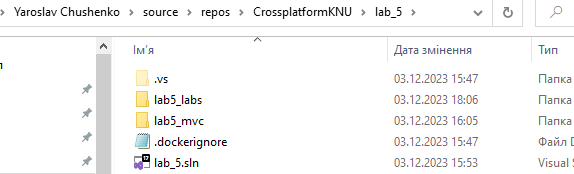
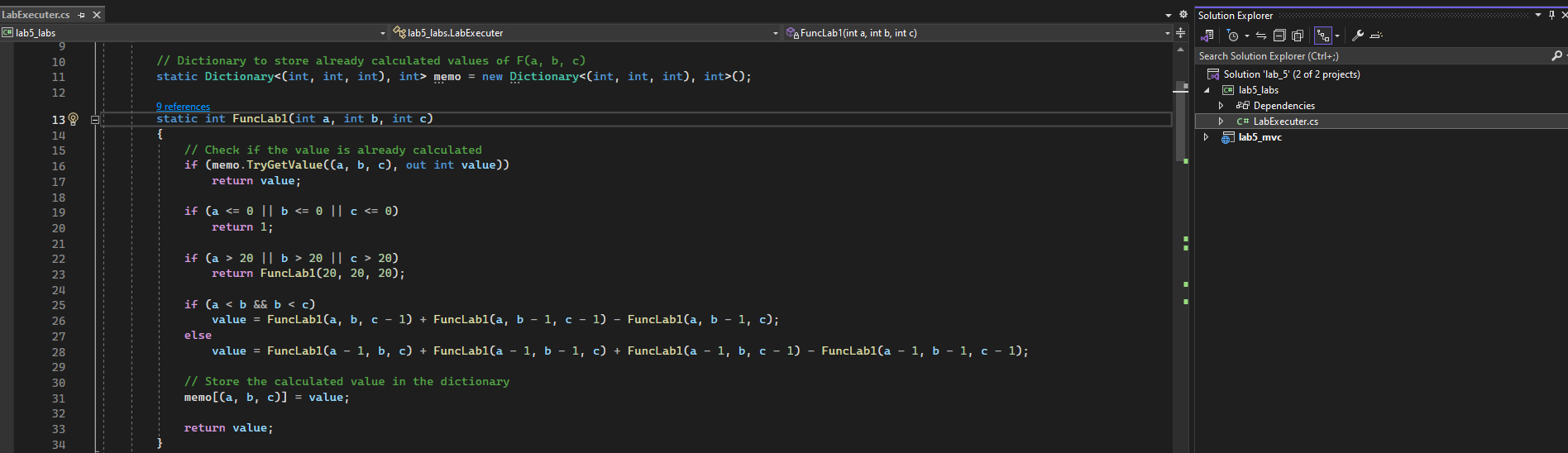
|  |  |  |  |
| --- | --- | --- | --- |
| **МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**  **КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені Тараса Шевченка**  **ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**  **Кафедра програмних систем і технологій**  **Дисципліна**  **«Кросплатформне програмування»**  **Лабораторна робота 4** | | | |
| **Виконав:** | Чушенко Ярослав Володимирович | **Перевірив**: | Петрівський В.Я |
| Група | ІПЗ-41 | Дата перевірки |  |
| Форма навчання | денна | Оцінка |  |
| Спеціальність | 121 |
| 2023 | | | |

**Загальний варіант – 54**Репозиторій на Github: [Посилання](https://github.com/PingvinAustr/CrossplatformKNU)  
 **Умова завдання:**Створити веб застосунок (ASP.NET Core MVC), що задовольнятиме наступним вимогам.

