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

випускна кваліфікаційна робота бакалавра

# «ІНФОРМАЦІЙНА СИСТЕМА УПРАВЛІННЯ КОНТЕНТОМ ЕЛЕКТРОННОЇ КОМЕРЦІЇ»

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

Виконавці проекту

| № | П.І.Б. | Група |
|---|---|---|
| 1 | Кадочніков Д. С. | КН-П-192 |
| 2 | Крупиця Д. В. | КН-П-192 |
| 3 | Єфремова М. Ю. | КН-Д-191 |
| 4 | Тімченко В. О. | КН-П-192 |

Автор звіту

_____Тімченко Владилсав Олексійович

Одеса – 2023

# АНОТАЦІЯ

УДК 004.9
Т-41

Тімченко В. О. Інформаційна система управління контентом електронної комерції: Автореферат випускної кваліфікаційної роботи на здобуття бакалаврського рівня вищої освіти зі спеціальності «122 Комп'ютерні науки». - Одеса: ОТУШ, 2023 - 13 с.

Робота присвячена дослідженням, розвиненню та впровадженню інформаційної системи, головним призначенням якої є керування контентом електронної комерції. Додаткові функції системи включають реєстрацію, авторизацію та автентифікацію, засоби обмеження доступу до контенту, ведення журналів активності користувачів системи.

Актуальність роботи визначається швидкістю розвинення електронної комерції та просуванням електроних систем управління контентом українського виробництва, які матимуть змогу створювати конкуренцію на ринку електронної комерції. Особливу увагу звертає на себе той факт, що ці системи постійно розвиваються та модернізуються. Це дозволяє класифікувати задачі як такі, що формулюються в умовах швидкого розвинення інформаційних технологій. Додатково, актуальність диктується потребою впровадження мобільних інформаційних систем та реалізації у них покращених засобів захисту інформації, зокрема, персональних даних

Об'єктом дослідження у рамках даної роботи виступає алгоритм складання комплексної оцінки результатів за багатьма критеріями в умовах неповної визначеності вихідних умов. Предметом дослідження обрано різноманітність контенту, його зберігання, та видача цих даних за запитом, виконуючи всі умови(існують чи дані, чи є доступ до цих даних і т.п.). У якості задач дослідження було встановлене наступне:

- провести огляд інформаційних систем, які керують контентом електронної комерції.
- визначити програмні засоби, які дозволяють додавати дані про контент, зберігати їх, видавати за запитом.
- скласти програмне забезпечення, реалізувавши у ньому результати, досягнуті у попередніх задачах, провести випробування за різних умов та для різних пристроїв.
- впровадити у систему програмні засоби захисту даних для покращення показників інформаційної безпеки створеної системи.

Методи дослідження, що використані для досягнення поставлених задач, утворені суперпозицією методів системного аналізу, моделювання та когнітивної методології.

При складанні звіту використано 7 посилання на друковані та електронні джерела інформації.

ЗМІСТ

# ВСТУП

Розвиток і поширення інформаційних технологій супроводжується перенесенням великої кількості послуг у цифровий простір. Стратегія розвитку інформаційного суспільства в Україні визначає, що «системи управління контентом електронної комерції є необхідним інструментом соціально-економічного прогресу, одним з основних чинників інноваційного розвитку економіки» [1].

Інтернет комерція розвивається дуже швидко, тому кожна організації які займаються комерцією мають звернути на це увагу. Тому потрібно запропонувати їм сучасну систему управління контентом [2].

Швидкий розвиток електронної комерції потребує постійної модернізації систем управління контентом. Вона повинна ставати більш функціональною, безпечною та комфортною для користування [3]. Це актуалізує додаткові дослідження і розробки в зазначеній області.

В якості предмету даної роботи було обрано досить поширену систему управління контентом до функцій якої належать розміщення контенту від власника системи та інших організацій які хочуть розмістити свій контент, та пошук відповідних пропозицій з боку клієнтів. При розміщенні контенту має враховуватись вузька спеціалізація платформи.

Також функції переважної більшості подібних системи мають передбачати можливість збереження історії маніпуляцій з контентом, пошуку пропозицій, а також фактів їх прийняття чи відхилення. У перспективі - засобів оплати. Відповідно, це передбачає засоби авторизації та автентифікації, а також убезпечення персональних даних від несанкціонованого доступу інших користувачів.

