Приватний заклад вищої освіти
Одеський технологічний університет «ШАГ»
Кафедра інформаційних технологій та фундаментальної підготовки

Випускна кваліфікаційна робота бакалавра

# «СИСТЕМА УЗГОДЖЕННЯ ВЗАЄМОДІЇ НАДАВАЧІВ ТА СПОЖИВАЧІВ ПОСЛУГ»

на здобуття бакалаврського рівня вищої освіти
зі спеціальності «122 Комп'ютерні науки»

Виконавці проєкту

| № | П.І.Б. | Група |
|---|---|---|
| 1 | Клементьєв Д. С. | КН-П-202 |
| 2 | Луцевич Д. А. | КН-П-202 |
| 3 | Гринченко Н. С. | КН-П-202 |
| 4 | Радовська Н. Г. | КН-Д-201 |

Автор звіту

_____ Клементьєв Даніїл Сергійович

,

Одеса – 2024

# АНОТАЦІЯ

УДК 004.9
К-48

Клементьєв Д. С. Система узгодження взаємодії надавачів та споживачів послуг: Автореферат випускної кваліфікаційної роботи на здобуття бакалаврського рівня вищої освіти зі спеціальності «122 Комп’ютерні науки». - Одеса: ОТУШ, 2024 - 20 с.

Робота присвячена дослідженням, розвиненню та впровадженню інформаційної системи, головним призначенням якої є узгодження взаємодії надавачів та споживачів послуг. Додаткові функції системи включають реєстрацію, авторизацію та автентифікацію, засоби створення оголошень, персональний кабінет, пошуковий механізм.

Актуальність роботи визначається тим, що в сучасній ситуації цифрового світу необхідно оптимізувати процеси взаємодії між постачальниками послуг і споживачами. Важливість цього завдання зростає в зв'язку зі збільшенням кількості онлайн-сервісів, що пропонують різні послуги, і вимог до швидкого і зручного доступу до цих сервісів. Крім того, актуальність даного дослідження посилюється необхідністю інтеграції безпечних методів обробки і зберігання персональних даних користувачів. Це особливо важливо в контексті нових законодавчих вимог до захисту персональних даних.

Метою цього дослідження є інформаційна система, що забезпечує узгодження взаємодії надавачів та споживачів послуг. Предметом дослідження цієї системи виступає оптимізація процесів взаємодії користувачів системи, зокрема, пошукові алгоритми.

Призначення даного дослідження є розробка та впровадження інформаційних систем, що забезпечують ефективну взаємодію між постачальниками послуг та споживачами з урахуванням аспектів безпеки та зручності використання.

Задачі дослідження включають:

- Провести аналіз існуючих інформаційних систем для узгодження взаємодії надавачів та споживачів послуг, визначити їхні переваги та недоліки.
- Розробити програмний комплекс, що включає засоби реєстрації, авторизації, автентифікації, створення оголошень, персональний кабінет та пошуковий механізм.
- Провести тестування розробленої системи у різних умовах для оцінки її ефективності та надійності.
- Впровадити механізми захисту даних для забезпечення безпеки персональних даних користувачів.

Проведені випробування серверної частини були спрямовані на перевірку коректності роботи всіх ключових функцій системи, а саме: реєстрації, авторизації, редагування користувачів, створення та управління оголошеннями, а також завантаження зображень. Випробування здійснювалися за допомогою таких інструментів:

- Swagger UI: Використовувався для інтерактивного тестування кінцевих точок API, дозволяючи виконувати запити та переглядати відповіді сервера у реальному часі.
- Postman: Використовувався для детального тестування HTTP-запитів, включаючи налаштування параметрів запитів та аналіз отриманих відповідей.
- Azure Web App: Платформа, на якій було розгорнуто та протестовано серверну частину API.

Для вирішення поставленого завдання були використані методи когнітивного аналізу, композиції та програмної інженерії. Були проведені огляди та аналіз нормативних та технічних документів.

В результаті дослідження була розроблена інформаційна система, що дозволяє ефективно координувати взаємодію між постачальником послуги і споживачем. Система забезпечує надійну реєстрацію та автентифікацію користувачів, а також безпечне зберігання персональних даних. Проведені випробування показали високу ефективність і надійність розробленої системи.

Робота містить 8 посилань на друковані та електронні джерела інформації. Розроблена інформаційна система має значний потенціал для застосування у різних галузях цифрової економіки, сприяючи підвищенню ефективності та зручності надання та отримання послуг.

