# Документация

Съдържание

[Документация 1](#_Toc61797817)

[1.Използвани технологии за реализация 2](#_Toc61797818)

[2.Проследяване по компоненти 2](#_Toc61797819)

[2.1. Компоненти 2](#_Toc61797820)

[2.2.Model 3](#_Toc61797821)

[2.3.Вписване в системата 7](#_Toc61797822)

[2.4. Създаване на нов родителски профил 9](#_Toc61797823)

[2.5. Родителски профил 12](#_Toc61797824)

[2.6.Учителски профил 16](#_Toc61797825)

[2.7. Редактиране на родителски/учителски профил. 21](#_Toc61797826)

[2.8.Администраторски профил 28](#_Toc61797827)

[2.9. Допълнителни елементи и становища. 33](#_Toc61797828)

Приложението „EduWatch“ е изградено на базата на Model-View-Presenter/MVP/ архитектура. Тя е подходяща , защото основните и особености позволяват ясно разграничение на логиката на отделните компоненти.

Модулите на приложението са три : View , Model ,Presenter.

### 1.Използвани технологии за реализация

За изпълнението на проекта „EduWatch“ е използвана средата за разработка VisualStudio и езика C#. Приложението е WindowsForm Desktop Application. Употребен е Entity Framework и релационна база данни.

## 2.Проследяване по компоненти

### 2.1. Компоненти

В трите слоя на приложението се намират съответните му елементи – класове и интерфейси.

За View това са :

* IView – общ интерфейс с декларирани основни методи на потребителския интерфейс /void Hide(),void Show(),void Close() иMessageResult Message()/.Той се наследява от интерфейсите на всяка форма.
* ILoginView – интерфейс на формата за впиване
* LoginForm -форма за вписване
* ISingUpView – интерфейс на формата за създаване на нов потребителски профил
* SingUpForm - формата за създаване на нов потребителски профил
* ITeacherView – интерфейс на учителския профил
* TeacherForm – учителски профил
* IParentView – интерфейс на родителски профил
* ParentForm – родителски профил
* IEditView – интерфейс на настойка на форма за настройка на родителски и учителски профил
* EditForm - форма за настройка на родителски и учителски профил
* IAdminView – интерфейс за администраторски профил
* AdminForm – администраторски профил

За Presenter това са :

* ILoginPresenter – интерфейс на LoginPresenter
* LoginPresenter – клас съдържащ логиката на формата за вписване
* ISingUpPresenter – интерфейс на SingUpPresenter
* SingUpPresenter – клас съдържащ логиката на формата за създаване на нов родителски профил
* ITeacherPresenter – интерфейс на TeacherPresenter
* TeacherPresenter – клас съдържащ логиката на формата на учителски профил
* IParentPresenter – интерфейс на ParentPresenter
* ParentPresenter – клас съдържащ логиката на формата на родителски профил
* IEditPresenter – интерфейс на EditPresenter
* EditPresenter – клас съдържащ логиката на формата за редактиране на родителски и учителски профил
* IAdminPresenter – интерфейс на AdminPresenter
* AdminPresenter – клас съдържащ логиката на администраторски профил
* PresenterFactory – клас , съдържащ методи за извикване на различните Presenter-и

### 2.2.Model

* Слой за съхранение на данни
* Отговаря за комуникацията с базата данни
* Използва Entity Framework за достъп до релационна база данни
* Осведомява Presenter при извличане , актуализиране и изтриване на данни.
* Presenter извлича от Model данните за визуализиране в потребителския изглед.
* Model и View нямат пряка връзка помежду си /взаимно не знаят за съществуването си/.

За внедряване на базата данни в приложението е използвана технологията Entity Framework , която дава възможност за работа с DB , използвайки обекти. Целта е постигане на по-високо ниво на абстракция. В случая EF е вписана в Model /между бизнес логиката в Presenter и базата данни/.

Entity Framework съдържа визуализация на таблиците от базата данни и връзките помежду им.

EF създава Entity Data Model въз основа на обекти със свойства get и set от различни типове данни.

В случая това са: Teacher , Parent , Admin , Grade , Note , Student , Subject.Използва този модел при заявка или записване на данни за обект в основната база данни. Позволява използването на LINQ заявки за извличане на информация от базата. Проследява промените настъпили в обектите /промяна на стойностите във свойствата/ , които трябва да бъдат изпратени в DB.

В приложението са използвани следните таблици:

Parent:

• parent\_id - индекс на родителя /първичен ключ/

• parent\_firstN – име на родителя

• parent\_lastN – фамилия на родителя

• username – потребителско име

• password\_hash – хеширана парола

Student:

• student\_id – индекс на ученика /първичен ключ/

• student\_firstN – име на ученика

• student\_lastN – фамилия на ученика

• student\_PIN – ЕГН на ученика

• parent\_id – индекс на родител /външен ключ/

* Code – секретен номер на ученика

• grade – оценка

Таблицата Student и таблицата Parent имат връзка помежду си, чрез parent\_id , което се явява външен ключ за ученическата таблица и основен ключ за родителската.

Teacher:

• teacher\_id – индекс на учителя /първичен ключ/

• teacher\_firstN – име на учителя

• teacher\_lastN – фамилия на учителя

• username – потребителско име

• password\_hash – хеширана парола

Subject:

• subject\_id – индекс на предмета /първичен ключ/

• subject\_name – име на предмета

• teacher\_is – индекс на преподавателя /вторичен ключ/

Таблицата Subject и таблицата Tacher имат връзка помежду си , чрез teacher\_id , което се явява външен ключ за таблица на предметите и основен ключ за учителската.

