**Картина, която съдържа черен, тъмнина

Описанието е генерирано автоматично Технически университет – София**

**Факултет Компютърни системи и технологии**

**Курсов проект**

**Дисциплина: Програмни среди**

**Изготвил: Пламен Славчев**

**Факултетен номер: 501221066**

**Административна група: 49**

**Документация за проект на MAUI**

Като необходимост преди да започнем със самия MAUI проект е необходимо интеграционна среда в случая използваме **Visual Studio 2022**, инсталиран **.NET Framework** и **MAUI** шаблони за проекти, които се инсталират във Visual Studio.

Проектът е направен с архитектура изполваща **MVVM** – Model, View, ViewModel модела. Съответно основните класове са:

* **ViewModel – ите**, те управляват данните и бизнес логиката: DatabaseUserViewModel и LogEntityViewModel.
  + DatabaseUserViewModel-ът отговаря за списъка с потребители.  
    A computer screen shot of a program

    Description automatically generated
  + LogEntityViewModel-ът отговаря за списъка с логовете.  
    A computer screen shot of text

    Description automatically generated
* View – тата (XAML файлове), те отговарят за потребителския интерфейс (UI):
  + DisplayLogs – показва списък с логовете от базата данни  
    A screen shot of a computer program

    Description automatically generated
  + LogPage – представя съдържанието на DisplayLogs в страницая  
    A computer screen with white text

    Description automatically generated
  + DisplayUsers - показва списък с потребителите от базата данни  
    A screen shot of a computer program

    Description automatically generated
  + UserPage – представя съдържанието на DisplayUsers в страница  
    A screen shot of a computer program

    Description automatically generated
* Конвертор, използва се за преобразуване на данните при визуализация – PasswordMaskConverter – “маскира“ паролите на потребителите при визуализация в потребителският интерфейс.  
  A screen shot of a computer code

  Description automatically generated

ViewModel-ът **DatabaseUserViewModel** имплементира интерфейса **InotifyPropertyChanged**, който служи за уведомяване на UI-a за промени в данните. В конструкторът се извиква findAllUsers(), метод който извлича всички потребители записани в базата данни и ги записва като ObservableCollection от User entity, тази колекция се обновява автоматично при промяна на данните. Методът OnPropertyChanged уведомява UI-а за промени в свойството.

ViewModel-ът LogEntityViewModel също имплементира **InotifyPropertyChanged,** отново има ObservableCollection от LogEntity и използва findAllLogs() метода. Съдържа полето \_selectionEntity, което се използва за да се следи, кой лог е маркиран в момента и да се извика метода ListLogDetails. Този метод тригърва DisplayAlert с данните на лога.

View-то **DisplayLogs** съдържа ListView, който показва списък с логовете, но само времето в което се е записал даденият лог. Има и атрибют SelectedItem, който се свързва с SelectionEntity.

View-то **DisplayUsers** отново съдържа **ListView**, който показва списъка с потребители и тяхната информация. Също така използва конвъртъра **PasswordMaskConverter** при визуализиране на паролите не потребителите.

Конвъртърът PasswordMaskConverter имплементира интерфейса IValueConverter и има метод Convert, който да маскира символите на паролата с звездички.

Трудности срещнати по време на програмирането на проекта:

* Липсата на DataGrid в MAUI и заманя с ListView
* Празна визуализация на потребителите, сбъркано име на единт от байндингите (синтактична грешка)
* Проблем при референцирането на DataLayer и Welcome проектите – сменяне на output типът на двата проекта да е Library вместо Exe.