Детальне технічне завдання до проекту наводиться далі. У рамках даного звіту розглядається клієнтська частина, що являє собою web-додаток. Звіт складається з трьох розділів у яких описується загальна характеристика системи, деталі створення клієнтської частини та результати її розгортання та тестування. Висновки відбивають результати, що досягнуті у рамках даної частини проекту, загальний висновок є сукупністю звітів усіх учасників проекту.

# ТЕХНІЧНЕ ЗАВДАННЯ ДО ПРОЕКТУ

**Загальний опис.**
Програмне забезпечення (ПЗ) призначене для створення розподіленої системи управління електронної комерції, що поділяються за різними ролями. Головна функція ПЗ полягає в наданні функціональної системи керування контентом для клієнтів-продавців. Додаткова функція ПЗ має дозволити залишати відгуки та оцінки щодо усіх видів товарів

**Призначення:**
ПЗ орієнтується на різні види узгодження користувачів: за належністю контенту до того чи іншого користувача, за видом послуги, за вартістю, за рейтингом тощо. ПЗ створюється за прикладом маркетплейсу, у той же час має складатись за принципом універсальності, мати змогу змінити цільове призначення.

**Вимоги до функціоналу:**
*Реєстрація користувачів:*
Користувач ПЗ має змогу зареєструватись за допомогою авторизаційних даних.
При реєстрації обов'язково зазначається засіб оперативного контактування (ЗОК) - телефон. На тестовому етапі, за умови ускладнення замовлення послуг телефонії (та/або СМС) можливе використання електронної пошти.
Також при реєстрації зазначається роль користувача: продавець або покупець. Одна особа має використовувати різні реєстраційні дані, якщо хоче бути і продавцем і покупцем.
Для можливості ввічливого спілкування між користувачами при реєстрації має зазначатись ім'я, за яким до користувача будуть звертатись.

*Особистий кабінет продавця*:
Основне робоче місце користувача ПЗ. Повинен забезпечити усі необхідні засоби для

- Розміщення пропозицій
- Встановлення обмежень щодо рейтингу споживача
- Перегляд історії

Особистий кабінет покупця
Так само, являє собою основний робочий засіб. Має реалізовувати можливості

- Перегляд історії покупок
- Пошук товарів(тільки актуальні дані, взяті з бази даних)
- Перегляд відгуків та рейтингу товарів і продавців

*Інструмент керування контентом*
ПЗ має надати інструмент керування контентом. Клієнт-продавець має мати можливість додавати товари, видаляти їх та редагувати. Усі ці дані належать лише одному клієнту и тільки він має доступ до змін цих даних. Також має бути інструмент для адміністраторів, якій сможе управляти як товарами, так і користувачами.

*Платіжний засіб*
ПЗ надає користувачу вибір засобів оплати. Це може бути оплата онлайн, або оплата після отримання товару.

**Архітектура**
ПЗ має бути спроектоване за клієнт-серверною архітектурою з розподіленими вузлами:
Backend - серверне програмне забезпечення, що забезпечує збереження та оброблення поточних даних, взаємодію з базою даних. Надає програмний інтерфейс взаємодії (API) з іншими вузлами
ClientApp (https://soundparadise.store) - додаток, доступний з веб-простору (сайт), що забезпечує візуальний інтерфейс користувачів усіх категорій.

Усі модулі ПЗ повинні мати засоби тестування функціональності.

**Технології**
Зберігання вихідних кодів ПЗ повинно здійснюватись за допомогою системи контроля версій (вибір конкретної системи узгоджується групою проекту). Структура файлів та каталогів повинна відбивати архітектуру ПЗ, на зразок:

- root
    - Api
    - Client

Для кожного з модулів рекомендується (з можливістю часткової зміни) наступні технології створення:

Backend - C#,.NET, MSSQL
Web - JS, HTML, CSS,або JS фреймворки, зокрема ReactJS, AngularJS

Вихідний код повинен оформлятися у відповідності до загальноприйнятих стандартів, на зразок стандартів GNU http://www.gnu.org/prep/standards/, у тому числі з додержанням вимог до коментування та документування коду.