Note:

• note\_id – индекс на забележката /първичен ключ/

• note - забележка

• note\_date – ден на забележката

• note\_seen – видяна ли е забележката

• student\_id – индекс на ученика/външен ключ/

• subject\_id – индекс на предмета/външен ключ/

Таблицата Note и таблиците Subject и Student имат връзка помежду си , чрез ключове student\_id и subject\_id , които се завяват първични за Student и Subject.

Grade:

• grade\_id – индекс на оценката/първичен ключ/

• grade - оценка

• grade\_seen – видяна ли е оценката

• comment – коментар за оценката

• date -дата на въвеждане

• student\_id – индекс на ученик

• subject\_id – индекс на предмет

Таблицата Grade и таблиците Subject и Student имат връзка помежду си , чрез ключове student\_id и subject\_id , които се завяват първични за Student и Subject.

Admin:

• admin\_id – индекс на админ /първичен ключ/

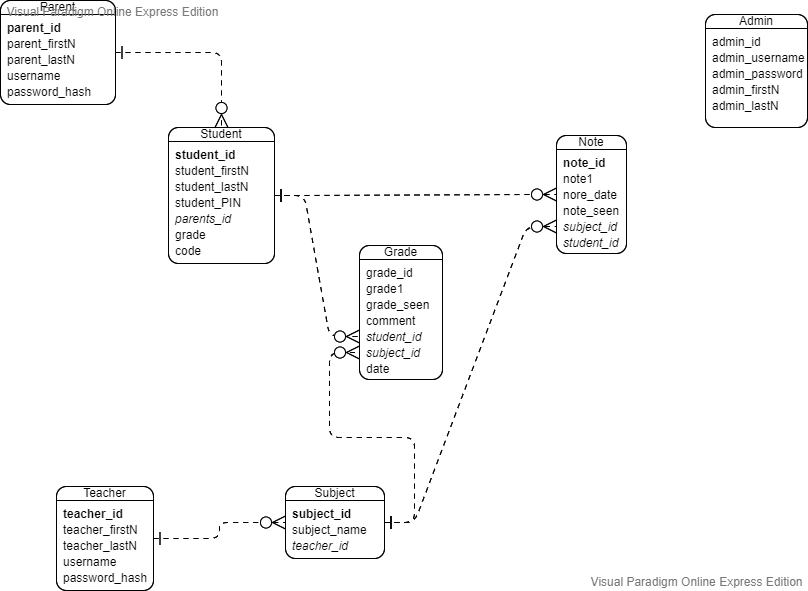
• admin\_username – потребителско име

• admin\_password - парола

• admin\_firstN - име

• admin\_lastN – фамилия

Диаграма показваща таблиците в базата данни/техните атрибути/ и връзки помежду им:



### 2.3.Вписване в системата

##### 1.ILoginView и LoginForm

ILoginView имплементира IView. В специфичния за формата за вписване в системата интерфейс са декларирани:

* Username - Пропърти за потребителско име.
* Pass - Пропърти за парола.
* Presenters.ILoginPresenter Presenter - Пропърти сочещо към ILoginPresenter. При различните събития чрез него ILoginView се обръща към методите , които се изпълняват в LoginPresenter.
* CreateParentView() – Метод за извикване на формата на родителски профил.
* CreateTeacherView() – Метод за извикване на формата на учителски профил.
* CreateSignUpView() - Метод за извикване на формата за създаване на нов профил.
* CreateAdminView() – Метод за извикване на формата на администраторски профил.
* StartApplication() - Метод за стартиране на приложението.

ILoginView се наследява от LoginForm. Дизайна на потребителска форма е изграден , чрез две текстови кутии за въвеждане на потребителско име и парола. Чрез тях пропъртитата получават стойностите си :

* Username – получава стойността си от userNameTextBox.Text
* Pass – получава стойността си passwordTextBox.Text

При натискане на бутоните „Вход“ и „Изход“ и линка „Създаване на нов профил“ се генерират събития , които при настъпването им се изпълняват от следните методи :

* LoginButton\_Click
* btnExit\_Click
* linkLabelCreateAccount\_LinkClicked

##### 2.ILoginPresenter и LoginPresenter .

Това e класа от Presenter , който съответства на формата , заедно в неговия интерфейс. В тях са декларирани и имплементирани следните методи, които посочват поведението на приложението след потребителско взаимодействие във LoginForm:

* void OnLoginButtonClick()
* void OnExitButtonClick()
* void ShowLoginView()
* void Start()
* void OnCreateAccountClick()

Класа LoginPresenter се асоциира с View и Model, чрез полетата view и data.

##### 3.Процеси при потребителско взаимодействие

При натискане на бутона за вход се изпълнява метода LoginButton\_Click ,чиято логика се изпълнява Presenter. Какво трябва да се случи при това взаимодействие е имплементирано в метода OnLoginButtonClick() от LoginPresenter:

* Ползвателят попълва потребителското си име и парола в текстовите кутии и избира като какъв иска да се впише в системата. След , което натиска бутона за вход.
* Presenter е асоцииран с View и Model чрез полетата view и data.
* Активира се метода OnLoginButtonClick() , който първо проверява дали са въведени правилно входните данни във View с проверки от вида If.
* Първата проверка има задача да върне към LoginForm съобщение , ако на свойството Username не е добавена стойност.
* При установяване на празен string системата връща потребителя за ново въвеждане на потребителско име и не му позволява да продължи.
* Втората проверка има задача да върне към LoginForm съобщение , ако на свойството Pass не е добавена стойност.
* При установяване на празен string системата връща потребителя за ново въвеждане на парола и не му позволява да продължи.
* Методът съдържа проверка , дали съществува въведения потребител. Тя посочва , че ако не съществува такъв профил в базата данни, LoginPresenter заявява на View да покаже съобщение за грешно потребителско име или парола и да не го допусне до системата.
* В метода се създава инстанция на класа за хеширане и проверка на пароли. Използва се за проверка дали въведената потребителска парола е правилна.
* Ако паролата не съответства на въведеното потребителско име LoginPresenter заявява на LoginForm да покаже съобщение за грешка при въвеждане.
* Ако входните данни от потребителя са коректно въведени и съответстват на данните за потребителско име , парола и вид потребител от Model , LoginPresenter заявява на View да покаже формите на съответните профили . Извлича данните на профилите от Model и ги изпраща към View за визуализация.

