**Звіт з реалізації аутентифікації та авторизації для проекту LambdaChess**

**Виконали**

Бенько Володимир  
Бучко Роман  
Долуда Андрій  
Зозуля Олег

**1. Підключення ASP.NET Identity до проекту**

**Поточний стан**

Аналіз файлової структури проекту показує, що ASP.NET Identity вже інтегрований у проект LambdaChess. Наявність наступних компонентів підтверджує це:

* Підключені пакети:
  + Microsoft.AspNetCore.Identity.EntityFrameworkCore
  + Microsoft.AspNetCore.Identity.UI
* Структура файлів:
  + Директорія Areas/Identity/Pages для сторінок Identity
  + Файл \_LoginPartial.cshtml в директорії Views/Shared
  + Міграції в директорії Migrations містять схему бази даних для Identity

**Рекомендації для повної інтеграції**

1. **Переконатися в наявності реєстрації сервісів Identity у Program.cs**:
2. builder.Services.AddDefaultIdentity<ApplicationUser>(options =>
3. options.SignIn.RequireConfirmedAccount = true)
4. .AddRoles<IdentityRole>()
5. .AddEntityFrameworkStores<ApplicationDbContext>();
6. **Додати налаштування Identity**:
7. builder.Services.Configure<IdentityOptions>(options =>
8. {
9. // Налаштування паролю
10. options.Password.RequireDigit = true;
11. options.Password.RequireLowercase = true;
12. options.Password.RequireNonAlphanumeric = true;
13. options.Password.RequireUppercase = true;
14. options.Password.RequiredLength = 8;
16. // Налаштування блокування
17. options.Lockout.DefaultLockoutTimeSpan = TimeSpan.FromMinutes(15);
18. options.Lockout.MaxFailedAccessAttempts = 5;
20. // Налаштування користувача
21. options.User.RequireUniqueEmail = true;
22. });

**2. Заповнення бази даних ролями відповідно до документації**

**Реалізація**

Потрібно створити метод ініціалізації ролей, який буде виконуватись при старті додатку:

public static async Task InitializeRoles(IServiceProvider serviceProvider)

{

var roleManager = serviceProvider.GetRequiredService<RoleManager<IdentityRole>>();

string[] roleNames = { "Admin", "Player", "Moderator" };

foreach (var roleName in roleNames)

{

var roleExist = await roleManager.RoleExistsAsync(roleName);

if (!roleExist)

{

await roleManager.CreateAsync(new IdentityRole(roleName));

}

}

}

Виклик цього методу потрібно додати в Program.cs:

// В кінці файлу Program.cs

using (var scope = app.Services.CreateScope())

{

var services = scope.ServiceProvider;

await InitializeRoles(services);

}

**3. Розширення моделі користувача із ASP.NET Identity**

**Реалізація**

Розширити стандартну модель Identity можна шляхом створення класу, успадкованого від IdentityUser:

public class ApplicationUser : IdentityUser

{

// Додаткові поля для шахового сервісу

public string DisplayName { get; set; }

public int Rating { get; set; } = 1200; // Початковий рейтинг

public int GamesPlayed { get; set; } = 0;

public int GamesWon { get; set; } = 0;

public int GamesLost { get; set; } = 0;

public int GamesDraw { get; set; } = 0;

public DateTime RegistrationDate { get; set; } = DateTime.UtcNow;

public DateTime LastActivityDate { get; set; } = DateTime.UtcNow;

public string ProfilePictureUrl { get; set; }

public string Country { get; set; }

// Навігаційні властивості

public virtual ICollection<GameSession> GameSessions { get; set; }

}

Також потрібно оновити ApplicationDbContext:

public class ApplicationDbContext : IdentityDbContext<ApplicationUser>

{

public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)

: base(options)

{

}

public DbSet<GameSession> GameSessions { get; set; }

protected override void OnModelCreating(ModelBuilder builder)

{

base.OnModelCreating(builder);

