Final Module
Summary and Cloud Deployment

דגשים לפרוייקט מסכם
חזרה על נושא API
העלאה לענן

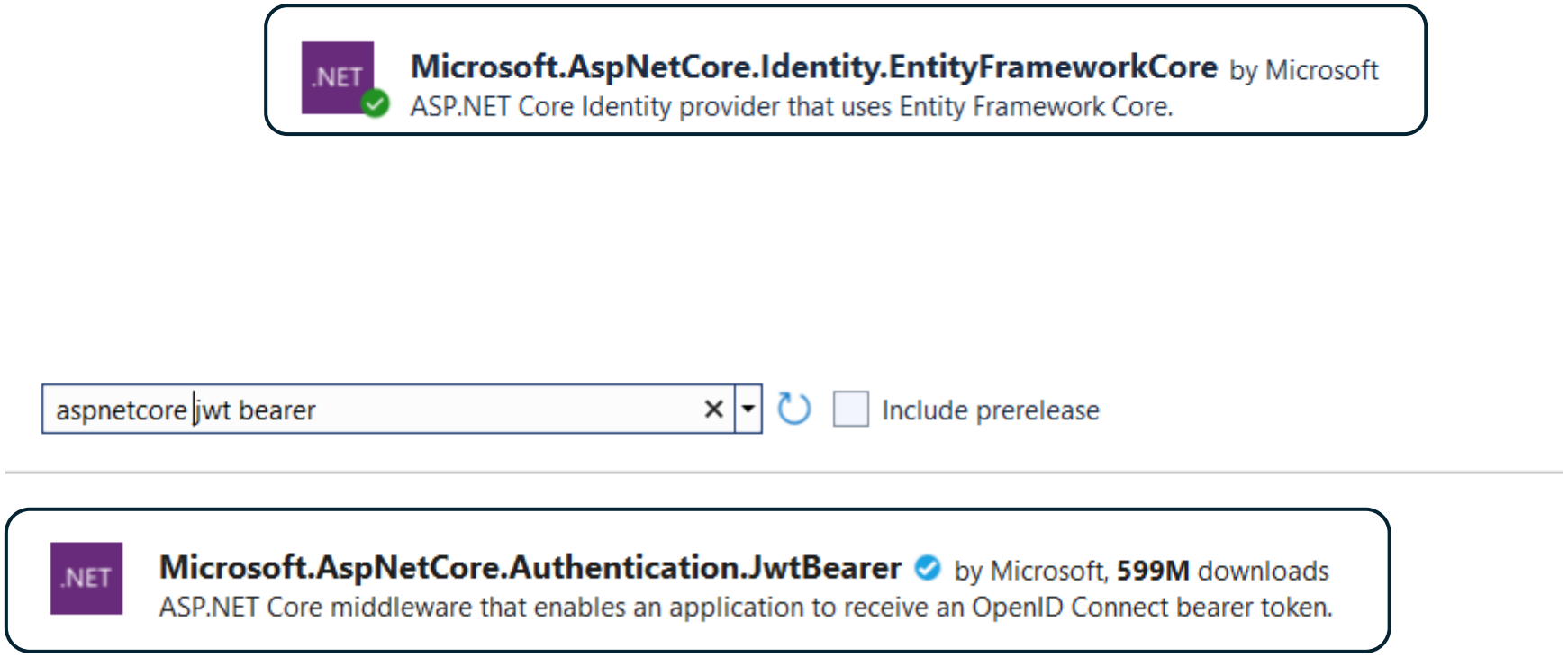
TomerBu

Final Module
Summary and Cloud Deployment

עבודה עם Identity

TomerBu

ספריות לעבודה עם Identity



מודל למשתמש – יורש מהמחלקה המובנית IdentityUser

```
namespace DAL.Models;

using Microsoft.AspNetCore.Identity;

public class AppUser:IdentityUser<int>
{
}
```

DTO להרשמה

```
namespace FinalAPI.DTOs;

using System.ComponentModel.DataAnnotations;
public class RegisterDto
{
    [Required]
    [EmailAddress]
    [Display(Name = "Email")]
    public required string Email { get; set; }

    [Required]
    [MinLength(2), MaxLength(20)]
    [Display(Name = "Username")]
    public required string Username { get; set; }

    [Required]
    [DataType(DataType.Password)]
    [Display(Name = "Password")]
    public required string Password { get; set; }
}
```

