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

Випускна кваліфікаційна робота бакалавра

«Інформаційна система керування авторськими мікроблогами (за прикладом сервісу Twitter)»

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

Виконавці проєкту

№	П.І.Б.	Група
1	Антонішин Д.	КН-П-191
2	Луста А.	КН-П-191
3	Синяков Д.	КН-П-191
4	Батула Д.	КН-А-191
5	Кечко О.	КН-Д-191
6	Потапенко К.	КН-Д-191

Автор звіту

_____ Антонішин Даніїл Олександрович

Одеса – 2023

АНОТАЦІЯ

УДК 004.9
Д-30

Антонішин Д.О. Інформаційна система керування авторськими мікроблогами (за прикладом сервісу Twitter): 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 системи, головним призначенням якої є взаємодія між користувачами та контентом від користувачів у вигляді мікроблогів. Додаткові функції системи включають реєстрацію, авторизацію, публікування та створення постів, коментування постів, механізм оцінки постів, засоби розповсюдження постів інших користувачів, ведення журналів активності користувачів системи.

Актуальність роботи визначається поширеністю задач, які ґрунтуються на створенні та публікації своїх думок у мікроблогах, на контактуванні та спілкуванні чи дискусій на різні теми. Особливу увагу звертає на себе той факт, що засобом використанні функціонала система керування авторськими мікроблогами виступає у вигляді кросплатформного мобільного додатка. З інтуїтивним та особливим інтерфейсом взаємодії користувача із даними.

Об'єктом дослідження у рамках даної роботи виступає алгоритм створення постів за різними критеріями в умовах неповної визначеності вхідних варіантів даних. Предметом дослідження обрано механізм створення та подальшого розвитку та обговорень з іншими користувачами. У якості задач дослідження було встановлене наступне:

- провести огляд інформаційних систем, що надають можливість створювати, публікувати, коментувати, ділитися, та оцінювати мікроблоги
- визначити програмні засоби (інтерфейси API), які дозволяють відправляти та отримувати дані користувача, Оброблювати та зберегти їх.
- 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 28 посилання на друковані та електронні джерела інформації.

ЗМІСТ

ВСТУП

Поширення персональної інформації про своє життя, думки чи реакції с кожним роком стають все більше популярними. Для когось це вже не просто хобі, а справжня робота. Все більше і більше інформації проходить через нашу стрічку у смартфоні. Тому створення українського схожого сервісу з притаманними для нас функціями, дизайном та особливостями натякнуло на потрібність такого сервісу

Сучасні мобільні пристрої — смартфони, планшети, всі вони надають доступ до різних месенджерів чи різних систем керування авторськими мікроблогами. Все глибше такі системи проникають у наше життя та нашу діяльність, створюючи окремі напрями для хобі, праці та просто для комунікації з іншими користувачами.

Водночас широкий вжиток інформаційних систем, особливо з поширенням інформації про своє життя або конфіденційну інформацію, потребує покращення інформаційної безпеки як щодо захисту персональних даних користувача.

В якості предмета даної роботи було обрано досить поширену систему керування авторськими мікроблогами до функцій якої належать розміщення мікроблогів, коментування та обговорення, що надходять з боку одних користувачів чи з боку інших.

Також функції переважної більшості подібних систем мають передбачати можливість додавання зображень, коментування постів користувача, можливість оцінити мікроблог, ділитися на своїй сторінці блогами інших, а також можливість наявності чата між користувачами додатка. Відповідно, це передбачає засоби авторизації та автентифікації, а також убезпечення персональних даних від несанкціонованого доступу інших користувачів.

Детальне технічне завдання до проєкту наводиться далі. У рамках даного звіту розглядається Web-API сервіс, що допомагає зв'язати та провести обмін між клієнтською та серверною частиною, створений для користувачів пристроїв з операційною системою «IOS» та «Android». Звіт складається з трьох розділів у яких описується загальна характеристика системи, деталі створення Web-API сервіс та результати її розгортання та тестування. Висновки відбивають результати, що досягнуті у рамках даної частини проєкту, загальний висновок є сукупністю звітів усіх учасників проєкту.