В LoginForm се намира и линк към прозорец за създаване на нов родителски профил в системата:

* При взаимодействие с потребителя LoginForm осведомява LoginPresenter.
* LoginPresenter заявява на View да скрие текущата форма.
* Нарежда извикване и зареждане на интерфейса и класа на новата форма /SingUpForm и ISingUpView/.
* Логиката на тази форма се съдържат в SingUpPresenter и неговия интерфейс затова метода се обръща към тях. Те се извикват и вече в тяхна задача да изпълняват потребителските заявки от SingUpForm.

При натискане на бутона за изход от системата :

* LoginForm отчита действието и осведомява LoginPresenter за него.
* LoginPresenter активира метода OnExitButtonClick() , който заявява на LoginForm да затвори приложението и на Model да прекъсне връзката с контекста на обекта.

### 2.4. Създаване на нов родителски профил

##### 1.ISingUpView и SingUpForm.

ISingUpView имплементира IView. В специфичния за формата за вписване в системата интерфейс са декларирани свойства:

* UserName - пропърти за потребителско име.
* Pass - Пропърти за парола.
* FirstName - Пропърти за име.
* Surname - Пропърти за фамилия.
* PasswordChar – Свойство за маскиране на парола.
* Свойство сочещо към ISingUpPresenter и SingUpPresenter – при различните събития в SingUpForm , чрез това пропърти се извикват съответните функции , които се изпълняват в Presenter.

В SingUpForm са имплементирани всички свойства и методи от интерфейсите. Дизайна на формата е от четири текстови кутии , в които се попълват – име , фамилия , потребителско име и парола. Чрез тях пропъртитата получават стойностите си :

* Username – Получава стойността си от textBoxUsername.Text.
* Pass - Получава стойността си от textBoxPassword.Text.
* Surname - Получава стойността си от textBoxSurname.Text.
* FirstName - Получава стойността си от textBoxName.Text.
* PasswordChar - Получава символния низ за маскиране от textBoxPassword.

За бутоните „Създай профил“ и „Отказ“ настъпват събития при натискането им , които се изпълняват от методите:

* btnCreate\_Click
* btnCancel\_Click

CheckBox „виж парола“ се използва за премахване на маската на зададените символи , които са попълнени като парола.То предизвиква събитието checkBoxViewPass\_CheckedChanged.

##### 2.ISingUpPresenter и SingUpPresenter

Клас от Presenter , който съответства на формата за създаване на нов родителски профил заедно с неговия интерфейс. В тях са декларирани и имплементирани следните методи, които посочват поведението на приложението след потребителско взаимодействие във SingUpForm:

* Start()
* OnCancelButtonClick()
* OnCreateButtonClick()
* OnPasswordVisibleCheckBoxChange()

В SingUpForm се изпълняват събитията при потребителското взаимодействие , но какво точно ще бъде показано на ползвателя се определя от SingUpPresenter. Класа SingUpPresenter се асоциира с View и Model, чрез полетата view и data , и с ILoginPresenter чрез полето loginPresenter.

##### 3.Процеси при потребителско взаимодействие

При натискане на бутона за вход се изпраща сигнал до SingUpPresenter. Какво трябва да се случи при това взаимодействие е имплементирано в метода OnCreateButtonClick() от SingUpPresenter:

* Потребителят попълва в текстовите кутии данните си и натиска бутона за създаване на нов профил.
* SingUpForm отчита действието и осведомява SingUpPresenter за него.
* Presenter активира метода OnCreateButtonClick().
* Неговата задача е да създаде нов родител в системата и да го добави в базата данни , като провери дали данните са въведени коректно.
* Това се случва , чрез серия от if проверки.
* Първата проверка има задача да установи дали е въведено потребителско име в текстовото поле на SingUpForm.
* При установяване на празен string SingUpPresenter нарежда на SingUpForm да визуализира съобщение за пропуска на потребителя и не позволява създаването на нов родителски профил.
* Втората проверка има задача да установи дали е въведена парола.
* При установяване на празен string SingUpPresenter нарежда на SingUpForm да визуализира съобщение за пропуска на потребителя и не позволява създаването на нов родителски профил.
* Третата проверка има задача да установи дали са въведени име и фамилия на потребителя.
* При установяване на празни string-ове SingUpPresenter нарежда на SingUpForm да визуализира съобщение за пропуска на потребителя и не позволява създаването на нов родителски профил.
* В приложението е задължително името и фамилията на родителя да бъдат изписани на кирилица. Ако потребителя въведе име и фамилия с латинската азбука , SingUpPresenter заявява на SingUpForm да визуализира съобщение с допусната грешка и не позволява създаването на нов родителски профил.
* Друга важна особеност е въвеждането на потребителското име с латински символи. Създадена е проверка , която прихваща неизпълнението на тази инструкция.

SingUpPresenter заявява на SingUpForm да върне съобщение за нарушеното задание и не позволява създаването на нов родителски профил.