1. Складається з двох проектів
   1. Безпосередньо веб додаток
   2. Бібліотека класів, що дає змогу запускати практичні 1, 2 або 3
2. Розгортання виконати на операційній системі Linux (Ubuntu/Debian) за допомогою віртуальної машини
3. Має настпуний інтерфейс
   1. Сторінка вітання з описом роботи
   2. Сторінка логіну, реестрації та профілю користувача.
   3. 3 сторінки для кожної підпрограмми з полями для вводу та виводу, з описом того, що робить підпрограма
   4. Доступ до сторінок з підпрограмами можливий після авторизації
4. Сторінка реестрації має дозволяти ввести, щонайменше наступні поля
   1. Імя користувача (50 символів, унікальне)
   2. ФІО (500 символів)
   3. Пароль, та підвердження паролю (однакові, що найменше 1 цифра, 1 знак, 1 велика літера, не менше 8 символів, не більше 16)
   4. Телефон (формат Україна)
   5. Електронна адреса RFC 822 (https://datatracker.ietf.org/doc/html/rfc822#section-6.1)
5. Cторінка профілю не має можливості редагування тільки відображає інформацію що введена при реестрації
6. Для авторизації і аутенфікації слід використати будь-який сервер, що реалізує OAuth2.
   1. <https://identityserver4.readthedocs.io/en/latest/> - або аналоги (можливо найпростіший шлях)
   2. <https://www.okta.com/pricing/#customer-identity-products> (developer edition бзкоштовна) – (трохи часу на розібратись, але набагато легше в використанні через велику кількість готових бібліотек для підключення, можливий варіант 1 – 2 сервера на группу розгорнути)
   3. Google Identity – є така опція але мені здається, що тут доведеться занурюватись в протокол і одна з найскладніших
   4. Інші варіанти, що реалізують OAuth2
7. **Під час захисту потрібно буде показати розгорнуті віртуаьлні машини, сервер авторизації і що локальні налаштування працюють. Програму перевірятиму оффлайн.**
8. **Всі скрипти, або інші команди що слід виконати для розгортання застосунку слід додати до репозитарію.**

**Виконання:**1. Створимо новий solution з підтримкою docker, в якому розгорнемо MVC-проект та бібліотеку класів:  
  
  
2. До бібліотеки класів додамо клас для виклику задач, що були вирішені протягом ЛР 1-3:  
  
  
  
3. Створимо Dockerfile:  
#See https://aka.ms/customizecontainer to learn how to customize your debug container and how Visual Studio uses this Dockerfile to build your images for faster debugging.

FROM mcr.microsoft.com/dotnet/aspnet:8.0 AS base

USER app

WORKDIR /app

EXPOSE 8080

EXPOSE 8081

FROM mcr.microsoft.com/dotnet/sdk:8.0 AS build

ARG BUILD\_CONFIGURATION=Release

WORKDIR /src

COPY ["lab5\_mvc/lab5\_mvc.csproj", "lab5\_mvc/"]

RUN dotnet restore "./lab5\_mvc/./lab5\_mvc.csproj"

COPY . .

WORKDIR "/src/lab5\_mvc"