ЗМІСТ

# ВСТУП

За результатами всеукраїнського опитування, проведеного Київським міжнародним інститутом соціології у 2023 році на замовлення Програми розвитку ООН (UNDP) в Україні за підтримки Швеції та у партнерстві з Міністерством цифрової трансформації України, кількість українців, які щоденно користуються інтернетом, протягом року зросла з 72% до 80%. З 2021 року цей показник збільшився на понад 10%. Державними електронними послугами користувалися 64% респондентів, а серед найпопулярніших ресурсів — портал та застосунок "Дія", якими користувалися 51% українців станом на початок жовтня 2023 року. Майже 80% користувачів оцінюють свій досвід користування державними електронними послугами як позитивний. Це свідчить про зростання рівня цифрової грамотності та задоволеність громадян електронними послугами [1]. Тому важливо, особливо у воєнний час, охопити всі сфери діяльності людей у цифрову екосистему, де люди зможуть віддалено отримувати та ділитися своїми послугами.

Питома вага кіберзагроз зростає і ця тенденція в міру розвитку інформаційних технологій та їх конвергенції з технологіями штучного інтелекту в найближче десятиліття посилюватиметься. Зростання такого впливу на функціонування структур управління як національних, так і транснаціональних формує нову безпекову ситуацію. Між світовими центрами сили відбувається поділ сфер впливу у кіберпросторі, посилюється їх прагнення за рахунок такого поділу забезпечити реалізацію власних геополітичних інтересів. [3]. Це вимагає подальших досліджень і розробок у цій сфері та підкреслює потребу впровадження сучасних засобів захисту інформації. Для забезпечення належного рівня кібербезпеки необхідно розробляти комплексні стратегії, що включають технічні, організаційні, правові та освітні заходи.

У цій роботі розглянуто популярну систему взаємодії користувачів, яка включає розміщення пропозицій від одних користувачів та пошук відповідних пропозицій іншими. При цьому враховуються різні фільтри, такі як геолокація обох користувачів.

Відповідно до функціональних вимог більшості цих інформаційних систем вони повинні забезпечувати можливість зберігання історії транзакцій, пошуку пропозицій, фіксації факту прийняття або відхилення. Перспективним є впровадження методів оплати. Це вимагає наявності заходів авторизації та автентифікації та захисту персональних даних від несанкціонованого доступу інших користувачів.

Детальне технічне завдання проєкту представлено далі. У рамках цього звіту розглядається клієнтська частина, що являє собою веб-додаток або мобільний додаток. Звіт складається з трьох розділів, у яких описується загальна характеристика системи, деталі створення клієнтської частини та результати її розгортання і тестування. Висновки відображають досягнуті результати в рамках цієї частини проєкту, а загальний висновок є сукупністю звітів усіх учасників проєкту.

ТЕХНІЧНЕ ЗАВДАННЯ ДО ПРОЄКТУ

**Загальний опис.**
Програмне забезпечення (ПЗ) розроблено з метою створення системи, яка налагоджує та спрощує взаємодію користувачів з різними ролями (клієнт і виконавець). Програмне забезпечення повинно працювати на різних пристроях (смартфони, планшети, комп'ютери), щоб користувачі мали можливість зручно взаємодіяти з системою у будь-який час та з будь-якого місця. Це забезпечує зручність та гнучкість у використанні послуг,

**Призначення:**
Система орієнтована на забезпечення ефективної взаємодії між користувачами різних ролей (клієнт і виконавець), створення зручного середовища для обміну інформацією між користувачами та надання можливості вибору (клієнти вибирають виконавця, а виконавці - клієнта).

**Вимоги до функціоналу:**
*Аутентифікація користувачів.*
Для використання всієї функціональності ПЗ вимагається аутентифікація користувача. Для реєстрації користувач повинен вказати свою електронну пошту, ім'я (для відображення його даних профілю у інших користувачів / звернення до нього) та придумати пароль (є можливість подальшого додавання реєстрації за допомогою сторонніх сервісів - Google, Apple тощо). Після реєстрації користувач повинен підтвердити свою електронну адресу шляхом введення коду / переходу за посиланням з листа, надісланого йому на пошту після реєстрації. Також під час реєстрації вказується роль користувача (клієнт / виконавець).
Для входу в обліковий запис використовується електронна пошта та пароль.

*Особистий кабінет*
Особистий кабінет користувача містить інформацію про користувача, його історію замовлень. Користувач може редагувати свої дані та змінювати аватар. Також доступна можливість вийти з облікового запису. Залежно від ролі, надаються наступні можливості:
*Для клієнта:*
Збереження улюбленців: клієнт може зберегти кілька своїх домашніх тварин.
Додавання оголошення - користувач розміщує пропозицію щодо послуги для пошуку виконавця.
*Для виконавця:*
Додавання резюме: виконавець може додати кілька резюме з різними категоріями послуг та категоріями тварин.
Додавання пропозиції в “збережені”: виконавець може додати декілька пропозицій та переглянути їх пізніше.