* Създадени са и проверки , които сравняват в базата данни дали вече съществува потребител с дадено потребителско име.
* Ако съществува такъв потребител с въведеното потребителско име SingUpPresenter заявява на SingUpForm да визуализира съобщение , за въвеждане на ново потребителско име поради наличие на заявеното в базата данни.
* Потребителят трябва да въведе парола с дължина от 8 до 25 символа. Ако условието не е изпълнено SingUpPresenter заявява на SingUpForm да покаже съобщение за невалидното въвеждане и не позволява създаването на профила.
* При успешно въвеждане на всички данни профила се добавя в базата данни като част от родителската таблица. Новият потребител се запазва в базата , заедно с въведените потребителско име , хеширана парола , име и фамилия.
* При отказ на базата данни да запази новия профил SingUpPresenter заявява на SingUpForm да върне на потребителя съобщение за моментната грешка.
* При успешно създаване на родителския профил SingUpPresenter заявява на SingUpForm да визуализира съобщение за успешно създаване на профила ,след което да затвори формата за SingUp.
* Накрая SingUpPresenter извиква Presenter-a на формата за вписване и активира метода ShowLoginView() , който нарежда на View да стартира формата за вписване.

При въвеждане на парола първоначално е маскирана. Създаден е checkbox,при натискането му се изпълнява метода checkBoxViewPass\_CheckedChanged ,чиято цел е показване на потребителя , каква е текущата парола която е посочил :

* При натискане SingUpForm изпраща съобщение до SingUpPresenter , че е поискана видимост на паролата.
* Активира се метода OnPasswordVisibleCheckBoxChange() , който се изпълнява в SingUpPresenter.
* Той премахва маската от паролата.
* SingUpPresenter заявява на SingUpForm да покаже въведената от потребителя парола.

SingUpForm съдържа бутон за затваряне на формата:

* При наличието на профил потребителя може да се върне в първоначалната LoginForm.
* View отчита потребителско взаимодействие /затваряне на SingUpForm / към съответстващия му SingUpPresenter.
* Изпълнява се метода на SingUpPresenter - OnCancelButtonClick(), който заявява на View да затвори SingUpForm и да визуализира отново формата за вписване заедно с нейната логика.

### 2.5. Родителски профил

Родителят достъпва системата през LogInForm, където въвежда своите данни под формата на потребителско име и парола. Ако предоставените данни са валидни, родителят получава достъп до своя профил и може да използва приложението.

##### 1.IParentView и ParentForm

Интерфейсът IView се имплементира от интерфейса на родителския профил IParentView. Методите, декларирани в IParentView, дефинират какъв тип поведение ще демонстрира обектът Parent при взаимодействие с дизайна на родителския профил:

Интерфейсът IView се имплементира от интерфейса на родителския профил IParentView. Методите, декларирани в IParentView, дефинират какъв тип поведение ще демонстрира обектът Parent при взаимодействие с дизайна на родителския профил:

* Presenters.IParentPresenter Presenter { set; } – пропърти, чрез което се осъществява връзката на IParentView с ParentPresenter. Чрез него IParentView се обръща към методите , които се изпълняват в ParentPresenter.
* ParentNameLabelText { get; set; } – пропърти, чрез което рarentNameLabel получава стойност във вид на име и фамилия на родителя;
* ComboBoxStudentSelectedIndex { get; set; } – пропърти, използвано за промяна избора на ученик. Използва се при if – проверки и в методи от ParentPresenter, за да се провери дали индексът не е -1, т.е дали падащото меню (ComboBox-а) не е върнато в начално състояние, когато не е избран никой ученик ;
* ComboboxSubjectSelectedIndex { get; set; } – пропърти, използвано в случаите, когато има промяна на избора на предмет от падащото меню. Използва се при if – проверки и в методи от ParentPresenter, за да се провери дали индексът не е -1, т.е дали падащото меню (ComboBox-а) не е върнато в начално състояние, когато не е избран никой предмет;
* SelectedStudentID { get; } –пропърти, чиято цел е да вземе стойността на id на ученика от ComboBox-a;
* SelectedSubjectID { get; } – пропърти, което се използва от ParentPresenter за извличане на ID на предмет, по който се обучава ученикът от базата данни в Model;
* ComboBoxSubjectEnabled { get; set; } – пропърти, което приема две стойности – true или false в зависимост от събитията, предизвикани от родителя. True прави ComboBox-а с предметите достъпен, а false го прави забранен за ползване;
* object GridViewData { get; set; } – пропърти за връзка с ParentPresenter. Взема информация от БД и я праща към IParentView за визуализация;
* AutoGenerateDataColumns { get; set; } – пропърти за автоматично генериране на колоните в таблицата DataGridView;
* SelectedGradesView { get; set; } – пропърти, приемащо две стойности – true или false в зависимост дали е избран радио бутонът за визуализация на оценките;
* AverageGradeButtonEnabled { get; set; } - пропърти, приемащо две стойности – true или false в зависимост дали бутонът за среден успех е отворен за ползване, или не;
* AverageGradeText { get; set; } – пропърти за визуализация на средния успех в TextBox – а, предназначен за това;
* GradesRadioButtonEnabledState { get; set; } – пропърти за състоянието на радио бутона за оценки. Приема две стойности – true или false, в зависимост дали може да се борави с RadioButton-а, или не;
* NotesRadioButtonEnabledState { get; set; } - пропърти за състоянието на радио бутона за забележки. Приема две стойности – true или false, , в зависимост дали може да се борави с RadioButton-а, или не;
* void FillInCorrespondingStudents(List<Tuple<int, string>> students) – метод с връзка към ParentPresenter, който приема списък от двойки (Tuple) от ученика и неговото ID, чрез което той/тя е свързан с конкретния родител в базата данни. Въз основа на тази информация, данните се подготвят за визуализация в ComboBox-a за избор на ученик.
* void FIllInCorrespondingSubjects(List<Tuple<int, string>> subjects) – метод с връзка към ParentPresenter, който приема списък от двойки (Tuple) от предмет и неговото ID, чрез което той/тя е свързан с конкретния ученик в базата данни. Въз основа на тази информация, данните се подготвят за визуализация в ComboBox-a за избор на предмет.
* ClearData() – метод, който изчиства съдържанието на данните в определените места за попълването им след приключване действията на родителя;
* void DisableUnseeingData() – чрез foreach блока, методът обхожда всички колони на DataGridView-то и ги прави ReadOnly, т.е. да не могат да бъдат променяни от потребителя. Следващият foreach блок обхожда всички клетки от колоната, съдържаща CheckBox за обратна връзка от родителя(„Видяно“), и ги прави отново readOnly, т.е. веднъж отбелязал/а, че се е запознал/а с информацията, родителят не може да се върне и да махне тикчето от състоянието „Видяно“.
* FormatDataDisplay() – метод, който визуализира данните от БД след като с помощта на ParentPresenter са били извлечени и предоставени на IParentView.

