3. Розробка мобільного android-додатку з застосуванням принципів clean architecture / Г. Козуб, Ю. Козуб, Г. Могильний, А. Жуков. *Вісник Східноукраїнського нац. ун-ту ім. Володимира Даля.* 2021. № 5(269). С. 5–10. URL: https://journals.snu.edu.ua/index.php/VisnikSNU/article/view/38

4. Козуб Ю., Козуб Г. Особливості розробки мультиплатформних застосунків на kotlin. *Вісник Хмельн. нац. ун-ту.* 2023. № 1. С. 224–229. URL: https://journals.khnu.km.ua/vestnik/?p=16772

УДК 004.056.523:004.651.54:004.439:681.5

Мисько Б. В., *здобувач вищої освіти 4 курсу спеціальності 122 Комп'ютерні науки,*
Фриз І. В., *канд. фіз.-мат. наук, старший викладач кафедри інформаційних технологій*

ІНТЕГРАЦІЯ JWT ТА АЛГОРИТМІВ ГЕШУВАННЯ ДЛЯ АВТЕНТИФІКАЦІЇ ТА АВТОРИЗАЦІЇ В CRM-СИСТЕМАХ

Донецький національний університет імені Василя Стуса, м. Вінниця

У сучасних інформаційних системах, зокрема у CRM-системах (Customer Relationship Management), безпека користувацьких даних та контроль доступу до ресурсів є критично важливими аспектами. З огляду на активне впровадження вебтехнологій та хмарних обчислень традиційні методи автентифікації, які базуються на збереженні паролів у відкритому вигляді або в сесійних змінних, втратили актуальність. Їм на зміну прийшли криптографічно стійкі алгоритми гешування та маркерні механізми авторизації, серед яких особливої уваги заслуговує JSON Web Token (JWT) [1].

Метою дослідження є розробка надійного механізму автентифікації та авторизації користувачів для CRM-системи. У межах роботи передбачено впровадження безпечного способу зберігання облікових даних, реалізацію цифрового токена для підтвердження ідентичності користувача, а також побудову гнучкої рольової моделі контролю доступу.

Гешування паролів – це незворотний процес, у якому зі вхідного значення (пароля) створюється фіксованої довжини геш-код. Одним із найбільш надійних алгоритмів є BCrypt, який має вбудований механізм сольової генерації та адаптивної складності [2]. Перевагами BCrypt є стійкість до атак методом перебору та можливість збільшення складності з часом.

JWT – відкритий стандарт (RFC 7519), який описує компактний, автономний спосіб безпечної передачі інформації між сторонами як JSON-об'єкт [1]. Токен складається з трьох частин: заголовка (header), корисного навантаження (payload) та підпису (signature). Його переваги включають можливість безсерверної авторизації, розширюваність і незалежність від конкретного протоколу передачі [3]. Архітектурна модель реалізації механізму автентифікації та авторизації базується на поєднанні трьох основних компонентів:

1. Гешування пароля для збереження паролів у захищеному вигляді:

```
namespace CRMSystem.WebAPI.Auth
{
    public class PasswordHasher : IPasswordHasher
    {
        public string Generate(string password) =>
            BCrypt.Net.BCrypt.EnhancedHashPassword(password);

        public bool Verify(string password, string hashedPassword) =>
            BCrypt.Net.BCrypt.EnhancedVerify(password, hashedPassword);
    }
}
```

2. Генерація JWT для створення цифрового токена на основі користувацьких даних:

```
public class JwtProvider(IOptions<JwtOptions> options)
    : IJwtProvider
{
    private readonly JwtOptions _options = options.Value;

    public string GenerateToken(User user)
    {
        Claim[] claims =
        [
            new("userId", user.Id.ToString()),
            new("username", user.Username),
            new("role", user.RoleId.ToString())
        ];

        var signingCredentials = new SigningCredentials(
            new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(_options.SecretKey)),
            SecurityAlgorithms.HmacSha256);

        var token = new JwtSecurityToken(
            claims: claims,
            signingCredentials: signingCredentials,
            issuer: _options.Issuer,
            audience: _options.Audience,
            expires: DateTime.UtcNow.AddHours(_options.ExpireHours));

        return new JwtSecurityTokenHandler().WriteToken(token);
    }
}
```