Ці функціональності дозволяють користувачам управляти своїми профілями відповідно до їхніх ролей.

*Пошук пропозиції*
Ця функціональність забезпечує пошук актуальних пропозицій для виконавця і включає наступні можливості:
Перегляд стрічки пропозицій. Виконавець може переглядати список актуальних пропозицій у вигляді карток з основною інформацією: назва, короткий опис, тип та інше.
Можливість фільтрації і сортування. Виконавець може фільтрувати пропозиції за категорією послуги, категорією тварини та іншими критеріями. Також є можливість сортування за датою, популярністю тощо.
Пошук пропозицій. Виконавець може знайти конкретну пропозицію за ключовими словами, назвою тощо.
Детальний перегляд пропозиції. При натисканні на картку пропозиції з ленти відкривається її детальний огляд з усією інформацією, наданою замовником.
Взаємодія з пропозицією. Виконавець може відгукнутися на пропозицію, зв'язатися з клієнтом для уточнення деталей, а також додати її до збережених.

*Пошук виконавця за їхніми резюме*
Ця функціональність забезпечує зручний і ефективний процес пошуку виконавців відповідно до вимог клієнта і включає наступні можливості:
Перегляд ленти резюме виконавців: Клієнт може переглянути список резюме, що були опубліковані виконавцями, у вигляді карток.
Можливість фільтрації і сортування: У клієнта є можливість фільтрувати резюме за категорією послуги, категорією тварини та іншими критеріями. Також доступна можливість сортування за рейтингом, популярністю тощо.
Детальний перегляд резюме: При натисканні на картку резюме відкривається їхнє детальне відображення з усією інформацією, яку вказав виконавець.
Взаємодія з виконавцем: Клієнт може відгукнутися на резюме, зв'язатися з користувачем для обговорення деталей і умов співпраці.

**Архітектура та технології:**
Програмне забезпечення розроблено за клієнт-серверною архітектурою. Завдяки цьому забезпечується горизонтальне і вертикальне масштабування, розподіл обчислювальних ресурсів (зниження навантаження на клієнтські пристрої, ефективне використання ресурсів сервера), централізоване управління даними (всі дані зберігаються на сервері), гнучкість при розробці і підтримці ПЗ (клієнт і сервер можуть бути розроблені і оновлені незалежно один від одного). Основні компоненти системи включають в себе backend, frontend (веб-додаток, мобільний додаток).

*Backend*

Серверна частина проекту побудована на основі Azure Web App та ASP.NET Core. Дані зберігаються в CosmosDB, що забезпечує високу продуктивність та масштабованість бази даних. Для зручного тестування та документування API використовується Swagger UI.

Основні технічні деталі:

- Azure Web App: Використовується для хостингу серверної частини додатку, забезпечуючи безперебійну роботу і масштабованість.
- ASP.NET Core: Основний фреймворк для розробки бекенду, що забезпечує високу продуктивність та підтримку сучасних стандартів.
- CosmosDB: Документо-орієнтована база даних, яка забезпечує масштабованість та високу доступність даних.
- Swagger UI: Інструмент для документування API, що дозволяє легко тестувати та взаємодіяти з різними кінцевими точками API.
- API: Розроблено набір RESTful API для взаємодії з фронтендом та мобільним додатком. Всі дані передаються у форматі JSON.

*Frontend*

Включає в себе веб сайт і мобільний додаток, які призначені для взаємодії користувачів із системою.

*Веб Сайт*

Веб інтерфейс замовника та виконавця який доступен з будь якого браузеру. Надає користувачам зручний інтерфейс для створення резюме та пропозицій, редагування особистих даних та відгуків на пропозиції.
Веб сайт розроблений на основі NextJs та компонентів shadcn/ui, tailwind використаний для стилізації.

*Мобільний додаток*

Це інтерфейс користувача, завдяки якому можна взаємодіяти із системою (переглядати і управляти пропозиціями, шукати виконавців та працювати з профілем користувача). Мобільний додаток взаємодіє з бекендом через API. Для розробки використовується кросплатформне рішення (фреймворк React Native з використанням Expo).