Дизайнът на родителската форма включва лейбъл за имената на родителя; два бутона съответно за настройки и изход ; две падащи менюта (ComboBox) за избор на ученик и предмет; два radioButton-a, предоставящи избор на родителя дали да се запознае с оценките на ученика или със забележките; DataGridView, където ще бъде показвана цялата нужна информация; бутон за среден успех и TextBox, където да бъде показан точният среден успех.

##### 2.IParentPresenter и ParentPresenter

Класът ParentPresenter осъществява връзка с View чрез полето view, с базата данни – чрез полето data.

В интерфейса IParentPresenter са декларирани методите свързани със събитията, предизвикани от родителя в родителския изглед. ParentPresenter имплементира тези методи.

* void Start() – метод, стартиращ родителската форма;
* void OnExitButtonClick() - метод, за изпълнение на събитие, предизвикано с натискане на бутона „Изход“. Запазва промените, направени от родителя, затваря родителския изглед и показва LogInForm-ата;
* void OnStudentSelection() – метод, за изпълнение на събитие, предизвикано с избора на ученик от падащото меню;
* void OnGradesButtonClick() – метод, за изпълнение на събитие, предизвикано при направен избор „оценки“ чрез кликане в radioButton-a. Запазва текущия избор, вика метода PrepareDataForView и прави if – проверка дали radioButton-ът „оценки“ е избран – ако да, прави бутонът „среден успех“ отворен за ползване. PrepareDataForView влиза в първата if – проверка и прави връзка с БД. Извлича всички оценки на избрания ученик по избрания предмет и ги прави видими в родителския изглед чрез DataGridView. ;
* void OnSubjectSelection() – метод, за изпълнение на събитие, предизвикано при направен избор на конкретен предмет от падащото меню (ComboBox). Прави връзка с БД от папка Model и чрез полето view визуализира съдържанието в падащото меню. Отваря за ползване radioButton-те за „оценки“ и „забележки“ и проверява дали потребителят е избрал „оценки“ – ако да, прави бутонът за среден успех отворен за ползване ;
* void OnAverageGradeButtonClick() – метод, за изпълнение на събитие, предизвикано при кликане на бутона „Среден успех“. ParentPresenter прави връзка с БД и извлича всички оценки на ученика, на които се намира средноаритметичното и се конвертира в string. Средният успех се визуализира чрез полето view;

##### 3.Процеси при потребителско взаимодействие

* Инициализацията на изгледа става в конструктора ParentPresenter. Чрез променливата parent от БД в Model се извличат данните (име и фамилия) на родителя. Чрез параметъра view тези данни се импортират в ParentNameLabelText и стават видими в родителския профил.
* Родителят избира ученик от падащото меню (comboBoxStudent), съдържащо имената на ученика/учениците, свързан/и с него. Чрез конструктора ParentPresenter се прави заявка към БД в Model и се вземат всички ученици, свързани с конкретния родител. ParentPresenter визуализира данните в падащото меню чрез параметъра view и метода FillInCorrespondingStudents .
* Родителят инициира събитие с избора си на ученик. Методът OnStudentSelection праща заявка към базата данни и по ID на ученика проследява предметите и забележките към тях, обединява ги в двойка (Tuple) и ги подготвя за визуализация в родителския профил. Методът FIllInCorrespondingSubjects в ParentForm осъществява визуализацията в comboBoxSubject.
* Родителят инициира ново събитие с избора си на предмет. ParentPresenter, чрез метода OnSubjectSelection и полето data, праща заявка към БД в Model и след като е посочен предметът, радио бутоните за избор на визуализация („оценки“ и “забележки“) стават отворени за взаимодействие.
* Ако родителт е решил да се запознае с оценките, прави избора си чрез клик в radioButtonGrades, радио бутонът с наименование „оценки“. Възниква събитие и методът OnGradesButtonClick го изпълнява. OnGradesButtonClick в ParentPresenter вика метода PrepareDataForView. Този метод влиза в първата if-проверка дали е избран изгледа за оценки и праща заявка към БД да даде достъп до всички оценки, които има ученика по избрания предмет. Методът подготвя данните за визуализация в DataGridView. Чрез параметъра view и метода FormatDataDisplay, ParentForm показва таблицата с: оценките на ученика, checkbox за състоянието „видяно“ и ред за уточнения към оценката.
* Ако родителят реши да уведоми, че се е запознал с оценките, може да постави тикче в checkbox-a. FormatDataDisplay прави всички редове и колони в DataGridView readOnly, освен тези за обратна връзка. Веднъж дал обратна връзка, потребителят не може да се върне и да премахне тикчето в checkBox-a – и състоянието „видяно“ вече е ReadOnly.
* При кликане в radioButtonGrades, радио бутонът с наименование „оценки“, възниква събитие. МЕтодът OnGradesButtonClick го изпълнява. if проверката в него прави бутонът за среден успех активен. При кликане на този бутон, възниква ново събитие и ParentPresenter реагира чрез метода OnAverageGradeButtonClick. View регистрира събитието в ParentForm чрез метода btnAverage\_Click. Полето view в метода OnAverageGradeButtonClick визуализира средноаритметичната стойност до втория от оценките в textBoxAverage – textbox-а за средния успех.
* Другата възможност за избор на родителя е да се запознае със забележките на ученика.При кликане на радио бутона „забележки“, ParentPresenter реагира чрез метода PrepareDataForView. Влиза във втората if-проверка и праща заявка до Model за извличане на всички забележки на ученика според избрания предмет. Методът подготвя данните за визуализация в DataGridView. View визуализира забележките в родителския профил чрез метода FormatDataDisplay.
* Процедурата със състоянието „видяно“ е валидна и при избор за визуализация на забележки.
* Направените промени от родителя в приложението се отразяват от ParentPresenter чрез метода SaveChanges. Методът обновява състоянието в БД в зависимост от това кои данни родителят е отбелязвал като видени. Ако има възникнал технически проблем в приложението, методът изпраща към View съобщение за грешка.
* Когато кликне върху бутона „Изход“, родителят предизвиква ново събитие. View реагира чрез метода btnExit\_Click – то прави връзка с метода OnExitButtonClick в Presenter.
* При смяна на потребителските данни, ParentNameLabelText се асоциира с нови име и фамилия чрез метода ResetViewAfterUserChangesTheirProfile. Методът променя View като comboBoxStudent получава нови стойности от БД, comboBoxSubject получава нови стойности за предмет, изчистват се предишните избори на „оценки“ и „забележки“. Presenter променя изгледа чрез полето view и метода ClearData, който изчиства данните в DataGridView и тези в TextBox-a за среден успех.