// Додаткові налаштування моделі

}

}

**4. Налаштування аутентифікації на основі кук**

**Реалізація**

В файлі Program.cs потрібно додати налаштування кук-аутентифікації:

builder.Services.ConfigureApplicationCookie(options =>

{

options.Cookie.HttpOnly = true;

options.ExpireTimeSpan = TimeSpan.FromDays(14);

options.LoginPath = "/Identity/Account/Login";

options.LogoutPath = "/Identity/Account/Logout";

options.AccessDeniedPath = "/Identity/Account/AccessDenied";

options.SlidingExpiration = true;

// Додаткові налаштування безпеки

options.Cookie.SecurePolicy = CookieSecurePolicy.Always; // Вимагає HTTPS

options.Cookie.SameSite = SameSiteMode.Strict;

});

**5. Налаштування авторизації на основі ролей**

**Реалізація**

Реалізація авторизації на основі ролей складається з кількох кроків:

1. **Додавання політик авторизації в Program.cs**:
2. builder.Services.AddAuthorization(options =>
3. {
4. options.AddPolicy("RequireAdminRole", policy => policy.RequireRole("Admin"));
5. options.AddPolicy("RequireModeratorRole", policy => policy.RequireRole("Admin", "Moderator"));
6. options.AddPolicy("RequirePlayerRole", policy => policy.RequireRole("Player", "Admin", "Moderator"));
7. });
8. **Використання атрибутів авторизації в контролерах**:
9. [Authorize(Policy = "RequireAdminRole")]
10. public class AdminController : BaseController
11. {
12. // Дії, доступні тільки адміністраторам
13. }
14. [Authorize(Policy = "RequireModeratorRole")]
15. public class ModerationController : BaseController
16. {
17. // Дії, доступні модераторам та адміністраторам
18. }
19. [Authorize(Policy = "RequirePlayerRole")]
20. public class GameController : BaseController
21. {
22. // Дії, доступні всім авторизованим користувачам
23. }
24. **Використання авторизації в окремих методах контролера**:
25. public class GameController : BaseController
26. {
27. [Authorize(Policy = "RequirePlayerRole")]
28. public IActionResult Play()
29. {
30. // Доступно всім авторизованим користувачам
31. }
33. [Authorize(Policy = "RequireModeratorRole")]
34. public IActionResult ManageGame(int id)
35. {
36. // Доступно тільки модераторам та адміністраторам
37. }
38. }
39. **Авторизація в представленнях**:
40. @if (User.IsInRole("Admin"))
41. {
42. <div class="admin-panel">
43. <!-- Елементи для адміністраторів -->
44. </div>
45. }
46. @if (User.IsInRole("Moderator") || User.IsInRole("Admin"))
47. {
48. <div class="moderation-tools">
49. <!-- Інструменти модерації -->
50. </div>
51. }

**6. Реалізація реєстрації користувача**

**6.1. Можливість введення необхідних даних та їх валідація**

Для реалізації форми реєстрації з розширеними полями потрібно створити або оновити сторінку реєстрації в Areas/Identity/Pages/Account/Register.cshtml:

**Модель (Register.cshtml.cs):**

public class InputModel

{

[Required]

[EmailAddress]

[Display(Name = "Email")]

public string Email { get; set; }

[Required]

[StringLength(100, ErrorMessage = "Довжина {0} повинна бути хоча б {2} і максимум {1} символів.", MinimumLength = 6)]

[DataType(DataType.Password)]

[Display(Name = "Пароль")]

public string Password { get; set; }

[DataType(DataType.Password)]

[Display(Name = "Підтвердження паролю")]

[Compare("Password", ErrorMessage = "Паролі не співпадають.")]

public string ConfirmPassword { get; set; }

[Required]

[StringLength(50, ErrorMessage = "Довжина {0} повинна бути хоча б {2} і максимум {1} символів.", MinimumLength = 3)]

[Display(Name = "Ім'я для відображення")]

public string DisplayName { get; set; }

[Display(Name = "Країна")]

public string Country { get; set; }

}

**Сторінка реєстрації (Register.cshtml):**

@page

@model RegisterModel

@{

ViewData["Title"] = "Реєстрація";

