Міністерство освіти і науки України  
Харківський національний університет радіоелектроніки

Кафедра «Програмної інженерії»

Лабораторна робота № 4  
«РОЗРОБКА IoT КЛІЄНТА.»

Роботу виконав:

студент 3 курсу

факультету

«Комп’ютерні науки»

групи ПЗПІ-22-6

Іванов Д.С.

Перевірив:  
ст. викладач кат. ПІ  
Сокорчук І.П.

Харків 2025

| № | Дата | Версія звіту | Опис змін та виправлень |
| --- | --- | --- | --- |
| 1 | 24.08.2025 | 0.1 | Створено початковий варіант звіту |
| 2 | 24.08.2025 | 0.2 | Доповнено архітектурний опис, додано UML-діаграми та фрагменти коду |
| 3 | 24.08.2025 | 1.0 | Фіналізація та підготовка до завантаження |

ЗАВДАННЯ

Метою лабораторної роботи є розробка та тестування мобільного клієнта для програмної системи FoodCare Monitor System, що забезпечує моніторинг умов зберігання продуктів.

Для цього необхідно:

1. Створити мобільний застосунок для популярної платформи, визначивши ключові технологічні рішення.
2. Реалізувати інтерфейс користувача та логіку для взаємодії з REST API, розробленим на попередніх етапах.
3. Описати архітектуру мобільного застосунку, механізми синхронізації даних та обробки сповіщень.
4. Сформувати звітну документацію з детальним описом виконаної роботи.
5. Створити демонстраційний відеозапис функціонування застосунку та завантажити його на YouTube.
6. Оформити та завантажити звіт у PDF-форматі на навчальну платформу.
7. Оновити GitHub-репозиторій, додавши програмний код та звіт до нової директорії Task4.

3. ОПИС ВИКОНАНОЇ РОБОТИ

3.1. Архітектура мобільного застосунку Мобільний клієнт для системи FoodCare Monitor System був розроблений на кросплатформенному фреймворку .NET MAUI з використанням мови програмування C#. Такий підхід дозволяє створювати єдину кодову базу для платформ Android та iOS, значно прискорюючи розробку та підтримку.

Архітектура застосунку побудована з використанням патерну MVVM (Model-View-ViewModel). Цей підхід забезпечує чіткий поділ між інтерфейсом користувача (View), бізнес-логікою (ViewModel) та даними (Model). Таким чином, логіка застосунку є незалежною від представлення, що полегшує тестування та подальшу модифікацію.

Взаємодія з REST API серверної частини відбувається за допомогою сервісів, які інкапсулюють HTTP-запити. Для автентифікації користувачів використовуються JWT-токени, що зберігаються в захищеному сховищі пристрою.

3.2. Функціональні модулі Були реалізовані наступні ключові функціональні модулі:

* Автентифікація: Модуль для входу в систему та виходу з неї, що забезпечує безпечний доступ.
* Списки та деталі: Модуль для відображення списку складських приміщень та детальної інформації по кожному, включаючи дані сенсорів.
* Моніторинг: Модуль, що дозволяє в реальному часі відстежувати показники температури та вологості з підключених сенсорів.
* Сповіщення: Система сповіщень, яка інформує користувача про критичні відхилення показників.

3.3. Реалізація компонентів та логіки Користувацький інтерфейс створено за допомогою мови розмітки XAML, а бізнес-логіка реалізована в класах ViewModel. Це дозволяє легко підключати UI-елементи до властивостей та команд ViewModel'ів, що є ключовою перевагою MVVM-патерну. Завдяки цьому забезпечується автоматичне оновлення інтерфейсу при зміні даних.

Для асинхронних операцій, таких як завантаження даних із сервера, використовується синтаксис async/await, що запобігає блокуванню основного потоку UI і забезпечує плавну роботу застосунку.