ТЕХНІЧНЕ ЗАВДАННЯ ДО ПРОЕКТУ

Загальний опис

Програмне забезпечення (ПЗ) для поширення інформації шляхом спілкування з іншими користувачами

Призначення

Надання можливості поширення та розповсюдження інформації в мережі Інтернет. Збереження даних та надання конфіденційності користувачам, забезпечуючи безпеку та захист особистої інформації

Вимоги до функціоналу:

- Реєстрація та авторизація
    - Реєстрація
    - Підтвердження особистості
    - Авторизація
- Особистий акаунт користувача
    - Особистий профіль
    - Зміна особистих даних
    - Зміна паролю
- Інформаційні пости
    - Створення постів
    - Поширення особистих постів
    - Поширення постів, що сподобалися
    - Поширення постів інших користувачів
    - Зберігання постів
    - Видалення постів
    - Коментування постів
- Підпис та підписники
    - Підпис на інших користувачів
    - Відписка від інших користувачів

Архітектура:

- Сервер бази даних - серверне програмне забезпечення, що забезпечує збереження та обробку поточних даних.

- Сервіс Web-API – серверне програмне забезпечення, що надає програмний інтерфейс для взаємодії (REST-API) з іншим програмним забезпеченням.

- Мобільний додаток керування контентом - додаток що встановлюється на мобільні пристрої та надає користувачам інтерфейс для взаємодії з функціоналом.

Технології:

- Для кожного з модулів рекомендується (з можливістю часткової зміни) наступні технології та методології створення:
  - Бази даних – MySQL
  - Сервіс Web-API – REST-API, ASP.NET Core, C#
  - Мобільний додаток – .NET MAUI, C#
- При розгортанні програмного забезпечення рекомендується задіяти хмарні технології
- При розробці програмного забезпечення необхідно задіяти зберігання вихідних кодів ПЗ повинно здійснюватись за допомогою системи контролю версій (вибір конкретної системи узгоджується групою проекту).
- 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

Відповідно до технічного завдання (ТЗ) система була відокремлена наступні архітектурні частини:

- “Design”,
- “Backend”,
- “Frontend”.

Частина **“Design”** являє собою список моделей та описом можливої поведінки користувача. Модель подає інтерфейси взаємодії із додатком та варіативність можливих дій користувача, траєкторії рухів. При створенні

інтерфейсів застосовано принципи інтуїтивної взаємодії з додатком, завдяки підходам UI/UX, Частина розміщена за посиланням: https://www.figma.com/file/DUaLAVhz9ZvboEtRyyWfJG/Wireframes?type=design&node-id=560-17692&mode=design&t=TPgFy5xHnjcabNPf-0

Частина "Backend" реалізована для централізації збереження та оброблення даних, що створюються та оновлюються. Дана частина містить базу даних, набір сервісів таких як збереження зображень у хмарі Amazon S3, сервіс шифрування даних та використання JWT токену для авторизації , та інструментів для обробки та управлінням даних у базі даних. Надає доступ до даних користувачів із бази даних MySQL. Ця частина розміщена в репозиторії за посиланням: https://github.com/plitkarka/Plitkarka.git

Частина "Frontend" реалізована у вигляді мобільного кросплатформного додатка, тобто на IOS та Android ОС, завдяки цьому користувачі можуть отримати комфортну взаємодію із нашим програмним забезпеченням. Надаючи лаконічний та мінімалістичний інтерфейс взаємодії користувачів із великим списком можливостей, функцій та цікавих реалізацій в проєкт. З клієнтської частини надається можливість користувачеві залишати, або змінити його дані які потім вирушать до Web-API сервісу для обробки та посилання на сервер. Web-API надає можливість з'єднати між собою серверною та клієнтською для обміну даних користувачів. Реалізує принцип REST-API архітектури для роботи з клієнтською частиною, завдяки використанню функціонала та можливостей HttpClient , даючи можливість відправляти дані для збереження та обробку на сервері так і для отримання даних користувачів на запит із клієнтського інтерфейсу. Також завдяки візуальному інтерфейсу дані із сервера відображаються у зрозумілому та упорядковному виді та можливість модифікації персональних даних. Ця частина розміщення в репозиторії за посиланням: https://github.com/plitkarka/client

Для збереження вихідних кодів було використано сервіс GitHub та технологію розподілену систему управління версіями Git. Також було реалізовано інтеграцію змін до вже робочого та готового до використання додатка, завдяки реалізації CI/CD сервісом GitHub Actions. Також для розгортання та перевірку на працездатність реалізовано Docker контейнер та віртуальну машину Amazon EC2 для автоматичного розгортання завдяки пайплайну із GitHub Actions.