}

<h1>@ViewData["Title"]</h1>

<div class="row">

<div class="col-md-6">

<form asp-route-returnUrl="@Model.ReturnUrl" method="post">

<h4>Створення нового облікового запису.</h4>

<hr />

<div asp-validation-summary="All" class="text-danger"></div>

<div class="form-group">

<label asp-for="Input.Email"></label>

<input asp-for="Input.Email" class="form-control" />

<span asp-validation-for="Input.Email" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="Input.DisplayName"></label>

<input asp-for="Input.DisplayName" class="form-control" />

<span asp-validation-for="Input.DisplayName" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="Input.Country"></label>

<select asp-for="Input.Country" class="form-control">

<option value="">-- Виберіть країну --</option>

<option value="UA">Україна</option>

<option value="US">США</option>

<!-- Додаткові країни -->

</select>

<span asp-validation-for="Input.Country" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="Input.Password"></label>

<input asp-for="Input.Password" class="form-control" />

<span asp-validation-for="Input.Password" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="Input.ConfirmPassword"></label>

<input asp-for="Input.ConfirmPassword" class="form-control" />

<span asp-validation-for="Input.ConfirmPassword" class="text-danger"></span>

</div>

<button type="submit" class="btn btn-primary">Зареєструватись</button>

</form>

</div>

</div>

@section Scripts {

<partial name="\_ValidationScriptsPartial" />

}

**6.2. Коректне збереження даних**

В обробнику форми реєстрації потрібно доповнити метод OnPostAsync для збереження додаткових полів:

public async Task<IActionResult> OnPostAsync(string returnUrl = null)

{

returnUrl ??= Url.Content("~/");

ExternalLogins = (await \_signInManager.GetExternalAuthenticationSchemesAsync()).ToList();

if (ModelState.IsValid)

{

var user = new ApplicationUser

{

UserName = Input.Email,

Email = Input.Email,

DisplayName = Input.DisplayName,

Country = Input.Country,

RegistrationDate = DateTime.UtcNow,

LastActivityDate = DateTime.UtcNow,

Rating = 1200 // Початковий рейтинг

};

var result = await \_userManager.CreateAsync(user, Input.Password);

if (result.Succeeded)

{

\_logger.LogInformation("Користувач створив новий обліковий запис з паролем.");

// Присвоєння ролі "Player" по замовчуванню

await \_userManager.AddToRoleAsync(user, "Player");

// Генерація токену підтвердження email

var code = await \_userManager.GenerateEmailConfirmationTokenAsync(user);

code = WebEncoders.Base64UrlEncode(Encoding.UTF8.GetBytes(code));

var callbackUrl = Url.Page(

"/Account/ConfirmEmail",

pageHandler: null,

values: new { area = "Identity", userId = user.Id, code = code, returnUrl = returnUrl },

protocol: Request.Scheme);

await \_emailSender.SendEmailAsync(Input.Email, "Підтвердження email",

$"Підтвердіть ваш обліковий запис, <a href='{HtmlEncoder.Default.Encode(callbackUrl)}'>натиснувши тут</a>.");

if (\_userManager.Options.SignIn.RequireConfirmedAccount)

{

return RedirectToPage("RegisterConfirmation", new { email = Input.Email, returnUrl = returnUrl });

}

else

{

await \_signInManager.SignInAsync(user, isPersistent: false);

return LocalRedirect(returnUrl);

}

}

foreach (var error in result.Errors)

{

ModelState.AddModelError(string.Empty, error.Description);

}

}

// Якщо дійшли до цього місця, сталася помилка, поверніть форму

return Page();

}

**6.3. Підтвердження реєстрації (відправка email або підтвердження Адміністратором)**

**Реалізація відправки підтвердження email:**

1. **Налаштування сервісу відправки email в Program.cs**:

builder.Services.AddTransient<IEmailSender, EmailService>();

builder.Services.Configure<EmailSettings>(builder.Configuration.GetSection("EmailSettings"));

1. **Створення класу налаштувань email**:

public class EmailSettings

{

public string SmtpServer { get; set; }

public int SmtpPort { get; set; }

public string SenderEmail { get; set; }

public string SenderName { get; set; }

public string UserName { get; set; }

public string Password { get; set; }

}

1. **Реалізація сервісу відправки email**:

public class EmailService : IEmailSender

{

private readonly EmailSettings \_emailSettings;

private readonly ILogger<EmailService> \_logger;

public EmailService(IOptions<EmailSettings> emailSettings, ILogger<EmailService> logger)