# РОЗДІЛ 1
# ЗАГАЛЬНА ХАРАКТЕРИСТИКА СИСТЕМИ

Відповідно до технічного завдання (ТЗ), система була спроектована з використанням клієнт-серверної архітектури. Система включає наступні архітектурні компоненти:

- «Mockup»
- «Backend»
- «Frontend»

Частина «Mockup» являє собою набір моделей та описів поведінки користувачів. Модель відображає інтерфейси взаємодії користувачів різного типу з інформаційною системою, траєкторії їхніх рухів та переходів між різними інтерфейсами. При створенні інтерфейсів застосовано принципи розумної навігації, включно з підходами UI/UX, висновками колористики та ергономіки. Частина розміщена на платформі Figma:
https://www.figma.com/design/KRhpRsAPICoDGu8lJHkUjy/%D0%94%D0%B8%D0%BF%D0%BB%D0%BE%D0%BC?node-id=0-1&t=iQUc0cKH063YVKH3-1

Частина «Backend» реалізована як централізований вузол збереження та обробки даних, які створюються та модифікуються протягом життєвого циклу інформаційної системи. Ця частина включає базу даних проєкту та програмний інтерфейс (API) доступу до основних її функцій, таких як керування контентом, авторизація, узгодження профілів користувачів та керування оголошеннями. Сервер API доступний за посиланням: https://shelterbackapi.azurewebsites.net/swagger/.

Частина «Backend» виконана на основі програмного забезпечення .NET C#, з використанням баз даних CosmosDB та AzureDB. Додатково використано бібліотеку Newtonsoft JSON для роботи з JSON-даними.

Клієнтська частина є інтерфейсом взаємодії користувача з системою. Включає в себе веб сайт і мобільний додаток, які мають зручний і зрозумілий користувацький інтерфейс. Таке рішення забезпечує високу ефективність і доступність обміну інформацією між клієнтами та виконавцями незалежно від пристрою та платформи, які вони використовують. Усіма функціональностями системи можна скористатися як на веб сайті, так і в мобільному додатку (IOS і Android).

Для покращення безпеки, з урахуванням частого підключення мобільних пристроїв до незахищених мереж передачі даних, інформаційний обмін між частинами «Backend» та «Frontend» захищений шифруванням за допомогою сучасного криптоалгоритму SHA-256. Це забезпечує безпеку системи в відкритих мережах та спрощує задачі автентифікації та доступу до контенту з обмеженим доступом. Важливо зазначити, що алгоритм SHA-256 широко використовується в галузі криптовалют, де безпека даних є критично важливою. Наприклад, біткоїн та інші криптовалюти використовують SHA-256 для забезпечення цілісності та безпеки транзакцій [4]. Слід зазначити, що алгоритм SHA-256 є найпоширенішим у різних сферах цифрових технологій [5].

# РОЗДІЛ 2
# РЕАЛІЗАЦІЯ СЕРВЕРНОЇ ЧАСТИНИ

Серверна частина інформаційної системи реалізована з використанням наступних технологій:

- Хмарний сервіс застосунків.
- Засоби розроблення: мова програмування C# (з використанням .NET Core) та середовище розробки Visual Studio.
- Додаткове ПЗ та засоби: Azure для хостингу бази даних та сервісів (Azure Web App), Cosmos DB для управління базами даних, Newtonsoft JSON для роботи з JSON.

У складі серверної частини передбачено та реалізовано наступні функції:

- Реєстрація користувача та авторизація
- Створення, редагування та видалення оголошень
- Завантаження зображень для профілю користувача та оголошень
- Отримання інформації про користувачів та оголошення

Основні компоненти серверної частини:

1. *Конфігураційний файл appsettings.json*
   Містить налаштування для підключення до Cosmos DB.

```
{
    "Azure": {
        "CosmosDB": { "EndpointUri":
"https://shelter-back-api-db.documents.azure.com:443/",
        "PrimaryKey": "************************************",
        "DatabaseName": "ToDoList",
        "CollectionName": "Items"
        }
    }
}
```

2. *Program.cs*

Налаштування додатка, додавання служб, підключення до бази даних Cosmos DB:

```
builder.Services.AddSingleton<CosmosClient>(serviceProvider =>
{
    var configuration =
serviceProvider.GetRequiredService<IConfiguration>();
    var cosmosDbSection =
configuration.GetSection("Azure").GetSection("CosmosDB");
    return new CosmosClient(cosmosDbSection["EndpointUri"],
cosmosDbSection["PrimaryKey"]);
});

builder.Services.AddSingleton<CosmosDbService>();
builder.Services.AddSingleton<IKdfService, KdfService>();
builder.Services.AddSingleton<IHashService, HashService>();
```

3. *Services*

Містить сервіси, які забезпечують виконання ключових функцій бізнес-логіки додатка. Ці сервіси відповідають за обробку даних, взаємодію з базою даних, шифрування та хешування паролів, а також інші допоміжні операції.

