Практическая работа №3
Разграничение прав доступа на уровне REST API

1 Цель работы

1.1 Научиться выполнять разработку web-API для доступа к БД.
1.2 Научиться работать с JWT.

2 Литература

2.1 https://metanit.com/sharp/aspnet6/13.2.php

3 Подготовка к работе

3.1 Повторить теоретический материал (см.п.2).
3.2 Изучить описание практической работы.

4 Основное оборудование

4.1 Персональный компьютер.

5 Задание

5.1 Разграничение прав доступа в контроллерах

5.1.1 Добавить в БД таблицы Role(RoleId, Name) и User(UserId, Login, Password, RoleId). Настроить первичные ключи, связи, автоинкремент, добавить 3 роли (администратор, менеджер, посетитель) и для каждой роли по 2 пользователя.

5.1.2 Создать контекст БД и модели данных для пользователей, ролей, фильмов, сеансов, билетов.

5.1.3 Создать контроллеры для фильмов, сеансов и залов. Используя атрибуты уровня класса и метода разрешить в контроллерах анонимный доступ, доступ всем авторизованным пользователям и доступ пользователям с определенной ролью/ролями:

- все пользователи (включая неавторизованных) могут просматривать информацию о фильмах и фильме, сеансах и сеансе,
- администратор может просматривать и редактировать всю информацию,
- менеджер может редактировать информацию о фильмах и сеансах,
- посетитель может добавлять билет.

Варианты указания доступа для нескольких ролей:

```
[AuthorizeRoles("Administrator", "Assistant")]
[Authorize(Roles = "Administrator,Assistant")]
```

Чтобы роли хранились централизованно, можно описать их отдельном классе:

```
public static class Role
{
   public const string Administrator = "Administrator";
   public const string Assistant = "Assistant";
}
```

5.2 Генерация токена

5.2.1 Создать API, который использует JWT для аутентификации. Настроить в конфигурации ключ, issuer и audience и применить их для настройки JWT.

5.2.2 Реализовать сервис, который генерирует токен на основе логина и роли пользователя, переданных в параметрах (эти данные должны записываться в Claims). Время жизни токена: 15 минут.

5.3 Создание контроллера для авторизации

5.3.1 Добавить в приложение контроллер AccountController

5.3.2 Добавить в контроллер метод для авторизации, который принимает в параметрах объект User, проверяет корректность данных и возвращает код ошибки с информацией об ошибке или токен, если ошибки не было. Генерацию токена протестировать.

5.4 Авторизация в клиентском приложении

5.4.1 Добавить оконное приложение, в котором реализовать интерфейс для ввода логина и пароля.

5.4.2 Создать в оконном приложении сервис для авторизации, обращающийся к API.

5.5 Обращение к методам, требующим авторизацию

5.5.1 Добавить в приложение сервисы для доступа к фильмам, сеансам, билетам.

5.5.2 Все сервисы, которые требуют авторизации, должны принимать в параметрах конструктора токен и добавлять его в заголовки объекта типа HttpClient.

5.5.3 Реализовать вызов методов сервисов в оконном приложении (вызвать только часть методов для демонстрации доступа неавторизованных пользователей, авторизованных пользователей, пользователей с определенной ролью).

6 Порядок выполнения работы

6.1 Выполнить все задания из п.5.

6.2 Ответить на контрольные вопросы.

7 Содержание отчета

7.1 Титульный лист

7.2 Цель работы

7.3 Ответы на контрольные вопросы

7.4 Вывод

8 Контрольные вопросы

8.1 Какие атрибуты можно указать у методов REST для настройки доступа для авторизованных и неавторизованных пользователей?

8.2 Для чего используется JWT?

8.3 В чем отличие между авторизацией с использованием cookie и с использованием JWT с точки зрения безопасности?

8.4 Какие настройки можно указать при создании токена?

8.5 Почему для авторизации следует использовать метод POST?