DTO להתחברות

```
using System.ComponentModel.DataAnnotations;

namespace FinalAPI.DTOs;

public class LoginDto
{
    [Required, EmailAddress]
    public required string Email { get; set; }
    [Required, MinLength(2), MaxLength(40)]
    public required string Password { get; set; }
}
```

Database Context – Entity Framework:

```
using Microsoft.EntityFrameworkCore;
using DAL.Models;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;

namespace DAL.Data;

public class ContextDAL(DbContextOptions<ContextDAL> options)
    : IdentityDbContext<AppUser, IdentityRole<int>, int>(options)
{
    public DbSet<Category> Categories { get; set; } = default!;
    public DbSet<Product> Products { get; set; } = default!;

//…
}
```

הגדרות בקובץ Program.cs

```
// Add services to the container.
var connectionString = builder.Configuration.GetConnectionString("DefaultConnection") ??
throw new InvalidOperationException("Connection string 'DefaultConnection' not found.");
builder.Services.AddDbContext<ApplicationDbContext>(options =>
    options.UseSqlServer(connectionString));

builder.Services.AddIdentity<AppUser, IdentityRole<int>>(options =>
{
    //options.SignIn.RequireConfirmedAccount = true;
    options.User.RequireUniqueEmail = true;
    options.Password.RequiredLength = 8;
})
    .AddEntityFrameworkStores<ContextDAL>();
```

1

```
app.UseRouting();

app.UseAuthentication();
app.UseAuthorization();

app.MapControllerRoute(
    name: "default",
    pattern: "{area=Home}/{controller=Home}/{action=Index}/{id?}");
//app.MapRazorPages();

app.Run();
```

2

```
//who are you
app.UseAuthentication();
//are you allowed
app.UseAuthorization();
```

Database Context – Entity Framework: Package Manager Console

Add-Migration Auth

1

Update-Database

2

ניצור קונטרולר להרשמה/התחברות/התנתקות/הצגת פרטים
AuthController

```
using DAL.Models;
using FinalAPI.DTOs;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;

namespace FinalAPI.Controllers;

[Route("api/[controller]")]
[ApiController]

public class AuthController(
    UserManager<AppUser> userManager, SignInManager<AppUser> signInManager) : ControllerBase
{
    [HttpPost("register")]
    public async Task<IActionResult> Register([FromBody] RegisterDto dto)
    {
        if (ModelState.IsValid)
        {
            var user = new AppUser { UserName = dto.Username, Email = dto.Email };
            var result = await userManager.CreateAsync(user, dto.Password);
            if (result.Succeeded)
            {
                return Ok();
            }
            return BadRequest(result.Errors);
        }
        return BadRequest(ModelState);
    }
}
```

הרשמה – פעולת דטה-בייס פשוטה
רק נזכור שהסיסמא מוצפנת ע"י הספריה
וכל היתר זה שמירה בדטה-בייס

ונבדוק עם SWAGGER
שהכל עובד תקין

```
[HttpPost("login")]
public async Task<IActionResult> Login([FromBody] LoginDto dto)
{
    if (ModelState.IsValid)
    {
        var user = await userManager.FindByEmailAsync(dto.Email);
        if (user != null)
        {
            var result = await signInManager.PasswordSignInAsync(
                user.UserName, dto.Password, false, lockoutOnFailure: false
                );
            if (result.Succeeded)
            {

                return Ok("Cool");
            }
            return Unauthorized();
        }
        return Unauthorized();
    }
    return BadRequest(ModelState);
}
```

אין צורך בעוגיה

פעולת התחברות
AuthController
חלק ראשון

אחרי התחברות מוצלחת – נרצה
לתת JWT למשתמש ולזהות אותו
באמצעות JWT
(בעמודים הבאים)

הנפקה של JWT
הקובץ AppSettings.JSON

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "ContextDAL": "Server=DESKTOP-
VADFGDK;Database=FinalProjectDB;TrustServerCertificate=
True;Trusted_Connection=True;MultipleActiveResultSets=t
rue"
  },
  "JwtSettings": {
    "Issuer": "YourIssuer",
    "Audience": "YourAudience",
    "SecretKey": "YourLongLongSecretHere"
  }
}
```

הנפקה של JWT

