Курсов проект по ПС – Програма за управление на потребители и студенти

**Имена:** Александър Тенев  
**Фак. номер:** 501222019  
**Административна група:** 49Б  
**Дисциплина:** Програмни Среди  
**Тема:** WPF MVVM приложение за управление на потребители

**Съдържание**

1. Увод
2. Анализ на съществуващи разработки
3. Проектиране
4. Реализация
5. Потребителско ръководство
6. Заключение
7. Литература
8. Приложение

**1. Увод**

Настоящият проект цели създаването на настолно WPF приложение по модел MVVM за управление на потребители и студенти. Приложението поддържа role-based влизане, разглеждане и добавяне на потребители, списъци със студенти и други функционалности, които илюстрират реална употреба на Entity Framework, MVVM архитектура и слоев модел на разработка.

**2.Анализ на съществуращи разработки**

Целеви техники/технологии и приложения:

1. Entity Framework Core (ORM)
   * Положителни страни:
     + Автоматично управление на база данни (създаване, миграции).
     + Лесно генериране на уникални идентификатори (ValueGeneratedOnAdd).
     + Интеграция със SQLite за локално съхранение (идеално за десктоп приложения).
   * Отрицателни страни:
     + Липса на асинхронни операции, което може да забави UI при големи заявки.
     + Хардкоднато конфигуриране на пътя на базата данни, което ограничава гъвкавостта.
2. **WPF с MVVM архитектура**
   * Положителни страни:
     + Четливо разделение на логика (ViewModel), UI (View) и данни (Model).
     + Реактивен UI чрез INotifyPropertyChanged и автоматичен data binding.
     + Преизползваемост на компоненти (напр. AgeToBrushConverter).
   * Отрицателни страни:
     + Сложност при управление на зависимости между ViewModels.
     + Ограничена поддръжка за валидация на входни данни (напр. възраст, пароли).
3. **SQLite (база данни)**
   * Положителни страни:
     + Лек и портативен, подходящ за десктоп приложения.
     + Нулева конфигурация за разработчици.
   * Отрицателни страни:
     + Ограничена поддръжка за сложни SQL заявки и транзакции.
4. **Потребителски интерфейс (UI)**
   * Положителни страни:
     + Интуитивна навигация според ролите (админ, професор, студент).
     + Визуална обратна връзка (оцветяване на редове по възраст, статусни съобщения).
   * Отрицателни страни:
     + Липса на защита срещу неоторизиран достъп (напр. студенти да виждат чувствителни данни).
     + Няма потвърждение при критични действия (изтриване на всички потребители).

**3.Проектиране**

**Кой ще използва продукта?**

* **Целеви потребители**:
  + Администратори
    - Управление на потребителски акаунти (добавяне, изтриване, редактиране).
    - Преглед на всички потребители и логове.
  + Професори и Инспектори
    - Преглед на списък със студенти.
    - Ограничен достъп до чувствителни данни.
  + Студенти (потенциални бъдещи потребители)
    - Само преглед на собствената информация.
* **Въздействие върху дизайна**:
  + **Ролево-базиран достъп (Role-Based Access Control - RBAC)**
    - Администратори виждат всички функционалности (AddUser, AllUsers).
    - Професори/инспектори виждат само Students.
  + **Интуитивен UI**
    - Скриване/показване на бутони според ролята (btnAddUser.Visibility = IsAdmin).

**2. Какви данни ще се използват**?

А. Входни данни от потребителя:

| **Данна** | **Описание** | **Валидация** |
| --- | --- | --- |
| Name | Потребителско име | Не трябва да е празно |
| Password | Парола | Не трябва да е празна |
| Age | Възраст | > 0, цяло число |
| Roles | Роля (Admin, Student, Professor) | Избор от предварително дефинирани стойности |
| Expires | Срок на валидност | Опционално, формат DateTime |

3. Как ще бъдат достъпени функционалностите?

1. **Вход в системата (Login)**:
   * Проверка на Name/Password срещу базата данни.
   * Зареждане на UI според ролята.

Потребителски интерфейс (UI):

Навигационен панел

Бутоните се показват/скриват според ролята:

xml

<Button x:Name="btnAddUser" Content="Add User" Visibility="{Binding IsAdmin}"/>

Страници:

|  |  |  |
| --- | --- | --- |
| Страница | Описание | Достъпна за роли |
| LoginControl | Вход в системата | Всички |
| AddUserControl | Добавяне на нов потребител | Администратори |
| AllUsersList | Списък с всички потребители | Администратори |
| StudentsList | Списък със студенти | Професори/Инспектори |

**Workflow на потребителя**:

A screenshot of a computer

AI-generated content may be incorrect.