RUN dotnet build "./lab5\_mvc.csproj" -c $BUILD\_CONFIGURATION -o /app/build

FROM build AS publish

ARG BUILD\_CONFIGURATION=Release

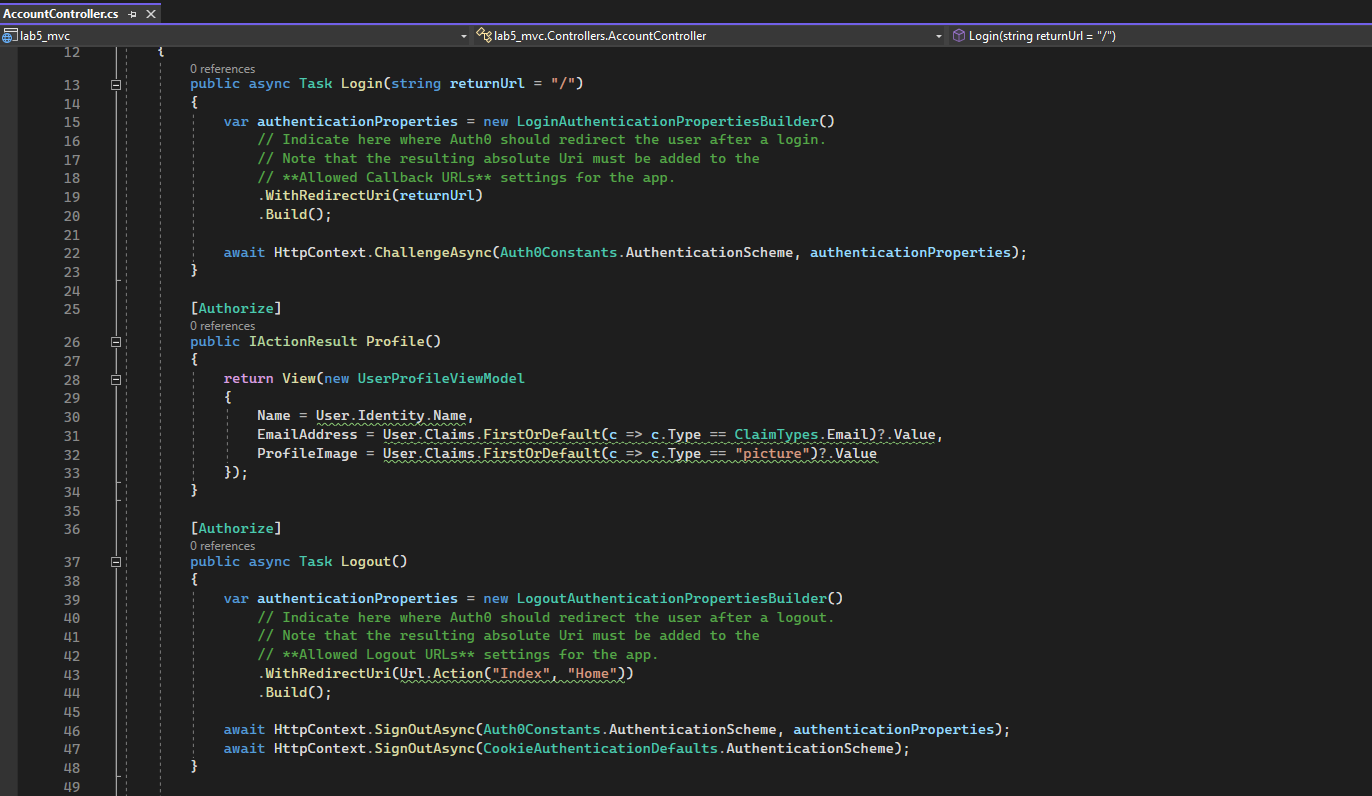
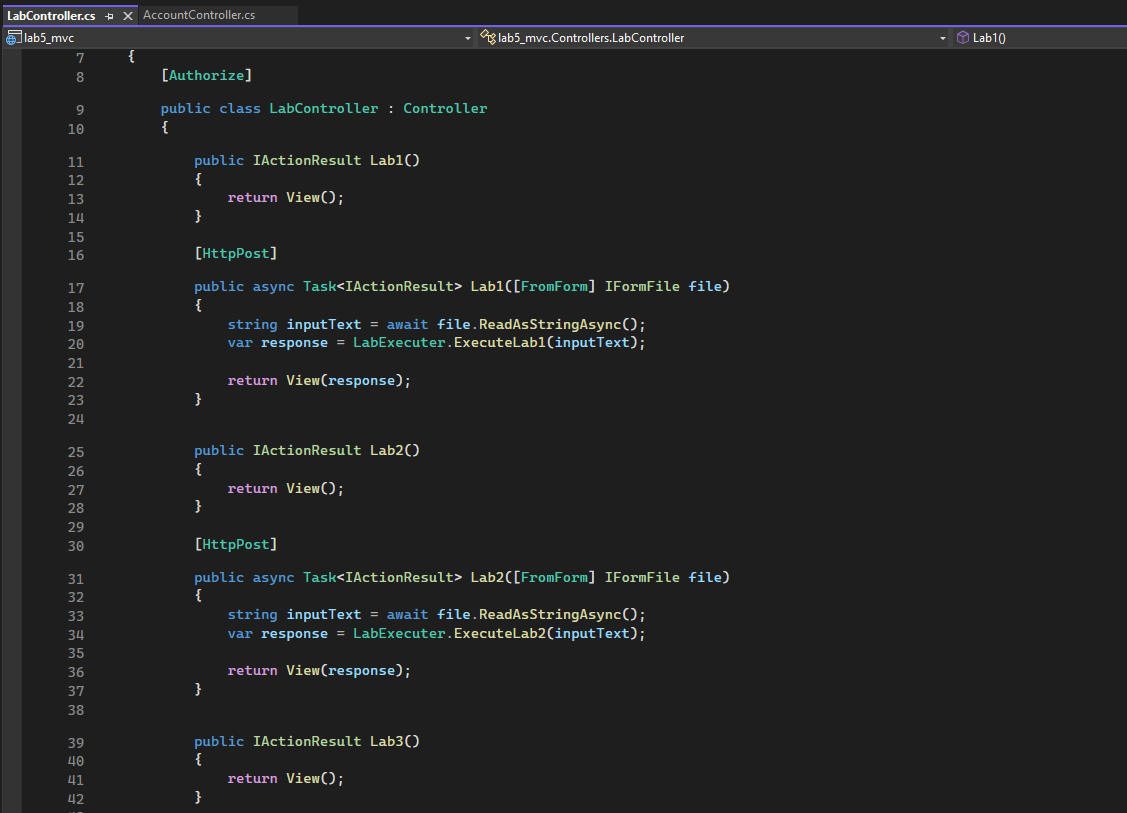
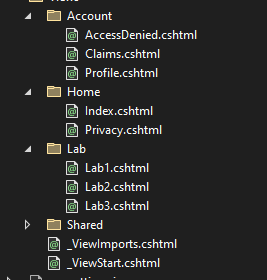