3. Авторизація за ролями для контролю доступу до ресурсів залежно від ролі користувача:

```
public static void AddApiAuthentication(this IServiceCollection services,
IConfiguration configuration)
{
    var jwtOptions =
configuration.GetSection(nameof(JwtOptions)).Get<JwtOptions>();
```

```
    services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)
        .AddJwtBearer(options =>
        {
            options.TokenValidationParameters = new
TokenValidationParameters
            {
                ValidateIssuer = false,
                ValidateAudience = false,
                ValidateLifetime = true,
                ValidateIssuerSigningKey = true,
                IssuerSigningKey = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(jwtOptions!.SecretKey))
            };

            options.Events = new JwtBearerEvents
            {
                OnMessageReceived = context =>
                {
                    context.Token = context.Request.Cookies["jwt"];
                    return Task.CompletedTask;
                }
            };
        });

    services.AddAuthorization();
}

public static void AddAuthorizationPolicy(this IServiceCollection services)
{
    services.AddAuthorizationBuilder()
        .AddPolicy(AuthorizationPolicies.AdminOnly, policy =>
            policy.RequireRole(((int)Roles.Admin).ToString()))
        .AddPolicy(AuthorizationPolicies.UserOnly, policy =>
            policy.RequireRole(((int)Roles.User).ToString()))
        .AddPolicy(AuthorizationPolicies.UserOrAdmin, policy =>
            policy.RequireRole(((int)Roles.User).ToString(),
((int)Roles.Admin).ToString()));
}

public static void AddCookiePolicy(this IApplicationBuilder app)
{
    app.UseCookiePolicy(new CookiePolicyOptions
    {
        MinimumSameSitePolicy = SameSiteMode.Strict,
        HttpOnly = HttpOnlyPolicy.Always,
        Secure = CookieSecurePolicy.Always
    });
}
```

Представлені фрагменти коду демонструють реалізацію безпечного механізму ідентифікації та розмежування доступу користувачів у CRM-системі. Для зберігання паролів застосовується стійкий криптографічний алгоритм, який унеможливлює їх пряме відновлення. Цифрові токени, сформовані на основі корис-

тувацьких атрибутів, дають змогу здійснювати перевірку прав доступу без необхідності зберігати сесію на сервері. Застосування ролей та політик доступу забезпечує гнучке управління дозволами в межах системи, а інтеграція токена в Cookies з налаштуваннями безпеки мінімізує ризик атак типу XSS та CSRF [3].

Отже, проведене проєктування та впровадження дали змогу створити модель, що забезпечує як безпечне зберігання облікових даних, так і контроль доступу на основі ролей без постійного зберігання сесійної інформації. Така архітектура відповідає актуальним вимогам до захисту персональних даних, є масштабованою, ефективною для розподілених систем і може бути адаптована для різних типів користувацьких інтерфейсів. Запропонований підхід доводить доцільність поєднання гешування та токен-орієнтованої авторизації як основи для побудови надійної та гнучкої системи безпеки у корпоративному програмному забезпеченні.

Список використаних джерел

1. A TOTP-based secure data storage system in the cloud environment using the JWT token approach / A. Shahnawaz, A. Mohd, A. Javed, M. Shabana. *International Journal of Systems Assurance Engineering and Management.* 2025. Vol. 16. P. 1565–1578. DOI: 10.1007/s13198-025-02775-8.

2. Enhancing the Security of JSON Web Token Using Signal Protocol and Ratchet System / P. Singh, G. Choudhary, S. K. Shandilya, V. Sihag. *Proceedings of Emerging Trends and Technologies on Intelligent Systems. Advances in Intelligent Systems and Computing.* Vol. 1414. Springer, Singapore. DOI: 10.1007/978-981-19-4182-5_31.

3. Pyroh M., Tereshchuk G., Toroshanko O. Authentication Principles as Security aspects of Web Development. *Measuring And Computing Devices In Technological Processes*. 2025. №. 1. P. 294–301. DOI: 10.31891/2219-9365-2025-81-36.

УДК 004.056.5

Первачук Р. Ю., *здобувач вищої освіти 4 курсу спеціальності 122 Комп'ютерні науки,*
Антонов Ю. С., *канд. фіз.-мат. наук, доцент, доцент кафедри інформаційних технологій*

РОЗРОБКА УТИЛІТ ДЛЯ АГРЕГАЦІЇ ІНТЕРНЕТ-РЕСУРСІВ, ЩО ПІДЛЯГАЮТЬ БЛОКУВАННЮ

Донецький національний університет імені Василя Стуса, м. Вінниця

У сучасному інформаційному просторі за допомогою інтернет-ресурсів здійснюються різноманітні кібератаки, від фішингу до ransomware або DDoS-атак [1–3]. У зв'язку з цим підтримка актуального переліку інтернет-ресурсів, що підлягають блокуванню, має здійснюватись як на основі локальних списків, так і на основі списків, доступних у мережі Інтернет. Блокування таких джерел є одним із базових засобів протидії кіберзагрозам як у державних, так і в приватних мережах. Проте наявні засоби фільтрації часто є вузькоспеціалізованими, обмеженими у форматах або залежними від хмарних сервісів, що знижує рівень автономності та адаптивності.