**4. Програмна организация**

**Архитектура**: MVVM (Model-View-ViewModel)

* **Модел (Model)**:
  + DatabaseUser, LogEntry (дефинират структурата на данните).
* **View (UI)**:
  + XAML файлове (AddUserControl.xaml, StudentsList.xaml).
* **ViewModel**:
  + AddUserViewModel, LoginViewModel (логика, валидация, комуникация с базата).

**Технологии**:

* **Entity Framework Core**:
  + За ORM (свързване на обекти с база данни).

csharp

Copy

Download

optionsBuilder.UseSqlite($"Data Source={databasePath}");

* **SQLite**:
  + Локална база данни за съхранение.
* **Data Binding**:
  + Свързване на UI елементи с ViewModel свойства:

xml

Copy

Download

Run

<TextBox Text="{Binding NewUserName, UpdateSourceTrigger=PropertyChanged}"/>

**Диаграма на компонентите**:

**5. Цели на предложените подобрения**

**Идея зад използваните техники**:

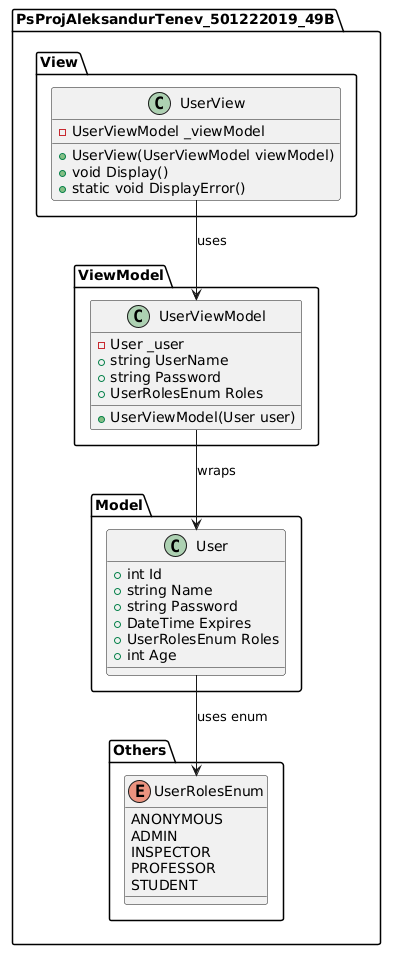
* **MVVM Архитектура**:
  + **Цел**: Отделяне на логиката от UI за по-лесно поддържане и тестване.
  + **Реализация**: ViewModel-ите управляват данните и команди, докато View се фокусира върху визуализацията.
* **Entity Framework Core**:
  + **Цел**: Опростяване на достъпа до база данни без ръчно пишене на SQL.
  + **Реализация**: Автоматично генериране на Id (ValueGeneratedOnAdd) и миграции.
* **WPF Data Binding**:
  + **Цел**: Реално-времово обновяване на UI при промяна на данните.
  + **Реализация**: INotifyPropertyChanged в ViewModel-ите.

**Очаквани ползи**:

* **Гъвкавост**: Лесно добавяне на нови роли или функционалности.
* **Сигурност**: Ролево-базиран достъп ограничава незаконен достъп.
* **Производителност**: Автоматичното генериране на заявки с EF Core оптимизира работата с базата.

**Крайна цел**:  
Създаване на **мащабируема**, **лесно поддържана** и **потребителско-ориентирана** система за управление на потребители в учебна среда, съобразена с нуждите на различните роли.

4. **Реализация на Системата**

**1. Реализация на Базата Данни (ER диаграма → SQLite)**

**Миграция към SQLite чрез EF Core**:

A computer screen shot of a program code

AI-generated content may be incorrect.

**2. Класове и Обекти (Модели)**

A computer screen shot of a code

AI-generated content may be incorrect.**Дефиниция на моделите**:

A computer screen with text and images

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

**3.Бизнес Процеси (Програмен Код)**

Бизнес процеси: Обяснение и Източници на Код

Бизнес процесите в приложението включват управление на потребителския животен цикъл (регистрация, вход, актуализация) и ролево-базиран достъп до данни. Всеки процес се реализира чрез комбинация от:

1. Валидация на входни данни (проверка за празни полета, валидни роли, възрастови ограничения).
2. Криптиране на чувствителни данни (хеширане на пароли с BCrypt).
3. Взаимодействие с базата данни (запис, четене, актуализация чрез EF Core).
4. Прилагане на бизнес правила (напр. само администратори могат да изтриват потребители).

**Ключови файлове за код:** A screen shot of a computer code

AI-generated content may be incorrect.

**Регистрация на потребител:**

* + **AddUserViewModel.cs (логика за валидация и запис)**
  + **UserData.cs (метод AddUser)**

A screen shot of a computer program

AI-generated content may be incorrect.