# РОЗДІЛ 1
# ЗАГАЛЬНА ХАРАКТЕРИСТИКА СИСТЕМИ

У відповідності до технічного завдання (ТЗ) система була спроектована за клієнт-серверною розподіленою архітектурою. У складі системи було умовно відокремлено наступні архітектурні частини:

- «Mockup»,
- «Backend»,
- «Frontend».

Частина «Mockup» являє собою набір моделей та описів поведінки користувача. Модель подає інтерфейси взаємодії користувачів різного типу з інформаційною системою, траєкторії рухів та переходів між різними інтерфейсами. При створенні інтерфейсів застосовано принципи розумної навігації, включно із підходами UI/UX. Частина розміщена за адресою https://www.figma.com/file/kXnNSvll8JZYSx44PQyAbs/%D0%94%D0%B8%D0%BF%D0%BB%D0%BE%D0%BC?type=design&node-id=0%3A1&t=BHFerLjmWwv6IIPJ-1.

Частина «Backend» реалізована як централізований вузол збереження та оброблення даних, що створюються та модифікуються протягом життєвого циклу інформаційної системи. Дана частина включає у себе базу даних проекту та програмний інтерфейс (API) доступу до основних її функцій, зокрема, керування контентом, авторизації, узгодження профілів користувачів різного типу та/або різних пристроїв.

Частина «Backend» виконана на базі програмного забезпечення .NET та СУБД «MS SQL Server Management Studio». Додатково використан програмний пакет «Entity Framework»

Клієнтська частина системи «Frontend», призначена для надання користувачеві доступу до функцій системи через засоби візуального інтерфейсу. Також до функцій даної частини належить збереження локальних даних для можливості часткового використання системи без мережі передавання даних (offline). За наявності у пристрої користувача модулів визначення локації, засобами даної частини мають бути передбачені функції одержання та оброблення їх сигналів. Вихідні коди та ресурси даної частини розміщені на репозиторії https://github.com/soundparadise-team/SoundParadise.Web Подальший звіт присвячений детальному опису цієї частини.

З метою покращення безпеки, на сервері є багато сервісів-валідаторів та перевірок даних. Це зроблено з метою збереження цілісності даних, які зберігаються у базі даних. Також встановлено цикл життя для токенів, які через певний проміжок часу видаляються з бази.

# РОЗДІЛ 2
# РЕАЛІЗАЦІЯ СЕРВЕРУ

Сервер інформаційної системи реалізована з використанням наступних технологій:

- ASP .NET WEB API.
- Засоби розроблення: мова програмування C# та середовище
- JetBrains Rider,
- Додаткове ПЗ та засоби:набір технологій «Entity Framework Core», «Swagger», «Postman», «DocFX»

У складі серверу передбачено та реалізовано наступні функції:

- Обробка та валідація даних
- Інжектування сервісів
- Налаштування CORS
- Налаштування Identity Server
- Автентифікація,
- Авторизація,
- Реєстрація
- REST API
- Логування
- Конфігурація Production/Development
  Versioning
- Policies
- JWT Bearer Токенізація
- Session Cookies
- Віртуальний підпис
- Хешування паролів та карток
- Інтеграція платіжної системи
- Генерація документації через DocFX та Swagger

,

Основна задача серверу - це отримувати дані від кліента, валідувати, обробляти та віддавати відповідь, але спочатку треба налаштувати контроллер для подальшої праці із запитами. Усі приклади будуть на основі UserController. Спочатку впроваджуємо залежності:

```
[ApiVersion(ApiVersions.V1)]
[BaseRoute(ApiRoutes.User.Base)]
public class UserController : BaseApiController
{
    private readonly ImageService _imageService;
    private readonly IOptions<UrlsOptions> _options;
    private readonly UserAuthenticationDtoValidator _userAuthenticationDtoValidator;
    private readonly UserCreateDtoValidator _userCreateDtoValidator;
    private readonly IUserCrud _userCrud;

    public UserController(IUserCrud userCrud, UserAuthenticationDtoValidator userAuthenticationDtoValidator,
```

```
            UserCreateDtoValidator userCreateDtoValidator, IOptions<UrlsOptions> options, ImageService
imageService)
    {
        _userCrud = userCrud;
        _userAuthenticationDtoValidator = userAuthenticationDtoValidator;
        _userCreateDtoValidator = userCreateDtoValidator;
        _options = options;
        _imageService = imageService;
    }
```