a. *HashService:* хешування паролів.

```
public string Hash(string text)
{
    using var md5 = System.Security.Cryptography.MD5.Create();
    return
Convert.ToHexString(md5.ComputeHash(System.Text.Encoding.UTF8.GetB
ytes(text)));
}
```

b. *KdfService:* генерація солі та отримання похідного ключа.

```
public String GenerateSalt() {
    byte[] salt = new byte[16];
    using (var rng = RandomNumberGenerator.Create()) {
        rng.GetBytes(salt);
    }
    return Convert.ToBase64String(salt);
```

```
}
public String GetDerivedKey(String password, String salt) {
    return _hashService.Hash(password + salt);
}
```

4. *Models*
   Містить моделі даних, які використовуються для представлення та передачі інформації між різними компонентами системи. Ці моделі визначають структуру даних, що зберігаються у базі даних та обробляються в додатку.

   a. *User.cs:* Модель користувача.

```
public class User
{
    public String id { get; set; }
    public string partitionKey { get; set; } = "user";
    public string Name { get; set; } = string.Empty;
    public string Email { get; set; } = string.Empty;
    public string PasswordHash { get; set; } = string.Empty;
    public string PasswordSalt { get; set; } = string.Empty;
    public string Role { get; set; } = "Client";
    public string Token { get; set; } = string.Empty;
}
```

   b. *Advt.cs:* Модель оголошення.

```
public class Advt
{
    public string id { get; set; }
    public string partitionKey { get; set; } = "advt";

    public string AuthorId { get; set; }
    public string Title { get; set; } = string.Empty;
    public string Description { get; set; } = string.Empty;
    public string Price { get; set; } = string.Empty;
    public string Category { get; set; } = string.Empty;
    public string City { get; set; } = string.Empty;
    public String Date { get; set; } =
DateTime.Now.ToString("yyyy-MM-dd HH:mm");
    public string? Image { get; set; }
}
```

Крім цих двох моделей, було створено ще дві моделі для запитів з фронтенду. Моделі UserRequestModel та AdvtRequestModel були створені для того, щоб не запитувати на клієнтській частині зайві дані, як: id, partitionKey, Image, Date, PasswordHash, PasswordSalt, Token.

5. *Controllers*
   Містить контролери, які відповідають за обробку HTTP-запитів і керування потоком даних між клієнтською та серверною частинами додатка. Контролери визначають кінцеві точки API та реалізують логіку обробки запитів від користувачів, взаємодіючи з відповідними сервісами для виконання бізнес-логіки.

   a. *AdvtController*: Управління оголошеннями

```
[HttpPost]
[Route("AddAdvt")]
public async Task<IActionResult> AddAdvt([FromBody]
AdvtRequestModel model)
{
    var newAdvt = new Advt
    {
        id = Guid.NewGuid().ToString(),
        partitionKey = "advt",
        AuthorId = model.AuthorId,
        Title = model.Title,
        Description = model.Description,
        Price = model.Price,
        Category = model.Category,
        City = model.City
    };

    await _cosmosDbService.AddItemAsync(newAdvt);
    return CreatedAtAction(nameof(AddAdvt), newAdvt);
}

[HttpPost]
[Route("UploadImage/{id}")]
public async Task<IActionResult> UploadAvatar(string id,
[FromForm] IFormFile file) {
    var fileName =
$"{Guid.NewGuid()}{Path.GetExtension(file.FileName)}";
    var filePath = Path.Combine("D:\\home", fileName);
```

```
    using (var stream = new FileStream(filePath, FileMode.Create))
    {
        await file.CopyToAsync(stream);
    }

    var advt = await _cosmosDbService.GetItemAsync<Advt>(id);
    advt.Image = fileName;
    await _cosmosDbService.UpdateItemAsync(id, advt);

    return Ok(new { message = "Image uploaded successfully" });
}
```

b. *AccountController*: Управління користувачами.

```
[HttpPost]
[Route("AuthenticateUser")]
public IActionResult AuthenticateUser(String userEmail, String
userPassword)
{
    var users = (_cosmosDbService.GetItemsAsync<User>("SELECT *
FROM c WHERE c.partitionKey = 'user'")).Result;
    var user = users.FirstOrDefault(u => u.Email == userEmail);

    if (user == null || _kdfService.GetDerivedKey(userPassword,
user.PasswordSalt) != user.PasswordHash)
    {
        return BadRequest();
    }

    var token = GenerateJwtToken(user.id);
    user.Token = token;
    return Ok(new { token, user });
}

private string GenerateJwtToken(string username) {
    var tokenHandler = new JwtSecurityTokenHandler();
    var key = Encoding.ASCII.GetBytes(_jwtSecret);
    var tokenDescriptor = new SecurityTokenDescriptor
    {
        Subject = new ClaimsIdentity(new[] { new
Claim(ClaimTypes.Name, username) }),
        Expires = DateTime.UtcNow.AddHours(1),
```

```
        SigningCredentials = new SigningCredentials(new
SymmetricSecurityKey(key), SecurityAlgorithms.HmacSha256Signature)
    };
    var token = tokenHandler.CreateToken(tokenDescriptor);
    return tokenHandler.WriteToken(token);
}
```