**Вход в системата:** A black screen with text

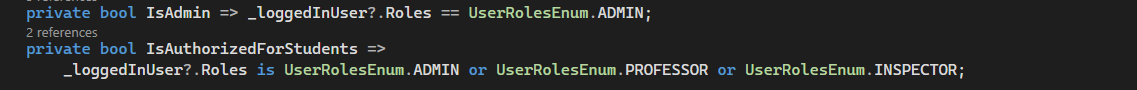
AI-generated content may be incorrect.

* **LoginViewModel.cs (проверка на идентичност)**

A screenshot of a computer

AI-generated content may be incorrect.

**Ролево-базиран достъп**:



**Изтриване на потребители**:

A computer screen with text

AI-generated content may be incorrect.

Диаграма на процесите:

A screenshot of a diagram

AI-generated content may be incorrect.

**4.Потребителски интерфейс**

**Използвани контроли в потребителския интерфейс**

1. **LoginControl - Форма за вход:**
   * TextBox за потребителско име
   * PasswordBox за парола
   * Button за вход
2. **AddUserControl - Форма за добавяне на нов потребител:**
   * TextBox за име
   * PasswordBox за парола
   * ComboBox за избор на роля
   * TextBox за възраст
   * DatePicker за избор на дата на изтичане
   * Button за добавяне на потребител
   * TextBlock за статусни съобщения
3. **AllUsersList - Списък с всички потребители:**
   * DataGrid за показване на потребителите
   * Button за изтриване на всички потребители
4. **StudentsList - Списък със студенти:**
   * DataGrid за показване на студентите
5. **MainWindow - Главен прозорец:**
   * Навигационни бутони (Add User, All Users, Students)
   * Button за изход от системата
   * ContentControl за динамично показване на различните контроли

**MainWindow**

A screenshot of a computer

AI-generated content may be incorrect.

AddUserControl

A screenshot of a computer

AI-generated content may be incorrect.

AllUsersList

A screenshot of a computer

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

Блок диаграма на потребителския интерфейс

A screenshot of a computer

AI-generated content may be incorrect.

**5.Потребителско ръководство (Резултати)**

Потребителят взаимодейства с продукта чрез интуитивен графичен потребителски интерфейс, съобразен с неговата роля. Основните стъпки и резултати от работа със системата включват: Вход в системата – всеки потребител въвежда потребителско име и парола, като интерфейсът се адаптира въз основа на тяхната роля – администраторите виждат менюта за добавяне, изтриване и преглед на всички потребители; професорите/инспекторите имат достъп само до списък със студенти; а студентите (ако бъде активирано) виждат лична информация. Добавяне на потребител става от администраторите чрез формата

AddUserControl, като при успешно добавяне се появява статусно съобщение. Преглед на списъци се осъществява чрез

DataGrid компоненти, които показват списъците със студенти и потребители, като потребителите могат да бъдат сортирани, а информацията се визуализира с акценти според възраст или статус. Изтриване на потребители е възможно само за администраторите чрез бутон за изтриване на всички потребители, но потвърждение към момента не се изисква, което е описано като потенциален риск. Изход от системата се осъществява чрез бутон, който изчиства текущата сесия и връща потребителя на екрана за вход. Системата реагира моментално при валидни операции и дава обратна връзка при грешки, например при невалидна парола или липсващи полета.

**6.Заключение**

Разработеният продукт отговаря на основните цели, поставени в началото на проекта – създаване на мащабируемо и лесно за поддръжка приложение с реална приложимост в учебна среда. Приложението покрива основните функционалности, необходими за управление на потребители и роли: успешно реализира ролево-базиран достъп; използва MVVM архитектура, което улеснява бъдеща разработка; UI интерфейсът е удобен и визуално разбираем. Системата показва стабилна работа и отговаря на нуждите на целевите потребители. Въпреки някои забележки (напр. липса на потвърждение при изтриване), приложението е готово за употреба и може да бъде надградено с минимални усилия в бъдеще.

**7.Литература**

**Microsoft Docs.** *WPF documentation.* ([https://docs.microsoft.com/en-us/dotnet/framework/wpf/](https://www.google.com/search?q=https://docs.microsoft.com/en-us/dotnet/framework/wpf/)).

**Microsoft Docs.** *Entity Framework Core documentation.* (<https://docs.microsoft.com/en-us/ef/core/>).

**Microsoft Docs.** *Model-View-ViewModel (MVVM) pattern documentation.* (<https://docs.microsoft.com/en-us/dotnet/architecture/maui/mvvm>).

**Microsoft Docs.** *C# documentation.* (<https://docs.microsoft.com/en-us/dotnet/csharp/>).

**Microsoft Docs.** *SQLite with .NET.* (<https://docs.microsoft.com/en-us/dotnet/standard/data/sqlite/>).

*Pro C# 10 with .NET 6: Foundational Principles and Practices in Programming.* Apress