РОЗДІЛ 2
РЕАЛІЗАЦІЯ WEB-API СЕРВІСУ КЛІЄНТСЬКОЇ ЧАСТИНИ

Web-API сервіс клієнтська частина інформаційної системи реалізована з використанням наступних технологій:

- Засоби розроблення: мова програмування C# (.NET 7.0) у середовищі Microsoft Visual Studio
- Додаткове ПЗ та засоби: бібліотека Newtonsoft для роботи із JSON об'єктами та файлами

У складі Web-API сервісу клієнтської частини передбачено та реалізовано наступні функції:

- Обробка даних із клієнтської частини
- Посилання запитів на збереження або модифікацію інформації на серверну частину
- Отримання відповіді від серверної частини
- Збереження інформації локально на пристрої

Перед початком роботи проводиться використання залежності (DI) та створення екземпляра із попередньою конфігурацією, завдяки самостійного файлу типу JSON із даними про конфігурацію . Початок використання сервісу із додатком супроводжується першими запитами на серверну частину, наприклад реєстрацією або авторизацію користувача мобільного додатку. При першому такому запиті до Web-Api сервісу проходить встановлена черговість дій в головному методі для всіх запитів

```
private async Task<HttpResponseMessage> SendRequestAsync(HttpMethod method, string url, HttpContent?
httpContent = null)    {
        await CheckAuthToken();
        var request = new HttpRequestMessage(method, CreateRequestURL(url));

        if (httpContent != null)
        {
            request.Content = httpContent;
        }

        var response = await _httpClient.SendAsync(request);

        if (response.StatusCode == HttpStatusCode.Unauthorized)
        {
            if (await GetNewAuthTokenPair())
            {
                return await SendRequestAsync(method, url, httpContent);
            }
            else
            {
                throw new Exception(await response.Content.ReadAsStringAsync());
            }
        }
        CheckStatusCode(response);
        return response;
    }
```

Перше що перевіряються це наявність авторизаційних токенів. В нашій реалізації використовуються JWT токени. Завдяки сервісу який працює із JSON проходить отримання токенів із файлу. Далі видаляється якщо є старий токен із заголовка запиту й при існуванні токена він підставляється у заголовок запиту та повертає пару токенів авторизації

```
 private async Task<TokenPairResponse> CheckAuthToken()
   {
      var authToken = await
JsonServices.DeserializeFromFileAsync<TokenPairResponse>(FileHandler.GetTokenPairFileLocati
on());
      _httpClient.DefaultRequestHeaders.Remove("AuthToken");

      if (authToken != null && authToken.AccessToken != "")
      {
         _httpClient.DefaultRequestHeaders.Add("AuthToken", authToken.AccessToken);
      }

      return authToken;
   }
```

JsonServices надає методи для роботи із JSON файлами для запису або читанню та повертає цю інформацію

```
public class JsonServices{
   public static async Task SerializeToFileAsync(string fileName, string json)   {
      using (FileStream fs = new FileStream(fileName, FileMode.OpenOrCreate))
      {
         byte[] serializedResult = Encoding.UTF8.GetBytes(json);
         await fs.WriteAsync(serializedResult);
         fs.Close();
      }
   }

   public static async Task<T> DeserializeFromFileAsync<T>(string fileName)   {
      string json;
      using (StreamReader r = new StreamReader(fileName))       {
         json = await r.ReadToEndAsync();
         r.Close();
      }
      return JsonConvert.DeserializeObject<T>(json);
   }

   public static async Task ClearFileAsync(string fileName)   {
      await SerializeToFileAsync(fileName, string.Empty);
   }
}
```