{

\_emailSettings = emailSettings.Value;

\_logger = logger;

}

public async Task SendEmailAsync(string email, string subject, string htmlMessage)

{

try

{

var message = new MimeMessage();

message.From.Add(new MailboxAddress(\_emailSettings.SenderName, \_emailSettings.SenderEmail));

message.To.Add(new MailboxAddress("", email));

message.Subject = subject;

var bodyBuilder = new BodyBuilder

{

HtmlBody = htmlMessage

};

message.Body = bodyBuilder.ToMessageBody();

using (var client = new SmtpClient())

{

await client.ConnectAsync(\_emailSettings.SmtpServer, \_emailSettings.SmtpPort, SecureSocketOptions.StartTls);

await client.AuthenticateAsync(\_emailSettings.UserName, \_emailSettings.Password);

await client.SendAsync(message);

await client.DisconnectAsync(true);

}

}

catch (Exception ex)

{

\_logger.LogError(ex, "Помилка відправки email");

throw;

}

}

}

**Підтвердження реєстрації адміністратором:**

1. **Налаштування параметрів Identity**:

builder.Services.AddDefaultIdentity<ApplicationUser>(options =>

{

options.SignIn.RequireConfirmedAccount = true;

options.SignIn.RequireConfirmedEmail = false; // Вимикаємо цю опцію

})

.AddRoles<IdentityRole>()

.AddEntityFrameworkStores<ApplicationDbContext>();

1. **Сторінка адміністрування користувачів**:

@page

@model AdminUsersModel

@{

ViewData["Title"] = "Керування користувачами";

}

<h1>@ViewData["Title"]</h1>

<table class="table">

<thead>

<tr>

<th>Email</th>

<th>Ім'я</th>

<th>Статус</th>

<th>Дії</th>

</tr>

</thead>

<tbody>

@foreach (var user in Model.Users)

{

<tr>

<td>@user.Email</td>

<td>@user.DisplayName</td>

<td>@(user.EmailConfirmed ? "Підтверджено" : "Очікує підтвердження")</td>

<td>

@if (!user.EmailConfirmed)

{

<form method="post">

<input type="hidden" name="userId" value="@user.Id" />

<button type="submit" class="btn btn-sm btn-success">Підтвердити</button>

</form>

}

</td>

</tr>

}

</tbody>

</table>

1. **Код-поза для обробки підтвердження**:

public class AdminUsersModel : PageModel

{

private readonly UserManager<ApplicationUser> \_userManager;

public AdminUsersModel(UserManager<ApplicationUser> userManager)

{

\_userManager = userManager;

}

public List<ApplicationUser> Users { get; set; }

public async Task OnGetAsync()

{

Users = await \_userManager.Users.ToListAsync();

}

public async Task<IActionResult> OnPostAsync(string userId)

{

var user = await \_userManager.FindByIdAsync(userId);

if (user != null)

{

// Генерація токену підтвердження

var token = await \_userManager.GenerateEmailConfirmationTokenAsync(user);

// Підтвердження email

var result = await \_userManager.ConfirmEmailAsync(user, token);

if (result.Succeeded)

{

// Перенаправлення назад на сторінку

return RedirectToPage();

}

}

return BadRequest();

}

}

**7. Реалізація логіну**

**7.1. Введення та валідація імені користувача (email) та паролю**

Для реалізації логіну потрібно оновити сторінку входу в Areas/Identity/Pages/Account/Login.cshtml:

**Модель (Login.cshtml.cs):**

public class InputModel

{

[Required]

[EmailAddress]

public string Email { get; set; }

[Required]

[DataType(DataType.Password)]

public string Password { get; set; }

[Display(Name = "Запам'ятати мене")]

public bool RememberMe { get; set; }

}

**Сторінка логіну (Login.cshtml):**

@page

@model LoginModel

@{

ViewData["Title"] = "Вхід";