Контролер приймає у якості перевизначення 2 атрібути:
[ApiVersion(“Версія API””] та [BaseRoute(“Назва контроллеру”]

Формат API:
{Сервер}/api/{Версія API}/{Контроллер}/{Метод контроллеру}

Перш за все треба визначити шаблони які перевизначають контроллери:

```
[ApiController]
public abstract class BaseApiController : ControllerBase
{

}

[AttributeUsage(AttributeTargets.Class)]
public class BaseRouteAttribute : RouteAttribute
{
    public BaseRouteAttribute(string template)
        : base("api/v{version:apiVersion}/" + template)
    {

    }
}
```

Після перевизначення контроллеру та інжектування необхідних сервісів треба описати кінцеві точки API (далі буде замінено словом ендпоінт). З їх допомогою клієнт може надсилати запити на сервер. У якості прикладу беремо реєстрацію:

```
 [HttpPost(ApiRoutes.User.RegisterUser)]
    [SwaggerResponse((int)HttpStatusCode.OK, "The user is successfully registered. Returns the
registered user object.",
        typeof(UserModel))]
    [SwaggerResponse((int)HttpStatusCode.BadRequest,
        "Invalid user registration data. Returns an error message with validation errors.",
        typeof(string))]
    public IActionResult RegisterUser([FromBody] UserCreateDto userCreateDto)
    {
        var validationResult = _userCreateDtoValidator.Validate(userCreateDto);
```

```
        if (!validationResult.IsValid)
        {
            var validationErrors = validationResult.Errors.Select(e => e.ErrorMessage).ToList();
            return BadRequest(new { errors = validationErrors });
        }

        var cartItemDtos = HttpContext.Session.Get<List<CartItemDto>>("Cart") ?? new
List<CartItemDto>();

        var user = _userCrud.RegisterUser(userCreateDto, cartItemDtos);
        return user == null!
            ? BadRequest(new { error = "Something went wrong while registering user" })
            : Ok(user);
    }
```

Ендпоінт у параметрі приймає скорочений об‘єкт користувача «UserDto». Також потрібна валідація не тільки зі сторони клієнта, а й сервера, яка дозволить обробляти відправлені дані, тому було прийнято рішення використати бібліотеку «FluentValidation» та зробити власні валідатори:

```
  public class UserCreateDtoValidator : UserDtoValidator<UserCreateDto>
{
    private readonly SoundParadiseDbContext _context;
    public UserCreateDtoValidator(SoundParadiseDbContext context)
        : base(context)
    {
        _context = context;

        RuleFor(u => u.Password)
            .NotEmpty().WithMessage("Password is required")
            .Must(UserValidationHelper.ValidatePassword).WithMessage(
                "Password must be at least 8 characters long and contain at least one lowercase letter,
one uppercase letter, and one digit");
    }
}

public abstract class UserDtoValidator<TUserDto> : AbstractValidator<TUserDto>
    where TUserDto : UserDto
{
    private readonly SoundParadiseDbContext _context;
    protected UserDtoValidator(SoundParadiseDbContext context)
    {
        _context = context;

        RuleFor(u => u.Name).NotEmpty().WithMessage("Name is required");
        RuleFor(u => u.Surname).NotEmpty().WithMessage("Surname is required");

        RuleFor(u => u.Username)
            .Custom((username, context) =>
            {
                if (_context.Users.Any(u => u.Username == username))
```

```
                context.AddFailure("Username is already taken");
            })
            .NotEmpty().WithMessage("Username is required");

        RuleFor(u => u.Email)
            .Must(UserValidationHelper.ValidateEmail).WithMessage("Invalid email format")
            .Must(email => UserValidationHelper.EmailIsNotTaken(email,
_context)).WithMessage("Email is already taken");

        RuleFor(u => u.PhoneNumber)
            .Must(UserValidationHelper.ValidatePhoneNumber)
            .WithMessage("Phone number must contain only numeric characters.")
            .Must(phoneNumber => UserValidationHelper.PhoneNumberIsNotTaken(phoneNumber,
_context))
            .WithMessage("Phone number is already taken");
    }

public static partial class UserValidationHelper
{
        public static bool ValidatePassword(string password)
    {
        if (string.IsNullOrWhiteSpace(password))
            return false;

        if (password.Length < 8)
            return false;

        var passwordRegex = PasswordRegex();
        return passwordRegex.IsMatch(password);
    }

       public static bool ValidateEmail(string email)
    {
        if (string.IsNullOrWhiteSpace(email))
            return false;

        var emailRegex = EmailRegex();
        return emailRegex.IsMatch(email);
    }

        public static bool EmailIsNotTaken(string email, SoundParadiseDbContext context)
    {
        return !context.Users.Any(u => u.Email == email);
    }

       public static bool ValidatePhoneNumber(string phoneNumber)
    {
        if (string.IsNullOrWhiteSpace(phoneNumber))
            return false;

        var phoneNumberRegex = PhoneRegex();
```

```
        return phoneNumberRegex.IsMatch(phoneNumber);
    }

        public static bool PhoneNumberIsNotTaken(string phoneNumber, SoundParadiseDbContext
context)
    {
        return !context.Users.Any(u => u.PhoneNumber == phoneNumber);
    }

    [GeneratedRegex("^(?=.*[a-z])(?=.*[A-Z])(?=.*\\d).{8,}$")]
    private static partial Regex PasswordRegex();
    [GeneratedRegex("^[^\\s@]+@[^\\s@]+\\.[^\\s@]+$")]
    private static partial Regex EmailRegex();
    [GeneratedRegex("^[0-9]*$")]
    private static partial Regex PhoneRegex();
}
```