Далі створюється та конфігурується запит на сервер із зазначеного методу запиту (GET, POST, PUT, DELETE) та URL на який буду запит на сервер. Також перевіряється наявність різноманітного контенту який може йти із запитом. Далі завдяки HttpClient та методу SendAsync(request) посилається раніше створений запит. Після посилання відповідь проходить перевірку статус код на Unauthorized. Якщо наявний токен більше не є актуальними, то проходить запит на отримання нових токені, це зроблено для безпеки авторизації користувача

```
private async Task<bool> GetNewAuthTokenPair()   {
      var authToken = await CheckAuthToken();
      var token = new TokenRequest() { RefreshToken = authToken.RefreshToken, UniqueIdentifier = await
DeviceIdService.GetDeviceRegistrationId() };
      var response = await _httpClient.GetAsync(CreateRequestURL(AuthHandler.GetNewTokenPair(token)));

      Console.WriteLine(await response.Content.ReadAsStringAsync());
      if (response.IsSuccessStatusCode)
      {
         var tokenPair = JsonConvert.DeserializeObject<TokenPairResponse>(await
response.Content.ReadAsStringAsync());
         await JsonServices.SerializeToFileAsync(FileHandler.GetTokenPairFileLocation(),
JsonConvert.SerializeObject(tokenPair, Formatting.Indented));
         return true;
      }
      else
      {
         await JsonServices.ClearFileAsync(FileHandler.GetTokenPairFileLocation());
         return false;
      }
   }
```

Проходить вже описана операція отримання токенів та запит на URL який відповідає за оновлення токенів. У разі успішності запиту, нові токени записуються у файл та повертається true, якщо не успішно, то просто видаляються старі токени. Це зроблено якщо користувач довго не користувався додатком на пристрої. Якщо повертається false, то викликається виключення із відповіддю сервера

Далі відбувається перевірка відповіді на різні статус коди. У разі яких викликається виключення. Та після всіх операцій повертається відповідь сервера

```
private async void CheckStatusCode(HttpResponseMessage response)
   {
      if (response.StatusCode == HttpStatusCode.InternalServerError)
      {
         throw new Exception("Some errors occured on the server");
      }
      if (response.StatusCode == HttpStatusCode.Forbidden)
      {
         throw new Exception(await response.Content.ReadAsStringAsync());
      }
     …
```

Далі ця відповідь використовується у типізованих запитах до сервера. Наприклад GET. Який отримує URL запиту та тип, що повертається. До методу виконання передається метод запиту та сам URL, потім отримана відповідь десеріалізуєтеся із зазначеним типом значення та повертається

```
public async Task<T?> GetRequest<T>(string url)
 {
    var response = await SendRequestAsync(HttpMethod.Get, url);

    return JsonConvert.DeserializeObject<T>(await response.Content.ReadAsStringAsync());
 }
```

Приклад запиту на отримання користувача за ID в UserClient

```
public async Task<UserData> GetByIdAsync(Guid id)
 {
    return await GetRequest<UserData>(UserHandler.GetUserById(id));
 }
```

РОЗДІЛ 3
РЕАЛІЗАЦІЯ СЕРВІСІВ НА СЕРВЕРНІЙ ЧАСТИНІ

Розробленні сервіси відокремленої обробки інформації та додаткового функціонала надають такі можливості :

- Засоби збереження зображень користувача на відокремленому хмарному сервісі Amazon S3
- Підтвердження електронної пошти користувача завдяки 6-значному коду перевірки
- Можливість скидання пароля, та заміна на новий від облікового запису

Під час роботи зі серверною частиною з'явилась необхідність у відокремленому збереженні зображень які надсилає користувач та для пізнішого використання їх на клієнтській частині. Проаналізував варіанти хмарних сховищ було вибрано сервіс Amazon S3 Bucket. Завдяки Amazon API була проведена конфігурація та інтеграція її у серверну частину.

