🧠 Zadanie: Rozszerzenie transformacji poświadczeń

📋 Kontekst biznesowy

System korzysta z logowania przez Google (OAuth2 / OpenID Connect). Po zalogowaniu użytkownik ma przypisany tylko podstawowy zestaw claimów (np. adres e-mail). Aby rozszerzyć tożsamość użytkownika o dane wymagane przez system wewnętrzny, wykorzystywane są klasy implementujące `IClaimsTransformation`. Dotychczas system automatycznie dodaje numer PESEL użytkownika na podstawie jego adresu e-mail, korzystając z lokalnej bazy danych.

🆕 **Nowe wymagania biznesowe**:

Klient zgłosił potrzebę rozszerzenia informacji o użytkowniku o dodatkowe dane:

- zestaw uprawnień użytkownika (np. `CanSendOffer`, `CanRemoveOrder`, `CanPrint`),
- nazwę organizacji i działu, w którym pracuje (np. `"Atari Corp"`, `"Sales"`).

Zależy mu na tym, aby rozwiązanie było elastyczne i pozwalało łatwo dodawać kolejne transformacje w przyszłości.

🔧 Obecny kod

```
public class PeselClaimsTransformation : IClaimsTransformation
{
    private readonly IUserRepository _userRepository;

    public PeselClaimsTransformation(IUserRepository userRepository)
    {
        _userRepository = userRepository;
    }

    public async Task<ClaimsPrincipal> TransformAsync(ClaimsPrincipal
principal)
    {
        var email = principal.FindFirst(ClaimTypes.Email)?.Value;

        if (email != null && !principal.HasClaim(c => c.Type == "pesel"))
        {
            var pesel = await _userRepository.GetPeselByEmailAsync(email);
            if (!string.IsNullOrWhiteSpace(pesel))
            {
```

```
                ((ClaimsIdentity)principal.Identity).AddClaim(new
Claim("pesel", pesel));
            }
        }

        return principal;
    }
}
```

🎯 Zadanie

1. Zaimplementuj nowe transformacje: `PermissionsClaimsTransformation`, `OrganizationClaimsTransformation`.
2. Przygotuj rozwiązanie zgodne z zasadą **pojedyńczej odpowiedzialności** (Single Responsibility Principle).

🧪 Oczekiwany wynik (`GET /api/user`)

Dla użytkownika `user@example.com`, po zalogowaniu przez Google i przetworzeniu claimów:

```
[
  { "type": "email", "value": "user@example.com" },
  { "type": "pesel", "value": "85010212345" },
  { "type": "permission", "value": "CanSendOffer" },
  { "type": "permission", "value": "CanRemoveOrder" },
  { "type": "permission", "value": "CanPrint" },
  { "type": "organization", "value": "Atari Corp" },
  { "type": "department", "value": "Sales" }
]
```

💡 Wskazówka: Kontroler do podglądu wyników

Aby sprawdzić wyniki działania transformacji, możesz utworzyć kontroler API, który pozwoli na podgląd aktualnych roszczeń użytkownika w systemie.

```
[ApiController]
[Route("api/[controller]")]
public class UserController : ControllerBase
{
    private readonly IHttpContextAccessor _accessor;
```

```
    public UserController(IHttpContextAccessor accessor)
    {
        _accessor = accessor;
    }

    [HttpGet]
    public IActionResult Get()
    {
        var claims = _accessor.HttpContext.User.Claims.Select(c => new {
c.Type, c.Value });
        return Ok(claims);
    }
}
```