}

<h1>@ViewData["Title"]</h1>

<div class="row">

<div class="col-md-6">

<section>

<form id="account" method="post">

<h4>Увійдіть за допомогою облікового запису LambdaChess.</h4>

<hr />

<div asp-validation-summary="All" class="text-danger"></div>

<div class="form-group">

<label asp-for="Input.Email"></label>

<input asp-for="Input.Email" class="form-control" />

<span asp-validation-for="Input.Email" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="Input.Password"></label>

<input asp-for="Input.Password" class="form-control" />

<span asp-validation-for="Input.Password" class="text-danger"></span>

</div>

<div class="form-group">

<div class="custom-control custom-checkbox">

<input asp-for="Input.RememberMe" type="checkbox" class="custom-control-input" />

<label asp-for="Input.RememberMe" class="custom-control-label"></label>

</div>

</div>

<div class="form-group">

<button type="submit" class="btn btn-primary">Увійти</button>

</div>

<div class="form-group">

<p>

<a id="forgot-password" asp-page="./ForgotPassword">Забули пароль?</a>

</p>

<p>

<a asp-page="./Register" asp-route-returnUrl="@Model.ReturnUrl">Зареєструватися як новий користувач</a>

</p>

</div>

</form>

</section>

</div>

</div>

@section Scripts {

<partial name="\_ValidationScriptsPartial" />

}

**Обробник логіну (Login.cshtml.cs - метод OnPostAsync):**

public async Task<IActionResult> OnPostAsync(string returnUrl = null)

{

returnUrl ??= Url.Content("~/");

ExternalLogins = (await \_signInManager.GetExternalAuthenticationSchemesAsync()).ToList();

if (ModelState.IsValid)

{

// Оновлення часу останньої активності користувача

var user = await \_userManager.FindByEmailAsync(Input.Email);

if (user != null)

{

user.LastActivityDate = DateTime.UtcNow;

await \_userManager.UpdateAsync(user);

}

// Спроба входу користувача

var result = await \_signInManager.PasswordSignInAsync(Input.Email, Input.Password, Input.RememberMe, lockoutOnFailure: true);

if (result.Succeeded)

{

\_logger.LogInformation("Користувач увійшов у систему.");

return LocalRedirect(returnUrl);

}

if (result.RequiresTwoFactor)

{

return RedirectToPage("./LoginWith2fa", new { ReturnUrl = returnUrl, RememberMe = Input.RememberMe });

}

if (result.IsLockedOut)

{

\_logger.LogWarning("Обліковий запис користувача заблоковано.");

return RedirectToPage("./Lockout");

}

else

{

ModelState.AddModelError(string.Empty, "Невдала спроба входу.");

return Page();

}

}

// Якщо дійшли до цього місця, сталася помилка, поверніть форму

return Page();

}

**8. Реалізація відновлення паролю**

**8.1. Можливість отримати посилання для відновлення паролю**

**Сторінка запиту відновлення паролю (ForgotPassword.cshtml):**

@page

@model ForgotPasswordModel

@{

ViewData["Title"] = "Забули пароль?";

}

<h1>@ViewData["Title"]</h1>

<div class="row">

<div class="col-md-6">

<form method="post">

<h4>Введіть свій email.</h4>

<hr />

<div asp-validation-summary="All" class="text-danger"></div>

<div class="form-group">

<label asp-for="Input.Email"></label>

<input asp-for="Input.Email" class="form-control" />

<span asp-validation-for="Input.Email" class="text-danger"></span>

</div>

<button type="submit" class="btn btn-primary">Надіслати</button>

</form>

</div>

</div>

@section Scripts {

<partial name="\_ValidationScriptsPartial" />

}

**Обробник запиту відновлення паролю (ForgotPassword.cshtml.cs):**

public class ForgotPasswordModel : PageModel

{

private readonly UserManager<ApplicationUser> \_userManager;

private readonly IEmailSender \_emailSender;

public ForgotPasswordModel(UserManager<ApplicationUser> userManager, IEmailSender emailSender)

{

\_userManager = userManager;

\_emailSender = emailSender;

}

[BindProperty]

public InputModel Input { get; set; }

public class InputModel

{

[Required]