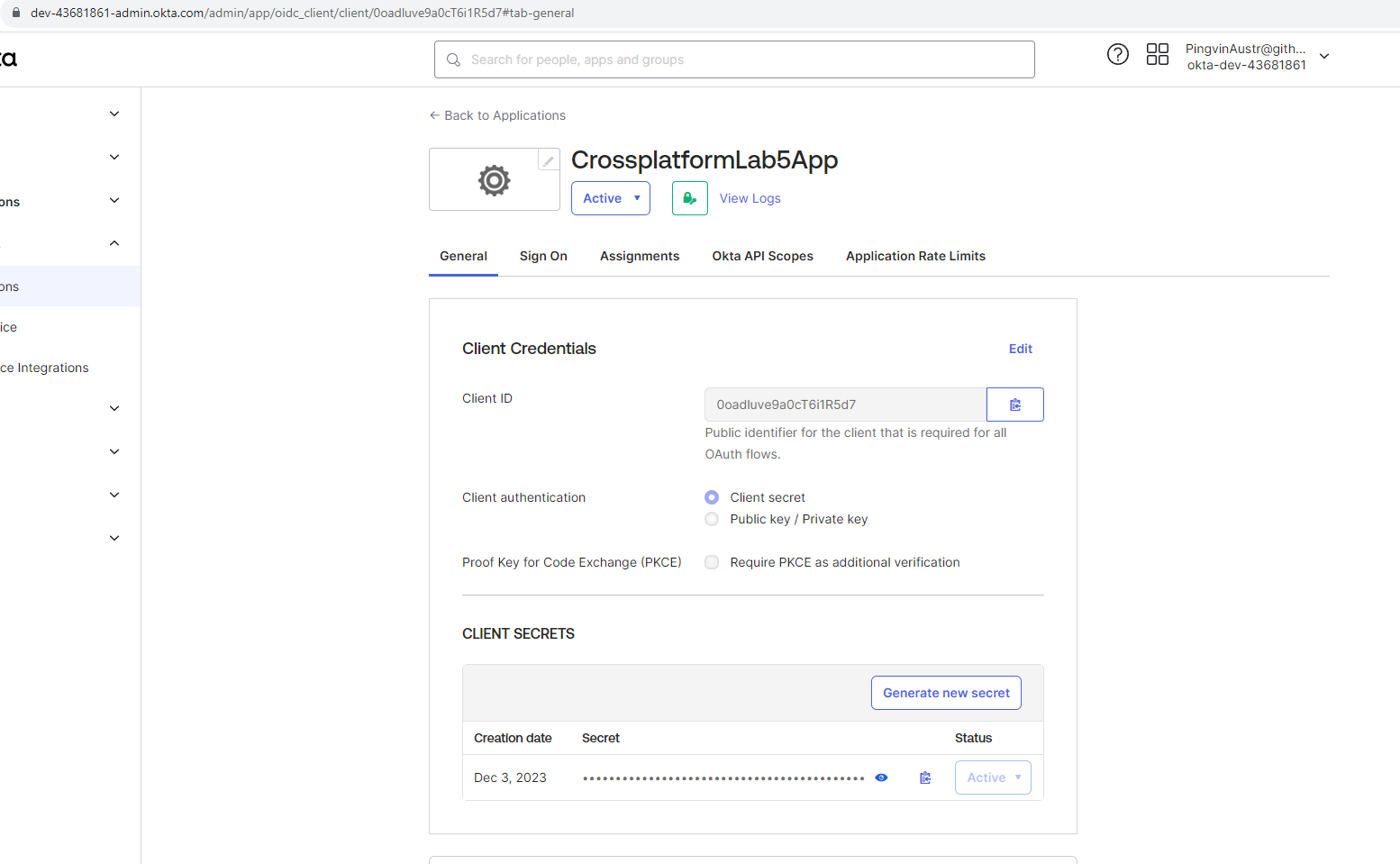
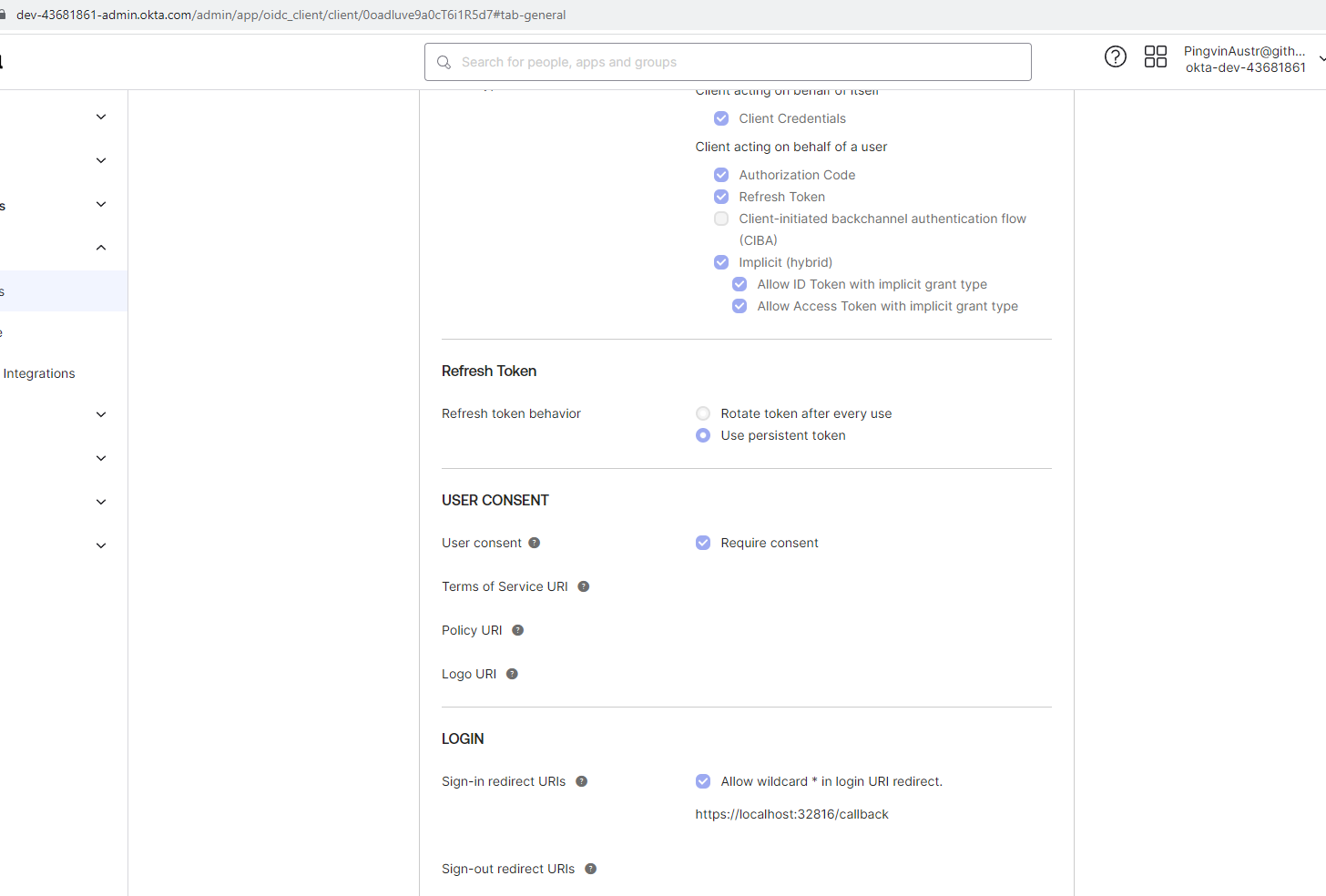
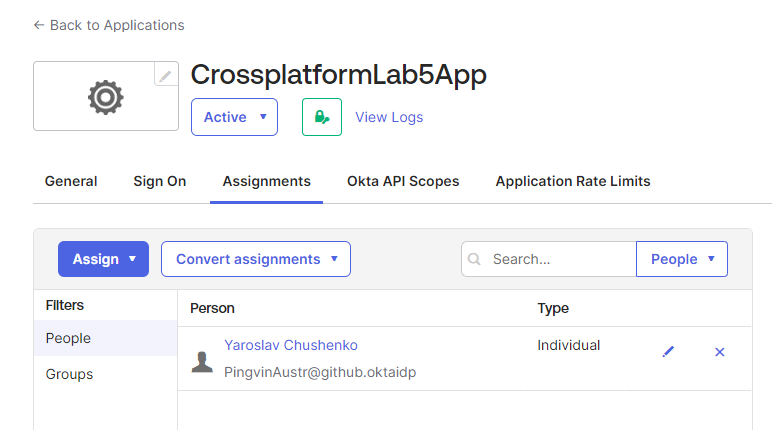
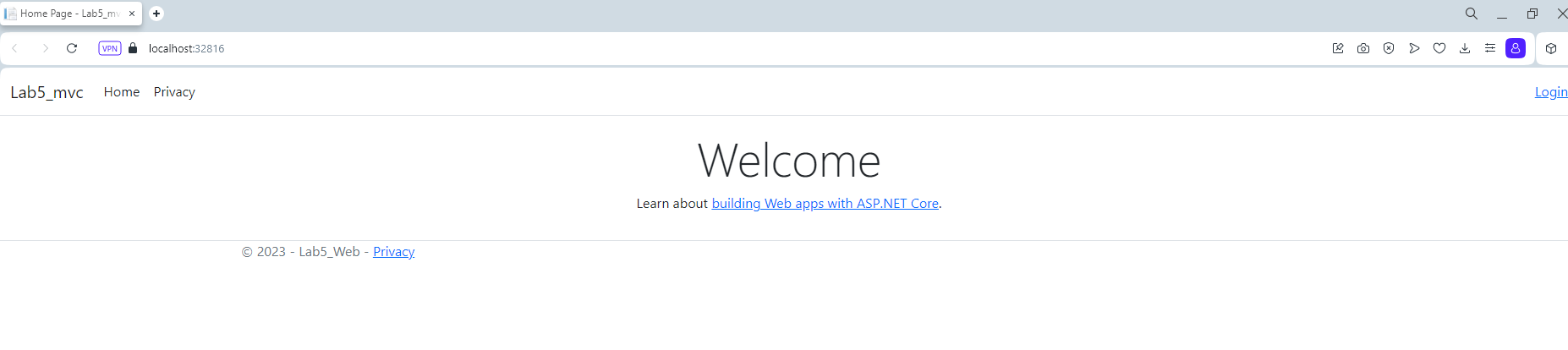
