience: "https://localhost:44338/",
        claims: claims,
        expires: DateTime.Now.AddDays(1),
        signingCredentials: cred);

    return new JwtSecurityTokenHandler().WriteToken(token);

}
```

Let's add the Role property

In the newly generated token we can see the role



```
eyJhbGciOiJodHRwOi8vd3d3LnczLm9yZy8yMDA
xLzA0L3htbGRzaWctbW9yZSNobWFjLXNoYTUxMi
IsInR5cCI6IkpXVCJ9.eyJodHRwOi8vc2NoZW1h
cy54bWxzb2FwLm9yZy93cy8yMDA1LzA1L2lkZW5
0aXR5L2NsYWltcy9uYW1lIjoic3RyaW5nIiwiaH
R0cDovL3NjaGVtYXMubWljcm9zb2Z0LmNvbS93c
y8yMDA4LzA2L2lkZW50aXR5L2NsYWltcy9yb2xl
IjoiVXNlciIsImV4cCI6MTY1NTcyOTUwMiwiaXN
zIjoiaHR0cHM6Ly9sb2NhbGhvc3Q6NDQzMzgvIi
wiYXVkIjoiaHR0cHM6Ly9sb2NhbGhvc3Q6NDQzM
zgvIn0.6YKkbzK_JyljmdeKc_nYEqyQ5dQaNsxb
n_ZVwpXi0A9HjazGMpRAYz9OjjTABDZ8hJJs0CE
7VgKKUUr8BngK2w
```

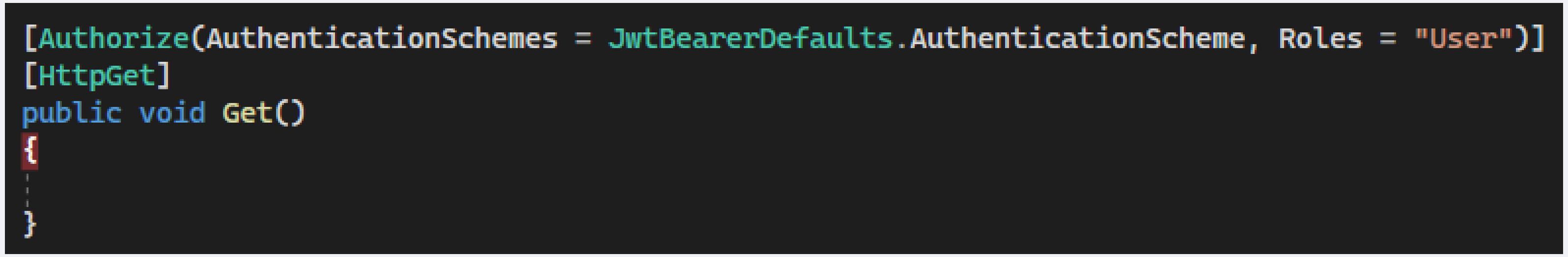
```
{
  "alg": "http://www.w3.org/2001/04/xmldsig-more#hmac-
sha512",
  "typ": "JWT"
}
```

```
{

"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/
name": "string",

"http://schemas.microsoft.com/ws/2008/06/identity/claim
s/role": "User",
  "exp": 1655729502,
  "iss": "https://localhost:44338/",
  "aud": "https://localhost:44338/"
}
```

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) □ secret base64 encoded
```

Let's add the Role property

Now we can create controllers with authorisations



```
[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme, Roles = "User")]
[HttpGet]
public void Get()
{

}
```

Task 1

- Add roles to the API of the last lecture, make some functionality available only to users with the ADMIN role, for first time edit record in the database to assign admin role, and then they can make other admins admins via a separate admin endpoint, and another endpoint that will remove the admin role (revert to 'user').