[EmailAddress]

public string Email { get; set; }

}

public async Task<IActionResult> OnPostAsync()

{

if (ModelState.IsValid)

{

var user = await \_userManager.FindByEmailAsync(Input.Email);

if (user == null || !(await \_userManager.IsEmailConfirmedAsync(user)))

{

// Не розкривайте, що користувач не існує або не підтверджений

return RedirectToPage("./ForgotPasswordConfirmation");

}

// Генерація токену для скидання паролю

var code = await \_userManager.GeneratePasswordResetTokenAsync(user);

code = WebEncoders.Base64UrlEncode(Encoding.UTF8.GetBytes(code));

var callbackUrl = Url.Page(

"/Account/ResetPassword",

pageHandler: null,

values: new { area = "Identity", code },

protocol: Request.Scheme);

await \_emailSender.SendEmailAsync(

Input.Email,

"Скидання паролю",

$"Для скидання паролю <a href='{HtmlEncoder.Default.Encode(callbackUrl)}'>натисніть тут</a>.");

return RedirectToPage("./ForgotPasswordConfirmation");

}

return Page();

}

}

**8.2. Можливість введення нового паролю та оновлення його в базі**

**Сторінка введення нового паролю (ResetPassword.cshtml):**

@page

@model ResetPasswordModel

@{

ViewData["Title"] = "Скидання паролю";

}

<h1>@ViewData["Title"]</h1>

<div class="row">

<div class="col-md-6">

<form method="post">

<h4>Введіть новий пароль.</h4>

<hr />

<div asp-validation-summary="ModelOnly" class="text-danger"></div>

<input asp-for="Input.Code" type="hidden" />

<div class="form-group">

<label asp-for="Input.Email"></label>

<input asp-for="Input.Email" class="form-control" />

<span asp-validation-for="Input.Email" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="Input.Password"></label>

<input asp-for="Input.Password" class="form-control" />

<span asp-validation-for="Input.Password" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="Input.ConfirmPassword"></label>

<input asp-for="Input.ConfirmPassword" class="form-control" />

<span asp-validation-for="Input.ConfirmPassword" class="text-danger"></span>

</div>

<button type="submit" class="btn btn-primary">Скинути</button>

</form>

</div>

</div>

@section Scripts {

<partial name="\_ValidationScriptsPartial" />

}

**Обробник скидання паролю (ResetPassword.cshtml.cs):**

public class ResetPasswordModel : PageModel

{

private readonly UserManager<ApplicationUser> \_userManager;

public ResetPasswordModel(UserManager<ApplicationUser> userManager)

{

\_userManager = userManager;

}

[BindProperty]

public InputModel Input { get; set; }

public class InputModel

{

[Required]

[EmailAddress]

public string Email { get; set; }

[Required]

[StringLength(100, ErrorMessage = "Довжина {0} повинна бути хоча б {2} і максимум {1} символів.", MinimumLength = 6)]

[DataType(DataType.Password)]

public string Password { get; set; }

[DataType(DataType.Password)]

[Display(Name = "Підтвердження паролю")]

[Compare("Password", ErrorMessage = "Паролі не співпадають.")]

public string ConfirmPassword { get; set; }

[Required]

public string Code { get; set; }

}

public IActionResult OnGet(string code = null)

{

if (code == null)

{

return BadRequest("Код для скидання паролю не надано.");

}

else

{

Input = new InputModel

{

Code = Encoding.UTF8.GetString(WebEncoders.Base64UrlDecode(code))

};

return Page();

}

}

public async Task<IActionResult> OnPostAsync()

{

if (!ModelState.IsValid)

{

return Page();

}

var user = await \_userManager.FindByEmailAsync(Input.Email);

if (user == null)

{

// Не розкриваємо інформацію про те, що користувач не існує

return RedirectToPage("./ResetPasswordConfirmation");

}

var result = await \_userManager.ResetPasswordAsync(user, Input.Code, Input.Password);

if (result.Succeeded)

{

return RedirectToPage("./ResetPasswordConfirmation");

}

foreach (var error in result.Errors)

{

ModelState.AddModelError(string.Empty, error.Description);

}

return Page();

}

}

public ResetPasswordModel(UserManager