### 2.6.Учителски профил

Учителят извършва дейността си чрез своя потребителски профил – достъпът до него се осъществява след коректно въведени входни данни (потр.име и парола) за вход в системата.

##### 1.ITeacherView и TeacherForm

Интерфейсът IView се имплементира от този на учителския профил ITeacherView. Методите, декларирани в ITeacherView, дефинират какъв тип поведение демонстрира обектът Teacher при взаимодействие с дизайна на учителския профил. Същите тези методи са имплементирани и override-нати в TeacherForm:

* Presenters.TeacherPresenter Presenter { set; } – пропърти, чиято цел е да свърже ITeacherView с TeacherPresenter. Чрез него ITeacherView се обръща към методите , които се изпълняват в TeacherPresenter.
* string TeacherNameLabelText { get; set; } – пропърти, чрез което teachersNameLabel получава стойност под формата на име и фамилия на учителя, въз основа на потр.име, което е въвел в LogInForm;
* int ComboBoxSubjectSelectedIndex { get; set; } – пропърти за задаване и връщане на числова стойност (индекс) на предметите от падащото меню (comboBoxSubject), в зависимост от текущото състояние на изгледа;
* int ComboBoxStudentSelectedIndex { get; set; } - пропърти за задаване и връщане на числова стойност (индекс) на учениците от падащото меню (comboBoxStudent), в зависимост от текущото състояние на изгледа;
* int SelectedSubjectID { get;} – пропърти, което връща стойността на ID на избрания от падащото меню (comboBoxSubject) предмет;
* int SelectedStudentID { get;} – пропърти, което връща стойността на ID на избрания от падащото меню (comboBoxStudent) ученик;
* bool ComboBoxStudentEnabled { get; set; } – пропърти, което задава и връща стойност true или false в зависимост от това дали падащото меню за избор на ученик comboBoxStudent е отворено за взаимодействие;
* bool ComboBoxSubjectEnabled { get; set;} - пропърти, което задава и връща стойност true или false в зависимост от това дали падащото меню за избор на предмет comboBoxSubject е отворено за взаимодействие;
* bool gradeRadioBtnEnabled { get; set; } - пропърти, което задава и връща стойност true или false в зависимост от това дали падащото меню за избор на предмет comboBoxSubject е отворено за взаимодействие;
* bool noteRadioBtnEnabled { get; set; } - пропърти, което задава и връща стойност true или false в зависимост от това дали падащото меню за избор на предмет comboBoxSubject е отворено за взаимодействие;
* string ComboBoxGrade { get; } - пропърти, което връща стойности под формата на цифри от 2 до 6 от в gradeComboBox – падащото меню за избор на оценка при добавяне на такава от учителя;
* string TextBoxComment { get; set; } – пропърти, което задава и връща стойност на textBoxComment при добавяне на коментар от учителя;
* string ComboBoxGrade1to12 { get; } - пропърти, което връща стойности под формата на цифри от 8 до 12 от в comboBoxGrade1to12– падащото меню за избор на клас при добавяне на нов запис или преглеждане на старите записи от учителя;
* object GridViewData { get; set; } – пропърти, което задава и връща данните (чрез обекта DataSource от тип DataGridView) за оценки и забележки при подготовка за визуализацията им в DataGridView таблицата;
* bool SaveButtonEnabled { get; set; } – пропърти, което задава и връща две стойности true или false, в зависимост дали бутонът „Добави“е отворен за взаимодействие;
* bool AutoGenerateDataColumns { get; set; } – пропърти, което връща и задава стойност по колони в DataGridView таблицата;
* bool SelectedGradesView { get; set; } – пропърти, което задава и връща стойност true или false в зависимост от това дали radioButtonGrade е избран, или не;
* string AverageGradeTextBox { get; set; } – пропърти, което задава и връща стойност под формата на среден успех на ученика в аverageGradeTextBox;
* bool IsDataGridViewVisible { get; } – пропърти, което връща стойност true или false в зависимост от това дали DataGridView таблицата е видима или не;
* bool IsTextBoxCommentVisible { get; } - пропърти, което връща стойност true или false в зависимост от това дали commentTextBox е видимо или не;
* bool IsCommentLabelVisible { get; } - пропърти, което връща стойност true или false в зависимост от това дали сommentLabel е видим или не;
* bool IsGradeLabelVisible { get; } - пропърти, което връща стойност true или false в зависимост от това дали gradeLabel е видим или не;
* bool IsGradeComboBoxVisible { get; } - пропърти, което връща стойност true или false в зависимост от това дали падащото меню с оценките gradeComboBox е видимо или не;
* bool IsSaveBTNVisible { get; } - пропърти, което връща стойност true или false в зависимост от това дали бутонът за запазване на промените в диалоговия прозорец за настройките на учителския профил е видим или не;
* bool ISAverageGradeBtnVisible { get; } - пропърти, което връща стойност true или false в зависимост от това дали бутонът за среден успех е видим или не;
* bool ISAverageGradeTextBoxVisible { get; } - пропърти, което връща стойност true или false в зависимост от това дали бутонът за среден успех averageGradeButton е видим или не;
* void FIllInCorrespondingSubjects(List<Tuple<int, string>> subjects) – метод, който приема двойка от елементи съответно от тип int и string и задава стойностите в comboBoxSubject във на предмети;
* void FillInCorrespondingStudents(List<Tuple<int, string>> students) – метод, който приема двойка от елементи съответно от тип int и string и задава стойностите в comboBoxStudent във вид на имена на ученици;
* void FillInGrades(string[] grades) – метод, чрез който gradeComboBox получава стойност във вид на оценки от 2 до 6. Прибавя масива от оценки към списъка с оценки в падащото меню ;
* void FillInWhichGrade(string[] grade) - метод, чрез който comboBoxGrade1to12получава стойност във вид на класове от 8 до 12. Прибавя масива от номера на класове към списъка с класовете в падащото меню ;
* void FormatDataDisplay() – метод, който визуализира DataGridView таблицата с данните, избрани от учителя в учителския изглед;
* void ClearData() – метод, който изчиства предишните стойности от DataGridView таблицата с данните и подготвя таблицата за новите данни ;
* void HideAll() – метод, който връща в начално състояние учителски изглед;