Це найважливіші частини коду та конфігурації для реалізації серверної частини системи. Із детальним кодом можна ознайомитись по ссилці: https://github.com/daniilklementiev/Shelter-back-api

# РОЗДІЛ 3
# РЕЗУЛЬТАТИ ВИПРОБУВАНЬ

Результатом компіляції програмного коду серверної частини є робочий API, задокументований та доступний для тестування через Swagger. Випробування серверної частини проводилося з метою перевірки коректності реалізації основних функцій, таких як реєстрація, авторизація, створення та управління оголошеннями, а також завантаження зображень.

Основна серія випробувань проводилася з використанням наступних інструментів та середовищ:

- Swagger UI: Використовувався для інтерактивного тестування кінцевих точок API.
- Postman: Використовувався для тестування запитів та аналізу відповіді сервера.
- Azure Web App: Платформа для розгортання та хостингу API.

*Реєстрація та авторизація користувачів*

Початок роботи нового користувача з системою починається з реєстрації. Під час реєстрації користувач надає основні дані, такі як ім'я, електронну пошту та пароль. Ці дані перевіряються на коректність, хешуються та зберігаються у базі даних. Авторизація здійснюється за допомогою JWT токенів, що генеруються під час успішного входу користувача.

На рис. 1 наведено приклади запитів та відповідей для реєстрації та авторизації користувачів через Swagger UI.

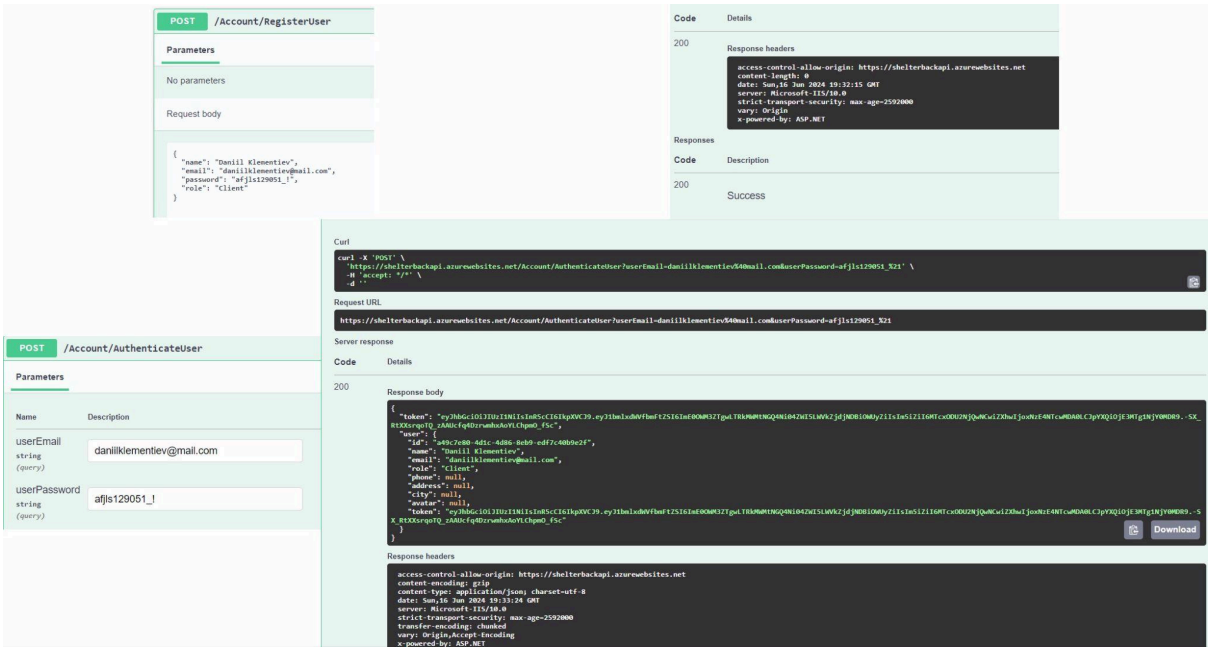


Рисунок 1 - Реєстрація та авторизація користувачів

*Створення та управління пропозиціями*

Користувачі системи можуть створювати оголошення, вказуючи основні деталі, такі як назва, опис, ціна, категорія та місто. Оголошення можуть редагуватися або видалятися автором. Усі запити щодо оголошень проходять через відповідні кінцеві точки API, задокументовані у Swagger.

