Лабораторная работа №16
Применение JWT

1 **Цель работы**

1.1 Научиться реализовывать разграничение прав доступа к БД с использованием JWT.

2Литература

2.1 Руководство по ASP.NET Core и C#. metanit.com – Текст : электронный // metanit.com, 2025. – URL: https://metanit.com/sharp/aspnet6/ - гл.13

3Подготовка к работе

3.1 Повторить теоретический материал (см.п.2).

3.2 Изучить описание практической работы.

4Основное оборудование

4.1 Персональный компьютер.

5Задание

Создать по необходимости сервисы для работы с БД, токенами, http-клиентом.

5.1 Генерация JWT

5.1.1 Реализовать в веб-API метод POST /auth/register для добавления в БД нового пользователя на основе логина и пароля.

Для передачи данных в метод создать класс LoginRequest (string Login, string Password).

Данные пароля хэшировать перед добавлением в БД. Роль по умолчанию – посетитель. Использовать таблицы пользователей и ролей из ЛР №15.

5.1.2 Реализовать в веб-API метод POST /auth/login для получения токена при аутентификации (на основе корректной пары логина и пароля пользователя из БД).

Для возврата данных из метода создать класс TokenResponse (string Token, string? RefreshToken).

В claims токена записать id, login, role пользователя. При создании токена записать в него издателя, потребителя, времени жизни, ключ безопасности.

5.1.3 Протестировать регистрацию пользователя и генерацию токена (токен проверить в VS / jwt.io или аналоге).

5.2 Разграничение прав доступа к API-методам в зависимости от авторизации

5.2.1 Добавить в веб-API в builder.Services возможность аутентификации с использованием токена (с валидацией издателя, потребителя, времени жизни, ключа безопасности).

5.2.2 Добавить в веб-API контроллеры со следующими методами (настроить доступ только для авторизованных пользователей, где это требуется):

- /profile – доступ в личный кабинет (требует авторизацию, возвращает данные о текущем (авторизованном) пользователе)
- /films – получение списка фильмов (не требует авторизации)

5.2.3 Протестировать доступ к методам в зависимости от роли в postman (вкладка Authorization, Type: Bearer, Token: сгенерированный в веб-API).

5.3 Разграничение прав доступа к API-методам в зависимости от роли

5.3.1 Добавить в веб-API контроллеры со следующими методами (настроить доступ только для авторизованных пользователей по указанным ролям):

- /tickets/{id} - проверка билетов (получение данных билета по id для билетера и посетителя)
- POST: /users добавление данных пользователя (для администратора)

5.3.2 Протестировать доступ к методам в зависимости от роли в postman (вкладка Authorization, Type: Bearer, Token: сгенерированный в веб-API).

5.4 Применение JWT в клиентском приложении

5.4.1 Выполнить в клиентском приложении (консольном или оконном) тестирование методов веб-API, которые не требуют авторизацию.

5.4.2 Выполнить в клиентском приложении (консольном или оконном) тестирование методов веб-API, которые требуют авторизацию, добавив заголовок в httpClient.

5.5 Применение refresh-токена

5.5.1 Добавить в БД таблицу RefreshToken(id, userId, token, expires, isRevoked).

5.5.2 Модифицировать метод /auth/login, чтобы кроме токена записывать в БД и возвращать refresh-токен.

5.5.3 Добавить в веб-API метод POST /auth/refresh для получения нового токена на основе переданного refresh-токена.

5.5.4 Модифицировать один из методов в клиентском приложении, чтобы при 401 (срок жизни токена истек), получить новый токен по refresh-токену и повторно выполнить требуемый запрос.

6Порядок выполнения работы

6.1 Выполнить все задания из п.5.

6.2 Ответить на контрольные вопросы.

7Содержание отчета

7.1 Титульный лист

7.2 Цель работы

7.3 Ответы на контрольные вопросы

7.4 Вывод

8Контрольные вопросы

8.1 Что такое JWT и для чего он предназначен?

8.2 Из каких трех основных частей состоит JWT и какое назначение каждой части?

8.3 В чем различие между использованием симметричного и асимметричного алгоритма для подписи JWT??

8.4 Что указывается в разделе claims в JWT?

8.5 Для чего используется refresh-токен?