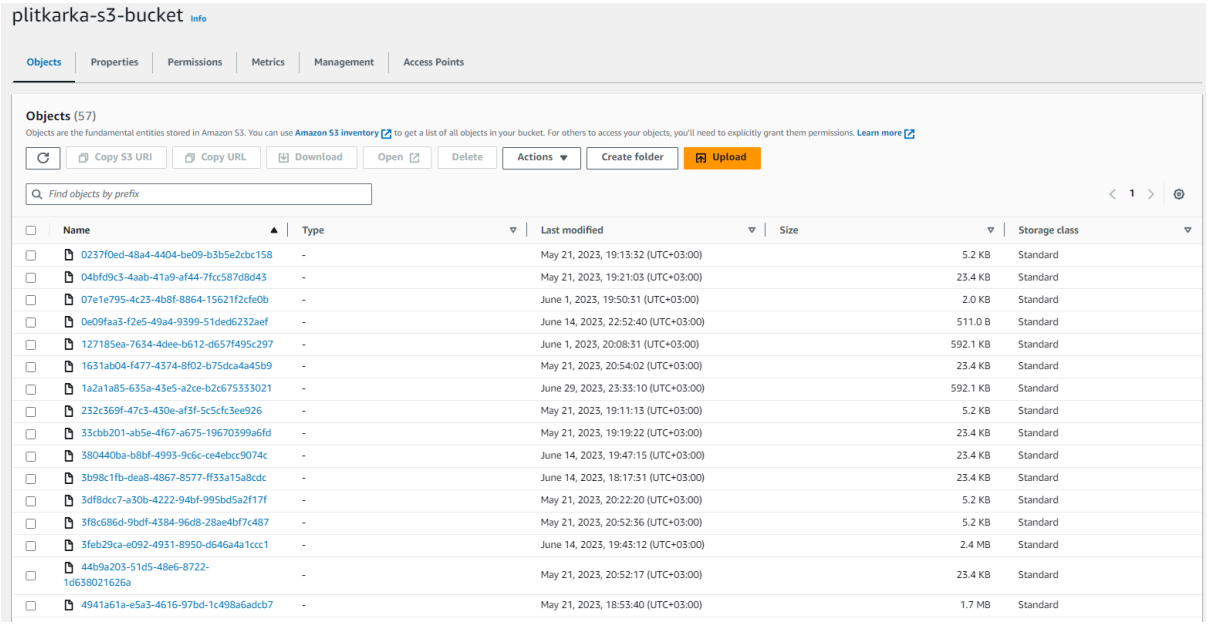
Попередньо було створено Bucket - місце де будуть зберігатися зображення. Після конфігурації S3 Bucket. Було проведено реалізацію механізму отримання та посилання зображень на зберігання.

```
public async Task<string> UploadImageAsync(IFormFile fileStream)   {
    if (fileStream.Length<=0)
    {
       throw new ValidationException("No file to load");
    }

    var extension = Path.GetExtension(fileStream.FileName);

    if (!_imageExtension.Contains(extension))
    {
       throw new ValidationException($"Wrong file extension: {nameof(fileStream)}");
    }

    var key = Guid.NewGuid().ToString();

    try      {
       var putRequest = new PutObjectRequest
       {
          BucketName = _s3Configuration.BucketName,
          Key = key,
          InputStream = fileStream.OpenReadStream(),
          ContentType = fileStream.ContentType,
          Metadata =
          {
             ["x-amz-meta-originalname"] = fileStream.FileName,
             ["x-amz-meta-extension"] = extension,
          }
```

```
        };
        await _s3Configuration.GetClient().PutObjectAsync(putRequest);
    }
    catch (S3ServiceException ex)
    {
        throw new S3ServiceException(ex.Message);
    }

    return key;
}
```

Після перевірок та компонуванні виконується запиту із зображенням на збереження в хмару.

Ось так виглядають всі зображення які попадають до хмари. До них не має прямого доступу, тільки після підтвердження користувача надається доступ до URL із зображення яке бачить користувачі. Попередньо всі назви зображень змінюються на їх хеш.



Після завершення розробки авторизації користувача потрібно підтвердити чи коректний адрес електронної пошти та, чи існує користувач. Завдяки сервісу який посилає заготовлені листи на пошту користувача зі кодом підтвердження. Попередньо використавши пошту нашого додатка було проведено конфігурацію та можливість розсилки листів користувачам, завдяки використанню бібліотеки MailKit та її функціоналу. Після підтверджень та конфігурації був реалізовано механізм посилки кодів на пошту, які попередньо були занесені в базу даних для подальшої перевірки на збіг.