4. ВИСНОВКИ

Під час виконання лабораторної роботи було успішно розроблено кросплатформенний мобільний застосунок на базі .NET MAUI для програмної системи FoodCare Monitor System.

Основні результати:

* Створено мобільний клієнт, що ефективно взаємодіє з бекенд-сервісами.
* Реалізовано ключовий функціонал, включаючи автентифікацію, перегляд даних складів та моніторинг сенсорів.
* Використання патерну MVVM та асинхронних операцій дозволило створити надійний, швидкий та зручний у використанні застосунок.
* Підготовлено повний звіт та супровідну документацію, що детально описує процес розробки та отримані результати.

ДОДАТОК А

Відеозапис та хронологічний опис

Хронологічний опис відеозапису:

* 0:00 - Вступ: огляд цілей лабораторної роботи та архітектури мобільного клієнта.
* 0:45 - Демонстрація процесу входу в систему з невірними даними.
* 1:15 - Успішна авторизація та перехід на головний екран.
* 1:30 - Огляд списку доступних складських приміщень.
* 2:05 - Перехід до сторінки деталей конкретного складу та перегляд його сенсорів.
* 3:10 - Демонстрація динамічного оновлення показників сенсорів у реальному часі.
* 4:00 - Сценарій "Порушення": показ сповіщення при виході показників за допустимі межі.
* 4:45 - Висновки та короткий підсумок.

Посилання на GitHub-репозиторій: [https://github.com/NureIvanovDmytro/apz-pzpi-22-6-ivanov-dmytro/tree/main/Lab4](https://www.google.com/search?q=https://github.com/NureIvanovDmytro/apz-pzpi-22-6-ivanov-dmytro/tree/main/Lab4)

ДОДАТОК Б

Програмний код

В.1 Код ViewModel для автентифікації користувача

C#

using CommunityToolkit.Mvvm.ComponentModel;

using CommunityToolkit.Mvvm.Input;

using FoodCare\_MobileClient.Services;

using FoodCare\_MobileClient.Views;

namespace FoodCare\_MobileClient.ViewModels

{

public partial class LoginViewModel : ObservableObject

{

[ObservableProperty]

private string username;

[ObservableProperty]

private string password;

[ObservableProperty]

private bool isBusy;

private readonly AuthService \_authService;

public LoginViewModel(AuthService authService)

{

\_authService = authService;

}

[RelayCommand]

public async Task LoginAsync()

{

if (IsBusy) return;

IsBusy = true;

try

{

var success = await \_authService.Login(Username, Password);

if (success)

{

await Shell.Current.GoToAsync($"//{nameof(WarehouseListView)}");

}

else

{

await Application.Current.MainPage.DisplayAlert("Помилка", "Невірне ім'я користувача або пароль.", "OK");

}

}

catch (Exception ex)

{

await Application.Current.MainPage.DisplayAlert("Помилка", $"Виникла помилка: {ex.Message}", "OK");

}

finally

{

IsBusy = false;

}

}

}

}

В.2 Код для отримання даних про сенсори

C#

using FoodCare\_MobileClient.DataTransferModels;

using System.Net.Http.Json;

namespace FoodCare\_MobileClient.Services

{

public class SensorService

{

private readonly HttpClient \_httpClient;

private readonly string \_baseUrl = "https://your-api-url.com/api";

public SensorService(HttpClient httpClient)

{

\_httpClient = httpClient;

}

public async Task<List<SensorDto>> GetSensorsByWarehouseIdAsync(int warehouseId)

{

try

{

var response = await \_httpClient.GetAsync($"{\_baseUrl}/warehouses/{warehouseId}/sensors");

response.EnsureSuccessStatusCode();

var sensors = await response.Content.ReadFromJsonAsync<List<SensorDto>>();

return sensors ?? new List<SensorDto>();

}

catch (Exception ex)

{

Console.WriteLine($"Error fetching sensors: {ex.Message}");

return new List<SensorDto>();

}

}

}

}