Чрез изброените свойства Presenter (TeacherPresenter) прави връзка с View (TeacherForm).

##### 2.ITeacherPresenter и TeacherPresenter

Класът TeacherPresenter осъществява връзка с View чрез полето view, с базата данни – чрез полето data, с Login формата – чрез loginPresenter. Параметърът user асоциира Presenter с Model, на локалната променлива teacher се присвоява стойността на user.

В интерфейса ITeacherPresenter са декларирани методите, свързани със събитията, предизвикани от администратора в администраторския изглед. TeacherPresenter имплементира тези методи.

* void Start() – визуализира учителската форма;
* void OnExitButtonClick() – метод, за изпълнение на събитие, предизвикано с натискане на бутона „Изход“. Затваря учителския изглед и чрез полето loginPresenter визуализира LogInForm-ата.

##### 3.Процеси при потребителско взаимодействие

* Инициализацията на изгледа става в конструктора TeacherPresenter. Чрез променливата teacher от БД в Model се извличат данните (име и фамилия) на учителя. Чрез параметъра view тези данни се импортират в teachersNameLabel и стават видими в учителския профил.
* Учителят избира номер на клас от падащото меню comboBoxGrade1to12. След това избира предмет от падащото меню comboBoxSubject. View реагира на това събитие чрез метода comboBoxSubject\_SelectionChangeCommitted – проверява дали падащото меню за избор на клас не е празно – ако не е, прави връзка с Presenter и метода OnSubjectSelection, ComboBoxStudentEnabled става true, т.е. отворен за избор на ученик, gradeRadioBtnEnabled и noteRadioBtnEnabled са все още false.
* OnSubjectSelection прави if-проверка дали има избран предмет и, въз оснива на избрания предмет и номер на клас, с помощта на полето data, извлича имената на ученика от БД в Model. Чрез полето view се извиква метода FillInCorrespondingStudents. Падащото меню за избор на ученик ComboBoxStudent става отворено за взаимодействие;

Получаване на информация за оценки по даден предмет

* При натискане на radioButton с наименование „оценки“ възниква събитие, което radioButtonGrade\_CheckedChanged изпълнява. При натискане на бутона „Виж дневника“ във View възниква събитие SeeDatabtn\_Click.
* Методът radioButtonGrade\_CheckedChanged прави връзка с Presenter с метода OnGradesButtonClick – той от своя страна вика метода за подготвяне визуализация на данните PrepareDataForView. If-проверката в него прави връзка с БД, взема всичката информация за оценките на посочения ученик по посочения предмет и я подготвя за визуализация в dataGridView таблица. Полето view и методът FormatDataDisplay правят визуализацията на данните в учителския изглед.
* С натискане на бутона „Виж дневника“ възниква събитие, което SeeDatabtn\_Click изпълнява. Методът влиза в if-проверка дали един от двата радио бутона „оценки“ или „забележки“ не е избран. В случая, избран е „оценки“, DataGridView таблицата с данните се визуализира, averageGradeTextBox за просвояване на стойността на средноартиметичното от оценките и бутонът за среден успех btnAverage стават видими.
* Ако учителят не е конкретизирал „оценки“ или „забележки“ иска да разгледа, else блокът в SeeDatabtn\_Click изпраща съобщение за внимание.
* Учителят се запознава със средния успех с натискане на бутона „Среден успех“. Възниква събитие btnAverage\_Click във View. То прави връзка с Presenter чрез събитието OnAverageGradeButtonClick. Ако currentGrades.Count е повече от 0, се пресмята средния успех до втория знак след десетичната запетая чрез разширението Average и стойността се преобразува в string.