На рис. 2 наведено приклади запитів та відповідей для створення, редагування та видалення оголошень.

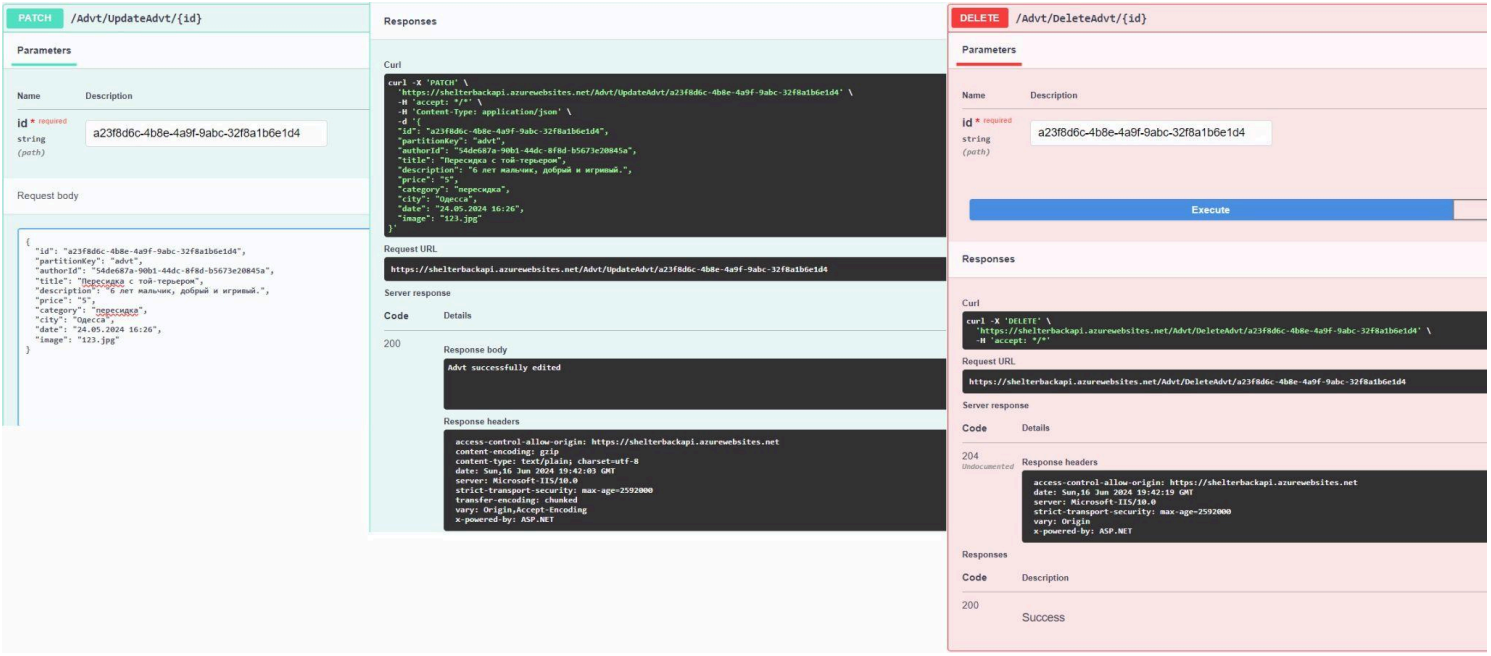


Рисунок 2 - Управління пропозиціями

*Завантаження зображень*

Для кожного оголошення користувач може завантажити зображення, яке зберігається у хмарному сховищі Azure. API підтримує завантаження файлів різних форматів, включаючи JPG та PNG, з обмеженням розміру файлу до 1 МБ.

На рис. 3 наведено приклади запитів та відповідей для завантаження та отримання зображень.