Особливістю API є у тому що він викликає CRUD сервіси через абстракції, для запобігання перетинання залежностей (dependency inversion). Абстракції IUserCrud для сервісу UserCrud:

```
public interface IUserCrud
{
    bool CreateUser(UserCreateDto userDto);
    RequestResult UpdateUser(UserDto userDto);
    RequestResult DeleteUser(Guid id);
    List<UserModel> GetAllUsers();
    UserModel GetUserById(Guid id);
    UserDto GetUserDtoById(Guid userId);

    UserModel? GetUserByCredentials(UserAuthenticationDto authenticationDto);
    UserModel GetUserByToken(string token);
    UserModel RegisterUser(UserCreateDto userDto, List<CartItemDto> cartItemDtos);
    RequestResult ConfirmUser(Guid userId, string token);
    RequestResult UploadAvatar(IFormFile file, UserModel userModel);
    string GenerateConfirmationLink(UserModel user);
    string GenerateUniqueToken();
    RequestResult Logout(string token);
    RequestResult AuthenticateUser(UserAuthenticationDto authenticationDto);
}
```

Після реєстрації користувачу приходить лист на пошту для підтвердження аккаунту:

Рисунок 1: Надходження листа на пошту


Якщо акаунт не підвердити то система не дозволить користувачеві зайти в його обліковий запис:

Рисунок 2: Результат виконання ендпоінту

# РОЗДІЛ 3
# РЕЗУЛЬТАТИ ВИПРОБУВАНЬ

Результатом праці став повний REST інтерфейс який взамодіє із сессіями, станами та запитами. API має згенеровану DocFX документацію та Swagger:

Рисунок 3: Документація ендпоінтів сутності користувача


Після проведення дослідження був прийнятий підхід у хешуванні та шифруванні даних.
Паролі користувачей хешуються за алгоритмом HMAC-SHA512 для оптимальної швидкості, безпеки та сумісності, а картки шифруються за допомогою криптографічного примітиву IV та ключа які зберігаються у змінних оточення:

Рисунок 3: Зберігання паролів користувача

| | user_id | name | surname | password_hash | password_salt |
|---|---|---|---|---|---|
| 1 | 598b63cc-9ef8-46b6-8c48-00196beb41c6 | ХРИСТЯ | ЛЮТА | 0x948A75B80F1561EBE582A3B3A1546FA1D8E07E740F7BD7087F… | 0x2E72739C7D704724EEA0AAC7B98388DCBD1147F5BB29553EBF… |
| 2 | 449972fe-1460-488f-8e85-001e316afa47 | МИКИТА | БАНДЕРА | 0x948A75B80F1561EBE582A3B3A1546FA1D8E07E740F7BD7087F… | 0x2E72739C7D704724EEA0AAC7B98388DCBD1147F5BB29553EBF… |
| 3 | 2b73e68a-f533-48e4-bf59-011c13401d8f | ЛЕВКО | МАЗАЙЛО | 0x948A75B80F1561EBE582A3B3A1546FA1D8E07E740F7BD7087F… | 0x2E72739C7D704724EEA0AAC7B98388DCBD1147F5BB29553EBF… |