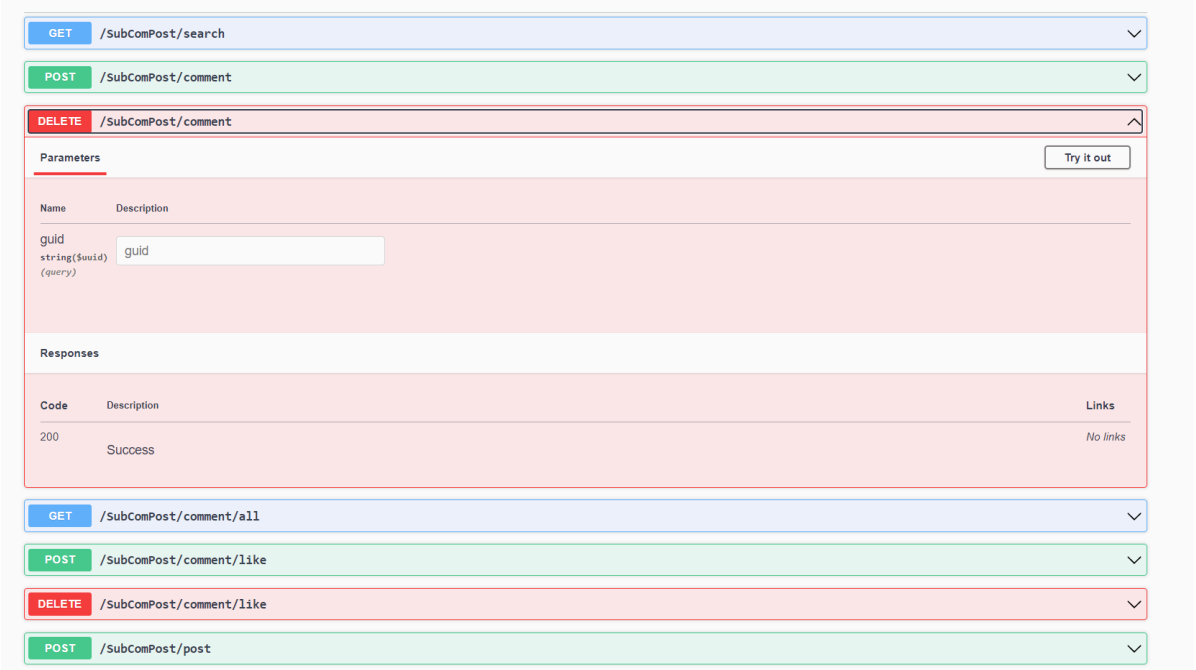
Так виглядає метод посилки листів на пошту користувача

```
public async Task SendEmailAsync(string email, string body, string subject)
 {
    if (_settings.ShouldSendEmails)
    {
        try
        {
            using var mail = new MimeMessage();

            mail.From.Add(new MailboxAddress(_settings.DisplayName, _settings.From));
            mail.To.Add(MailboxAddress.Parse(email));
            mail.Subject = subject;
            mail.Body = new BodyBuilder()
            {
                HtmlBody = body
            }.ToMessageBody();

            using var smtp = new SmtpClient();

            await smtp.ConnectAsync(_settings.Host, _settings.Port, SecureSocketOptions.StartTls);
            await smtp.AuthenticateAsync(_settings.UserName, _settings.Password);
            await smtp.SendAsync(mail);
            await smtp.DisconnectAsync(true);
        }
        catch (Exception ex)
        {
            throw new EmailServiceException(ex.Message);
        }
    }
 }
```

Завдяки раніше зробленому сервісу відсилання листів на пошту користувача з'явилася можливість скинути пароль, та отримати код підтвердження для користувача який ініціював зміну пароля. До пошти користувача надходить лист с кодом, який потрібно указати для вже зміни пароля та підтвердження пароля. Дані змінюються в базі даних та користувач отримує нову пару JWT токенів

РОЗДІЛ 4
РЕЗУЛЬТАТИ ВИПРОБУВАНЬ

Результатом перевірки програмного коду Web-API сервісу являється посилання запитів на серер, що можна перевірити завдяки окремому модулю Swagger із вебінтерфейс. Він надає згенерований інтерфейс на кожен запит який я додав до модуля. Завдяки використанню цього модуля, є можливість відправляти запити різної конфігурації та виду. Та також отримувати звіти про отриману інформацію як, наприклад: контент отриманий від сервера, заголовок відповіді та статус код.



Його використання є лише у розробці Web-API сервісу, виступаючи лише інструментом для перевірки працездатності сервісу.