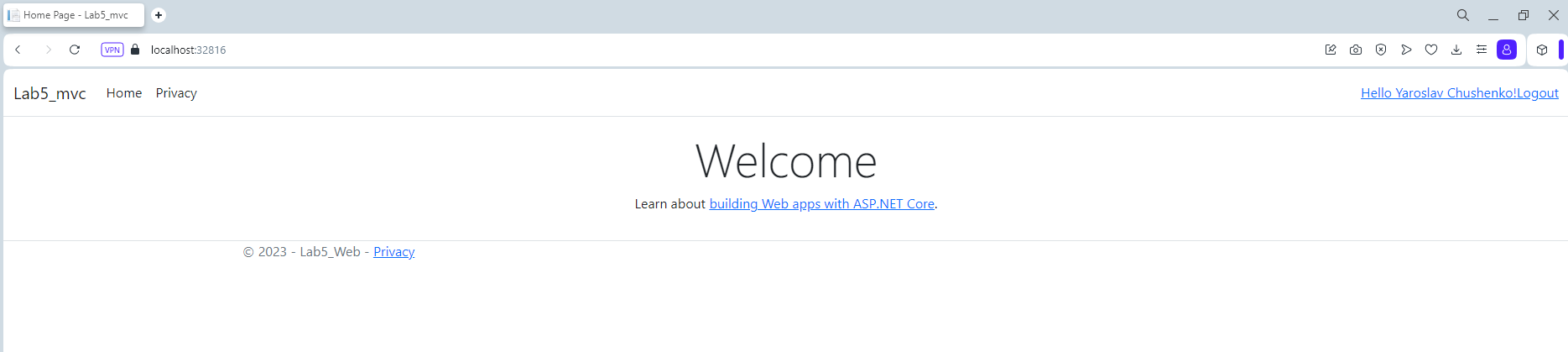
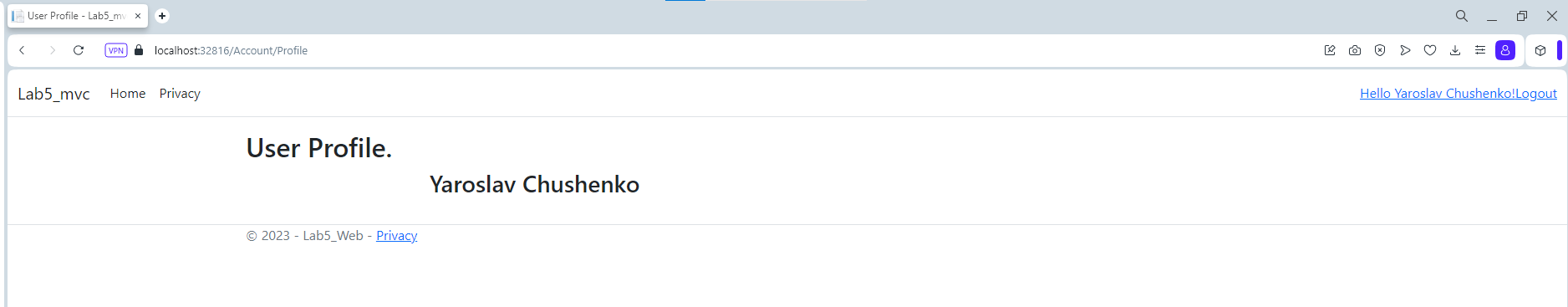
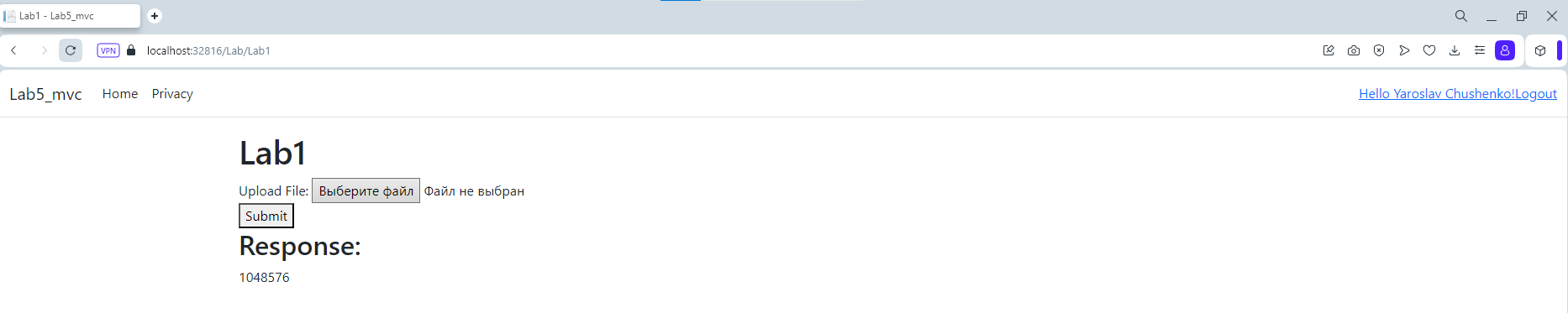