Рисунок 4: Зберігання картки користувача

| | card_id | user_id | encrypted_card_number | encrypted_expiry_date | encrypted_cvv |
|---|---|---|---|---|---|
| 1 | bdc2c6ae-996b-4f37-7a41-08db83841c42 | a1dd17a0-9f0c-4694-9ea3-428fe1f11973 | 0x8BDF0B077DC4755E2BCF756378AC31A1DEF4EA555087DF8AD2… | 0x40C032441DC911D457CCFF71D84714D2 | 0xF16546030820D4B18B92C5122F38AA01A |

Таким чином впроваджується безпека для запобігання витіку даних.

Для хешування використовується «HashingService», а для генерації віртуального підпису «EncryptionService».
У подальшому розвитку системи коли буде інтеграція платіжної системи яка дозволяє комунікацію у форматі «Host-To-Host», яку наприклад реалізує сервіс «LiqPay» токен картки буде  зберігатися також у базі у зашифрованому вигляді.

Проаналізувавши сучасне різноманіття платіжних сервісів, був обраний «Fondy» із «Client-To-Host» взаємодією для доказу концепції.
Таким чином оформляються замовлення:

Рисунок 5: Оформлення замовлення

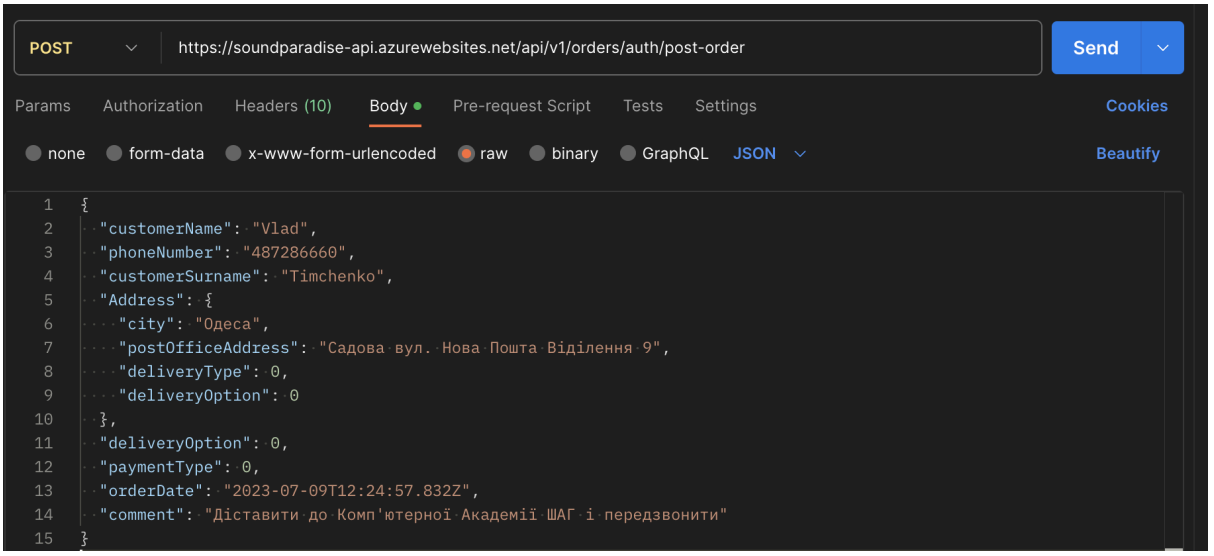


Рисунок 6: Замовлення створене

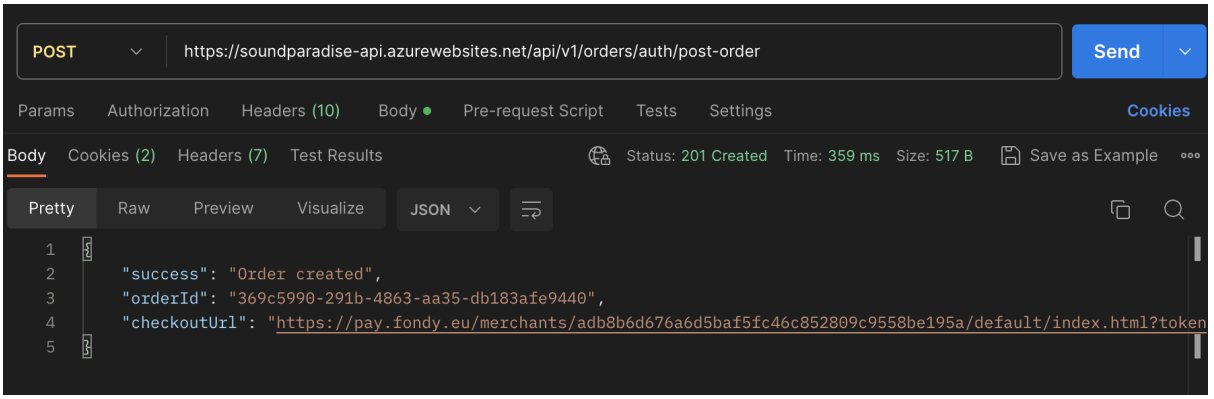

Рисунок 7: Оплата замовлення за допомогою сервісу «Fondy»

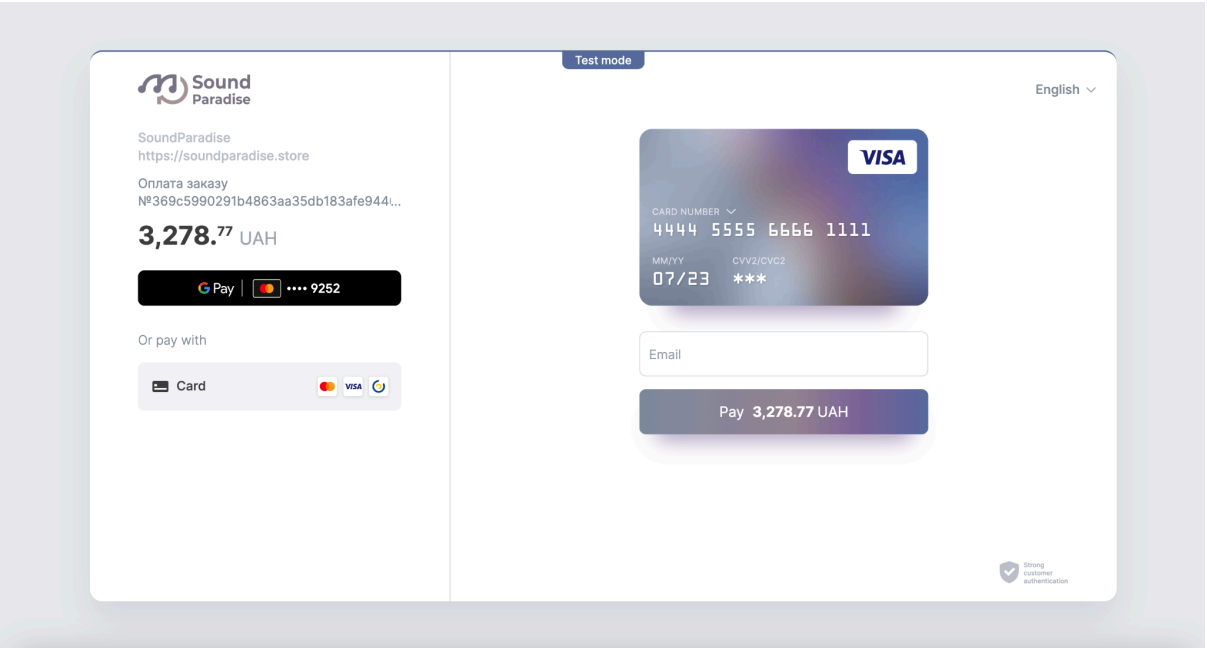


Рисунок 8 - Замовлення у базі

| | order_id | is_paid | comment | checkout_url | order_status | total_price | order_date |
|---|---|---|---|---|---|---|---|
| 1 | 369c5990-291b-4863-aa35-db183afe9440 | 0 | Діставити до Комп'ютерної Академії ШАГ і передзвонити | https://pay.fondy.eu/merchants/adb8b6d676a6d5baf5fc4… | 0 | 3278.77 | 2023-07-13 09:47:20.4796918 |

