Лабораторная работа №15
Разграничение прав доступа

1 **Цель работы**

1.1 Научиться реализовывать разграничение прав доступа к БД.

2Литература

2.1 Руководство по Entity Framework Core 9 и C#. metanit.com – Текст : электронный // metanit.com, 2025. – URL: https://metanit.com/sharp/efcore/ - гл.5

3Подготовка к работе

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

3.2 Изучить описание практической работы.

4Основное оборудование

4.1 Персональный компьютер.

5Задание

5.1 Проектирование БД пользователей через наследование

5.1.1 Создать проект UsersLibrary типа «Библиотека классов (Майкрософт)», выбрать платформу .NET 9. Добавить в проект пакеты для работы с БД SQLite.

5.1.2 Добавить в проект 3 директории: UsersTPH, UsersTPT, UsersTPC. В каждой создать набор моделей для хранения данных пользователей:

- пользователь (id, логин, пароль),
- посетитель (телефон, баланс на карте и заблокирован ли пользователь),
- билетер (ФИО, зарплата (необязательный)),
- администратор (телефон, email, зарплата (необязательный)).

5.1.3 Настроить для каждого набора моделей модели и контекст согласно указанному в названии проекта варианту наследования:

- названия БД UsersTPH.sqlite / UsersTPT.sqlite / UsersTPC.sqlite,
- для создания БД в конструкторе вызывать метод Database.EnsureCreated(),
- в зависимости от варианта наследования указать аннотации у классов-наследников или явно указать метод наследования.

5.1.4 Проверить результат в DBeaver, сравнить полученные схемы.

5.2 Создание таблиц для разграничения прав доступа пользователей

5.2.1 Добавить в БД MSSQL таблицу ролей пользователей CinemaUserRole:

- идентификатор (целое число, автоинкрементное значение, PK)
- название роли (строка длиной до 20 символов)

Созданную таблицу заполнить значениями Администратор, Билетер, Посетитель.

5.2.2 Добавить в БД MSSQL таблицу пользователей CinemaUser:

- идентификатор (целое число, автоинкрементное значение, PK)
- логин (строка длиной до 50 символов)
- хэш пароля (строка длиной до 200 символов)
- количество попыток неверно ввести пароль (по умолчанию – 0)
- дата разблокировки (необязательная)
- идентификатор роли (целое число, FK к таблице ролей пользователей)

5.2.3 Добавить в БД MSSQL таблицу привилегий CinemaPrivilege:

- идентификатор (целое число, автоинкрементное значение, PK)
- название (строка длиной до 100 символов)

В созданную таблицу добавить данные о привилегиях:

- доступ в личный кабинет,
- проверка билетов,
- просмотр списка фильмов,
- добавление пользователей,
- редактирование пользователей.

5.2.4 Добавить в БД MSSQL таблицу привилегий CinemaRolePrivilege для связи ролей с привилегиями и заполнить ее:

- доступ в личный кабинет – все,
- проверка билетов – билетер,
- просмотр списка фильмов – билетер и посетитель,
- добавление пользователей – администратор,
- редактирование пользователей – администратор.

5.3 Создание сервиса для аутентификации

5.3.1 Создать проект AuthLibrary типа «Библиотека классов (Майкрософт)», выбрать платформу .NET 9. Добавить в проект пакеты для работы с БД MSSQL.

5.3.2 Добавить в проект класс AuthService, в который добавить:

- закрытый метод хэширования пароля,
- открытый метод регистрации нового пользователя по логину и паролю (роль по умолчанию для новых пользователей: посетитель). Если такой логин есть – возвращать false, если удалось зарегистрировать – возвращать true,
- открытый метод аутентификации пользователей по логину и паролю. Если пользователь найден, возвращать его, иначе – null. Реализовать блокировку пользователя на минуту после трех некорректно введенных паролей,
- открытый метод получения роли пользователя по логину,
- открытый метод получения списка привилегий по логину,
- открытый метод получения списка привилегий по роли.

5.3.3 Создать оконное приложение, в котором реализовать возможность регистрации нового пользователя после ввода логина и пароля (пароль должен храниться в БД в хэшированном виде).

5.4 Аутентификация

5.4.1 Добавить в приложение класс UserSession с:

- открытым свойством CurrentUser типа Пользователь (Пользователь — тип данных из модели), сделать set приватным,
- открытым методом void SetCurrentUser(пользователь) для присваивания значения свойству CurrentUser,
- открытым методом void Clear() для присваивания значения null свойству CurrentUser.

Для того, чтобы сделать класс UserSession реализующим паттерн синглтон, добавить в класс следующий код:

```
private static readonly UserSession _instance = new();
private UserSession () {}
```

public static UserSession Instance => _instance;

Для использования методов класса UserSession использовать следующий код:

UserSession.Instance.Метод(...)

5.4.2 Добавить в оконное приложение возможность авторизации по логину и паролю. При авторизации полученный пользователь должен записываться в CurrentUser, а на главной форме должен отображаться его логин.

5.4.3 После авторизации отображать кнопку «Выйти», при нажатии на которую очищать данные пользователя.

5.5 Разграничение прав доступа

5.5.1 Реализовать скрытие отображение кнопок с требуемыми пунктами меню в зависимости от привилегий роли пользователя (перечислены в п.5.2.4).

5.5.2 Реализовать возможность смены роли пользователя администратором в интерфейсе редактирования пользователей.

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 Как изменить настройки подключения к БД в клиентском приложении?

8.2 Какими способами можно обеспечить хранение пользователей и ролей пользователей в БД (отобразить в виде ERD)?

8.3 Что такое «авторизация»?

8.4 Что такое «регистрация»?