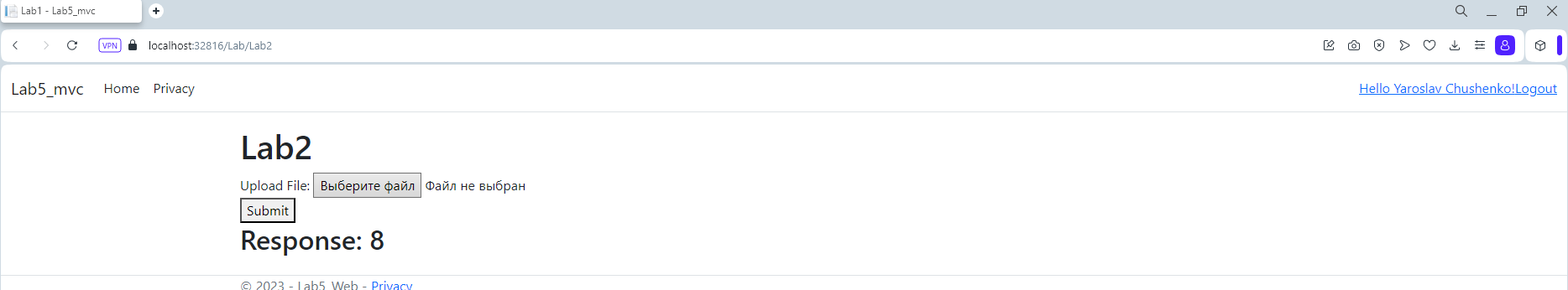
RUN dotnet publish "./lab5\_mvc.csproj" -c $BUILD\_CONFIGURATION -o /app/publish /p:UseAppHost=false