«Fondy» за допомогую callback відправрляє запит на сервер як тільки подія сплати виконалася і API змінює стан замовлення:

Рисунок 8 - Зміна стану

| | order_id | is_paid | comment | checkout_url | order_status | total_price | order_date |
|---|---|---|---|---|---|---|---|
| 1 | 369c5990-291b-4863-aa35-db183afe9440 | 1 | Діставити до Комп'ютерної Академії ШАГ і передзвонити | NULL | 1 | 3278.77 | 2023-07-13 09:47:20.4796918 |

Успішна зміна статусу замовлення.

У підсумку, всі функції API що були регламентовані технічним завданням було реалізовано та випробувано. Поточна версія застосунку, що використовує базу даних, доступна у репозиторії https://github.com/soundparadise-team/SoundParadise.Web/SoundParadise.Api

# ВИСНОВКИ

Проведено огляд інформаційних систем, які керують контентом електронної комерції, виявлено, що суттєве поширення мають системи, які пов'язані із маркетплейсом. Саме для таких систем керування контентом мають основне значення. У якості прикладу вибрано для реалізації маркетплейс(«Prom»).

Спроектовано, реалізовано та випробувано систему керування контентом електронної комерції. Система складена за розподіленою клієнт-серверною архітектурою та складається з трьох частин, розміщених за адресами:

- «Mockup» https://www.figma.com/file/gDr7qqU2ZRd0wt3giXqOnf,
- «Backend» https://github.com/soundparadise-team/SoundParadise.Web/SoundParadise.Api
- «Frontend» https://github.com/soundparadise-team/SoundParadise.Web/SoundParadise.Client

Сервер цієї системи може зберігати дані у базу данних, видавати їх за запитом. Також реалізована система версій що може відрізняти цей сервер від звичайного, та може зацікавити потенційного власника цієї системи який хоче розширюваний та гнучкий API. У систему впроваджено засоби покращення інформаційної безпеки: додаткове шифрування даних, що передаються комунікаційним каналом, за алгоритмом HmacSha256. Також вжито засобів підтвердження персональних даних користувача, зокрема, електронна пошта. Це ускладнює зловмисникам вживання підроблених (невласних) даних для дискредитації системи.

Систему було випробувано шляхом інсталяції у хмарні сховища та проведення тестових сеансів комунікації. Був налаштований власний домен та CI/CD використовуючи Azure та GitHub Workflows.

Результати випробування підтвердили загальну працездатність створеної системи.

# ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Стратегія розвитку інформаційного суспільства в Україні. Розпорядження Кабінету Міністрів України від 15 травня 2013 р. № 386-р. // Урядовий портал. URL: https://www.kmu.gov.ua/npas/246420577
2. 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
3. E-commerce в Україні: цифри, факти, перспективи розвитку онлайн-торгівлі // URL:https://www.site2b.ua/web-blog/e-commerce-v-ukraine-cifry-fakty-perspektivy-razvitiya-onlajn-torgovli.html
4. Впровадження CORS у ASP.NET застосунок // URL:https://learn.microsoft.com/en-us/aspnet/core/security/cors
5. Що таке .NET й чим займаються .NET-розробники? // URL:https://training.epam.ua/News/Items/301?lang=ru
6. Тенденції та перспективи розвитку інтернет-маркетингу в Україні // URL:http://www.spilnota.net.ua/ua/article/id-1086/
7. Електронна комерція // URL:https://azure.microsoft.com/ru-ru/solutions/ecommerce/
8. Веб-сайт електронної комерції у захищеному середовищі Служби додатків // URL:https://learn.microsoft.com/ru-ru/azure/architecture/web-apps/idea/ecommerce-website-running-in-secured-ase
9. Шифрування даних // URL:https://learn.microsoft.com/en-us/dotnet/standard/security/encrypting-data