```
namespace FinalAPI.Services;

public class JwtTokenService(IConfiguration configuration, UserManager<AppUser> userManager)
{
    public async Task<string> CreateToken(AppUser user)
    {
        var jwtSettings = configuration.GetSection("JwtSettings");
        var secretKey = jwtSettings["SecretKey"] ?? throw new Exception("Set Secret");
        var claims = new List<Claim> {
            new Claim(JwtRegisteredClaimNames.Sub, user.UserName),
        };
        var isAdmin = await userManager.IsInRoleAsync(user, "Admin");
        if (isAdmin)
        {
           // claims.Add(new Claim("isAdmin", "true"));
            claims.Add(
                 new Claim(ClaimTypes.Role, "admin")
            );
        }

        var key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(secretKey));
        var creds = new SigningCredentials(key, SecurityAlgorithms.HmacSha512Signature);
        var token = new JwtSecurityToken(
          issuer: jwtSettings["Issuer"],
          audience: jwtSettings["Audience"],
          claims: claims,
          expires: DateTime.Now.AddMinutes(30),
          signingCredentials: creds
        );
        return new JwtSecurityTokenHandler().WriteToken(token);
    }
}
```

הנפקה של JWT

הרשמה של השירות שלנו בקובץ
Program.cs

```
// Configure JWT authentication
//our small service for generating jwt tokens:
builder.Services.AddScoped<JwtTokenService>();
```

JwtTokenService jwtTokenService

נוסיף הזרקה בבנאי של:

1

שימוש בAuthController
פעולת התחברות:

```
[HttpPost("login")]
public async Task<IActionResult> Login([FromBody] LoginDto dto)
{
    if (ModelState.IsValid)
    {
        var user = await userManager.FindByEmailAsync(dto.Email);
        if (user != null)
        {
            var result = await signInManager.PasswordSignInAsync(
                user.UserName, dto.Password, true, lockoutOnFailure: false
                );
            if (result.Succeeded)
            {
                var token = await jwtTokenService.CreateToken(user);
                return Ok(new { token });
            }
            return Unauthorized();
        }
        return Unauthorized();
    }
    return BadRequest(ModelState);
}
```

2

נבדוק עם Postman
או עם Swagger
ונראה שאנחנו מקבלים JWT

זריעה של משתמשים ותפקידים בדטה-בייס:
DalContext.cs

```
var hasher = new PasswordHasher<AppUser>();
modelBuilder.Entity<IdentityRole<int>>().HasData(
    new IdentityRole<int>()
    {
        Id = 1,
        Name = "admin",
        NormalizedName = "ADMIN",
        ConcurrencyStamp = Guid.NewGuid().ToString()
    },
    new IdentityRole<int>()
    {
        Id = 2,
        Name = "user",
        NormalizedName = "USER",
        ConcurrencyStamp = Guid.NewGuid().ToString()
    });
```

זריעה של משתמשים ותפקידים בדטה-בייס:

DalContext.cs

```
var hasher = new PasswordHasher<AppUser>();
modelBuilder.Entity<AppUser>().HasData(
    new AppUser()
    {
        Id = 1,
        Email = "TomerBu@gmail.com",
        NormalizedEmail = "TOMERBU@GMAIL.COM",
        UserName = "TomerBu",
        NormalizedUserName = "TOMERBU",
        SecurityStamp = Guid.NewGuid().ToString(),
        PasswordHash = hasher.HashPassword(null, "123456")
    },
    new AppUser()
    {
        Id = 2,
        Email = "user@example.com",
        NormalizedEmail = "USER@EXAMPLE.COM",
        UserName = "user",
        NormalizedUserName = "USER",
        SecurityStamp = Guid.NewGuid().ToString(),
        PasswordHash = hasher.HashPassword(null, "123456")
    }
    );
```

זריעה של משתמשים ותפקידים בדטה-בייס:
DalContext.cs

```
modelBuilder.Entity<IdentityUserRole<int>>().HasData(
    new IdentityUserRole<int>()
    {
        RoleId = 1,
        UserId = 1,

    },
    new IdentityUserRole<int>()
    {
        RoleId = 2,
        UserId = 2
    }
    );
```

נבצע מיגרציה:
Add-Migration SeedUsers
Update-database

שיעורי בית:

מומלץ לחזור על המצגת ולנסות לממש בעצמכם את כל מה שמימשנו בכיתה בפרוייקט חדש לטובת התרגול.

כמו תמיד - מוזמנים להגיע לשיעור הבא עם שאלות שעלו לכם במהלך התרגול.