FROM base AS final

WORKDIR /app

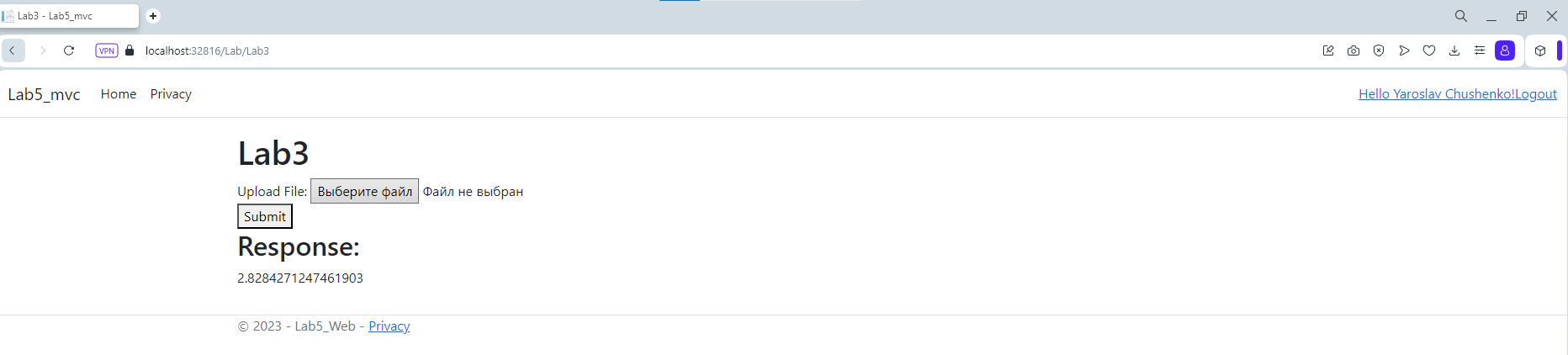
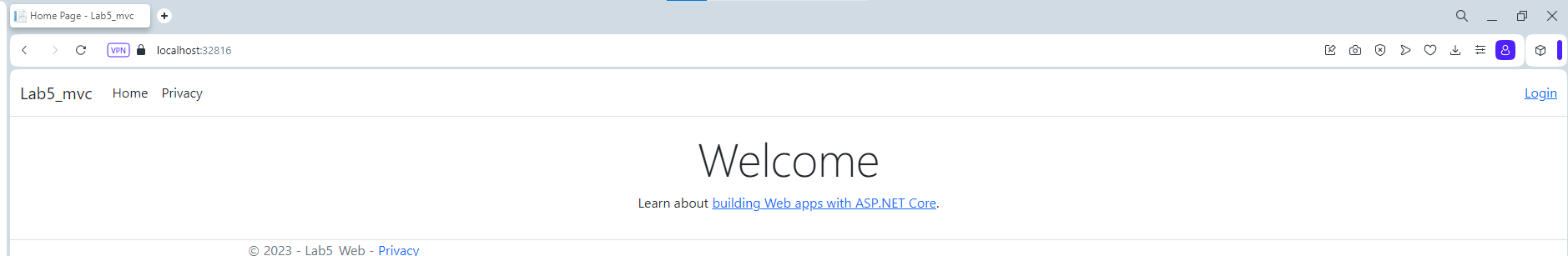
COPY --from=publish /app/publish .

ENTRYPOINT ["dotnet", "lab5\_mvc.dll"]

4. Додамо AccountController, що буде виконувати основні дії пов’язані з акаунтом:  
  
  
  
Для методів Login та Logout додамо параметри редіректів. Ці ж адреси треба додати до allowed-uri на Okta.  
  
  
5. Додамо LabController, що буде викликати задачі з бібліотеки класів:   
  
  
6. Додамо всі необхідні представлення, а саме – представлення для кожної лабораторної та авторизації:  
  
  
  
7. Зареєструємось на сервісі Okta та створимо застосунок:  
  
  
  
8. Виконаємо конфігурацію застосунку та додамо свого користувача до застосунку:  
  
  
  
  
9. Отримані від Okta ClientID, ClientSecret, OktaDomain додамо до файлу appsettings.json  
  
  
10. Запустимо програму  
  
  
  
11. Виконаємо авторизацію (Інакше сторінки будуть недоступні):  
  
  
12. Переглянемо профіль користувача:  
  
  
13. Далі послідовно запустимо кожну з ЛР 1-3 (до кожного скріншоту додано перевірені минулими ЛР вхідні та очікувані вихідні дані):  
  
  
Lab1:  
  
INPUT: 50 75 20  
OUTPUT: 1048576  
  
  
  
Lab2:  
  
INPUT: 106  
OUTPUT: 8  
  
  
  
Lab3  
  
INPUT:   
3

2 0

0 2

4 2  
OUTPUT: ~ 2,83  
  
  
  
Як можна побачити, результати вірні.  
  
  
14. Вийдемо з поточного акаунту через кнопку Logout:  
  
  
  
  
**Висновки:**

Протягом виконання поточної ЛР було отримано навички обробки авторизації користувачів за допомогою Okta. Була проведена інтеграція вищезазначеного програмного рішення в простий MVC-застосунок, що викликає раніше створені лабораторні роботи.