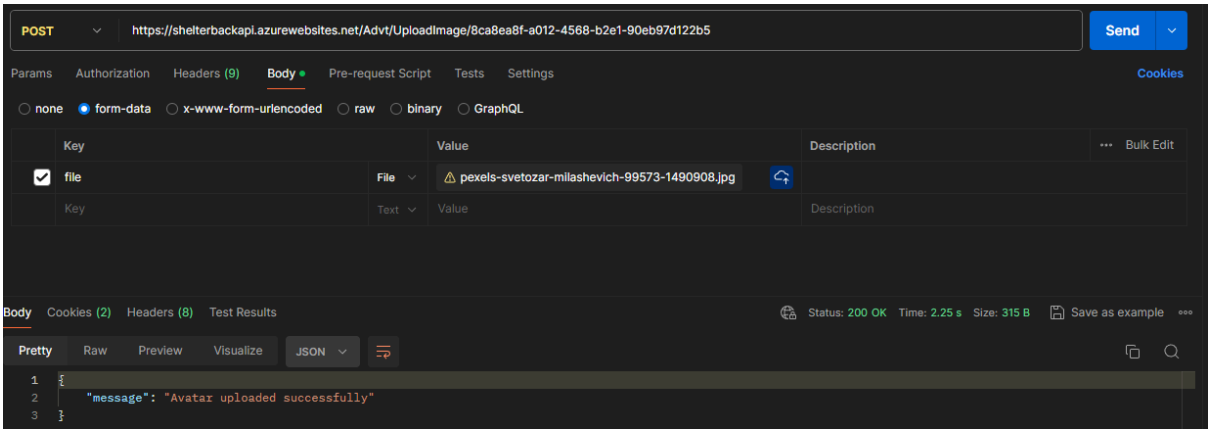


Рисунок 3 - Завантаження зображень

*Загальна функціональність системи*

Після успішної авторизації користувач отримує доступ до особистого кабінету, де може переглядати та редагувати свої персональні дані, переглядати статус своїх оголошень та взаємодіяти з іншими користувачами через внутрішню систему повідомлень.

Усі функції серверної частини, регламентовані технічним завданням, були реалізовані та випробувані. Поточна версія API, задокументована у Swagger, доступна за посиланням https://shelterbackapi.azurewebsites.net/swagger/index.html.

# ВИСНОВКИ

Проведено аналіз існуючих інформаційних систем для узгодження взаємодії надавачів та споживачів послуг, виявлено їхні переваги та недоліки. Основним результатом цього аналізу стало визначення ключових вимог до системи, яка повинна забезпечувати ефективну взаємодію між постачальниками послуг та споживачами з урахуванням аспектів безпеки та зручності використання.

Розроблено програмний комплекс, який включає в себе засоби реєстрації, авторизації та автентифікації користувачів, створення та управління пропозиціями надавачів та споживачів послуг, персональний кабінет, а також пошуковий механізм. Особлива увага приділялася забезпеченню безпеки персональних даних користувачів, включаючи шифрування даних та впровадження механізмів підтвердження особистості.

Систему було випробувано у різних умовах для оцінки її ефективності та надійності. Випробування проводилося з використанням Swagger UI для інтерактивного тестування кінцевих точок API, Postman для детального тестування HTTP-запитів, а також Azure Web App для розгортання та тестування серверної частини. Результати тестування підтвердили високу ефективність та надійність розробленої системи.

Впроваджено механізми захисту даних, які забезпечують безпечне зберігання та передачу персональної інформації користувачів. Це включає використання алгоритмів шифрування для захисту даних при їх передачі між клієнтською та серверною частинами системи.

Розроблена інформаційна система дозволяє ефективно координувати взаємодію між постачальниками послуг та споживачами, забезпечуючи надійну реєстрацію та автентифікацію користувачів, безпечне зберігання персональних даних, а також зручний та швидкий доступ до необхідних сервісів. Система має значний потенціал для застосування у різних галузях цифрової економіки, сприяючи підвищенню ефективності та зручності надання та отримання послуг.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. “Українці стали частіше користуватися інтернетом, 80% – онлайн щодня: соцопитування” // United Nations Development Programme URL: https://www.undp.org/uk/ukraine/press-releases/ukrayintsi-staly-chastishe-koristuvatysya-internetom-80-onlayn-shchodnya-sotsopytuvannya
2. Лященко А. А. Особливості реалізації стандартів доступу до баз геопросторових даних в середовищі універсальних СКБД. // Геоінформаційні технології у територіальному управлінні: матеріали II міжнар. наук.-практ. конф. Одеса: ОРІДУ НАДУ, 2015. с. 17-21 URL: http://www.oridu.odessa.ua/7/7/Book_new_2.pdf
3. Стратегія кібербезпеки України. Указ Президента України від 14 травня 2021 року // Урядовий кур'єр від 28.08.2021 — № 165. URL: https://zakon.rada.gov.ua/laws/show/447/2021/print
4. SHA-256 - Bitcoin Wiki URL: https://en.bitcoin.it/wiki/SHA-256
5. Cryptographic Hash Functions: Definition URL: https://www.investopedia.com/news/cryptographic-hash-functions/
6. Microsoft ASP.NET Documentation URL: https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-8.0
7. Swagger UI Documentation URL: https://swagger.io/docs/
8. Azure portal documentation URL: https://learn.microsoft.com/en-us/azure/azure-portal/