Для тестування була включена також серверна частина зі схожим принципом роботи та обробки запитів

ВИСНОВКИ

Проведено огляд користувацьких додатків, що надають можливість користувачів створювати, редагувати, дивитися та залишати свої думки та оцінки на дані в додатку, виявлено, поширення мають додатки, надають можливість створювати пости, коментувати, та контактувати з іншими учасниками . Саме для таких випадків найбільш притаманним є створення мобільного додатку

Визначено програмні засоби (інтерфейси API), які дозволяють отримувати або надсилати дані на сервер, зберігати інформацію локально на пристрої

Спроєктовано, реалізовано та випробувано взаємодія користувачів з додатком та його функціоналом. Система складена з трьох частин, розміщених за адресами:

- “Design” https://www.figma.com/file/DUaLAVhz9ZvboEtRyyWfJG/Wireframes?type=design&node-id=560-17692&mode=design&t=TPgFy5xHnjcabNPf-0,
- “Backend“ https://github.com/plitkarka/Plitkarka.git,
- “Frontend” https://github.com/plitkarka/client.

У додатку використовується система авторизації завдяки JWT токенам, які допомагають працювати із додатком на різних пристроях та без потрібності у повторній авторизації. JWT забезпечує захист передачі даних про авторизацію користувача, завдяки використанню двох токенів Access та Refresh. Один для підтвердження авторизації користувача, а інший для оновлення цього токену.

Систему було випробувано шляхом інсталяції на фізичні пристрої та проведення тестових сеансів комунікації. Результати випробування підтвердили загальну працездатність створеного додатка.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. “ЧИСТА АРХІТЕКТУРА” Роберт Мартин. URL: https://www.rulit.me/books/chistaya-arhitektura-iskusstvo-razrabotki-programmnogo-obespecheniya-read-613012-1.html
2. Інформація про клас HttpClient URL: https://learn.microsoft.com/ru-ru/dotnet/api/system.net.http.httpclient?view=net-7.0
3. 4 Steps To Improve HTTP Request Performance in Dotnet MAUI / Xamarin URL: https://doumer.me/improve-http-request-performance-in-dotnet-maui-xamarin/
4. Арно Лоре. Проєктування веб-API: Навчальний посібник / Пер. с англ. Д. А. Беликова.– М.: ДМК Пресс, 2020. URL: https://library.samdu.uz/files/8668b2041ca85de4c292e22cd31537d8_Proektirovanie_veb-API_2020_Arno_Lore.pdf
5. Leonard Richardson and Mike Amundsen with a Foreword by Sam Ruby 2013 Leonard Richardson, amundsen.com, Inc., and Sam Ruby. URL: https://sd.blackball.lv/library/RESTful_Web_APIs_(2013).pdf
6. Величко О.М. Інтелектуальні інформаційні системи: структура і застосування. Підручник / Величко О.М., Гордієнко Т.Б. - Херсон. - 2022. - 728 с. ISBN: 978-966-289-552-0
7. Булгакова О.С. Методи та системи штучного інтелекту: теорія та практика. Навчальний посібник / Булгакова О.С., Зосімов В.В., Поздєєв В.О. - Херсон. - 2020. - 356 с. ISBN: 978-966-289-364-9
8. Гоменюк С.І. Математичне моделювання геометричних об’єктів у паралельних комп’ютерних системах. Монографія / Гоменюк С.І., Чопоров С.В., Аль-Атамнех Б.Г. - Херсон. - 2019. - 112 с. ISBN: 978-966-916-714-9
9. Величко О.М. Основи системного аналізу і прийняття оптимальних рішень. Підручник / Величко О.М., Гордієнко Т.Б. - Херсон. - 2021. - 672 с.ISBN: 978-966-289-553-7
10. Чемакіна О.В. Дизайн системи візуальної інформації. Навчальний посібник / Чемакіна О.В., Рубцов А.Л., Свірко В.О. - Херсон. - 2019. - 200 с. ISBN: 978-966-289-216-1
11. Шеховцов А.В. Комп’ютерні технології для дизайнерів. Навчальний посібник (стереотипне видання) / Шеховцов А.В., Полєтаєва Г.Н., Крючковський Д.О., Бараненко Р.В. - Херсон. - 2019. - 318 с.ISBN: 978-966-2393-08-8
12. Чайковська М.П. Концептуально-методологічні засади управління маркетинговими ІТ-проєктами в умовах цифрових трансформацій. Монографія / Чайковська М.П. - Херсон. - 2021. - 370 с.ISBN: 978-966-289-537-7