Нов запис на оценки по даден предмет

* При натискане на radioButton с наименование „оценки“ и при натискане на бутона „Добави нов запис“ във View възниква събитие, което AddRecordbtn\_Click изпълнява.
* AddRecordbtn\_Click влиза в if-проверката и проверява дали radioButtonGrade е избран. Ако да, подготвя изгледа за добавяне на оценка.
* При натискане на бутона „Добави“, възниква събитие във View, което Savebtn\_Click изпълнява. Savebtn\_Click метода прави връзка с Presenter чрез метода OnSaveGradeButtonClick.
* OnSaveGradeButtonClick от своя страна, влиза в if-проверката дали TextBox-а с коментар към оценката е попълнен и дали от ComboBoxGrade менюто е избрана оценка.
* Ако да, data полето заявява на БД в Model да запази информацията. Изгледът се рестартира чрез HideAll и OnSaveGradeButtonClick (през полето view) изпраща съобщение за успешен запис.
* Ако не, OnSaveGradeButtonClick през полето view изпраща съобщение за неуспешен запис.

Получаване на информация за забележки по даден предмет

* Учителят избира „забележки“. След това клика на бутона„Виж дневника“ – възниква събитие във View, изпълняващо се от метода SeeDatabtn\_Click. if-проверката проверява дали един от двата радио бутона „оценки“ или „забележки“ е избран. В случая, избран е „забележки“, DataGridView таблицата с данните се визуализира, averageGradeTextBox за просвояване на стойността на средноартиметичното от оценките и бутонът за среден успех btnAverage стават видими.
* Ако учителят не е конкретизирал „оценки“ или „забележки“ иска да разгледа, else блокът в SeeDatabtn\_Click изпраща съобщение за внимание.
* Събитие възниква при смяна на избора от „оценки“ на „забележки“. Методът radioButtonGrade\_CheckedChanged го изпълнява и прави връзка с Presenter чрез метода OnGradesButtonClick.
* OnGradesButtonClick от своя страна извиква метода за подготовка и визуализация на данните PrepareDataForView. В else блока става връзка с БД, взема се всичката информация за забележките на посочения ученик и се подготвя за визуализация в dataGridView таблица. Полето view и методът FormatDataDisplay правят визуализацията на данните в учителския изглед.
* С натискане на бутона „Виж дневника“ възниква събитие, изпълнено от SeeDatabtn\_Click. Методът влиза в if-проверка дали един от двата радио бутона „оценки“ или „забележки“ не е избран. В случая, избран е „забележки“, DataGridView таблицата с данните се визуализира, averageGradeTextBox за просвояване на стойността на средноартиметичното от оценките и бутонът за среден успех btnAverage стават видими.
* Ако учителят не е конкретизирал „оценки“ или „забележки“ иска да разгледа, else блокът в SeeDatabtn\_Click изпраща съобщение за внимание.

Нов запис на забележки по даден предмет

* Отбелязва се избор „забележки“. След това, с кликане на „Добави нов запис“ във View възниква събитие, което AddRecordbtn\_Click изпълнява.
* AddRecordbtn\_Click влиза в else if-проверката и проверява дали radioButtonNote е избран. Ако да, подготвя изгледа за добавяне на забележка.
* При натискане на бутона „Добави“, възниква събитие, изпълнено от Savebtn\_Click във View. Savebtn\_Click метода прави връзка с Presenter чрез метода OnSaveNoteButtonClick.
* OnSaveNoteButtonClick от своя страна, влиза в if-проверката дали TextBoxComment със забележката е попълнен.
* Ако да, на локалната променлива note се присвояват данните, попълнени в TeacherForm. Чрез data полето става връзката на Presenter с БД в Model, където SaveChanges записва промените. Чрез полето view изгледът се рестартира и се генерира съобщение за успех.

Ако не, чрез view полето се визуализира съобщение за грешка

### 2.7. Редактиране на родителски/учителски профил.

Формата за редактиране е специфична за родителски и учителски профил. В тези два случая тя е активна. Основната и особеност са различните варианти за редакция , която тя предлага в двата си варианта. Формата за редакция е модален прозорец за учителския и родителския профил. При натискане на бутона за настройки , той се създава върху предишната форма. Деактивира главния прозорец. Потребителите трябва да взаимодействат с модалния прозорец преди да се върнат в главния.

##### 1.IEditView и EditForm.

Потребителската форма за редактиране на профил - EditForm има свой интерфейс IEditForm. В него са декларирани следните свойства:

* Presenters.IEditPresenter Presenter –Пропърти сочещо към IEditPresenter. При различните събития чрез него View се обръща към методите , които се изпълняват в Presenter.
* MainScreenVisible – Пропърти с булева стойност за видимост на главния панел от контроли
* ChangeNameScreenVisible - Пропърти с булева стойност за видимост на панел от контроли за смяна на име и фамилия
* ChangePassScreenVisible – Пропърти с булева стойност за видимост на панел от контроли за промяна на парола
* AddStudentScreenVisible – Пропърти с булева стойност за видимост на панел от контроли за добавяне на нов ученик /важи само за родителски профил/
* FirstNameTextBoxText – Пропърти за име
* LastNameTextBoxText – Пропърти за фамилия
* OldPassTextBoxText - Пропърти за стара парола
* NewPassTextBoxText - Пропърти за нова парола
* NewPassAgainTextBoxText – Пропърти за повтаряне на нова парола
* AddStudentButtonVisible - Пропърти с булева стойност за показване на бутон в MainScreen
* StudentPinTextBoxText – Пропърти за ЕГН на ученик
* StudentCodeTextBoxText – Пропърти за уникален ученически код

И методи:

* DisplayMainScreen() – Метод за показване на основния панел от контроли
* DisplayChangeNameScreen() – Метод за показване на панел от контроли за промяна на име и фамилия
* DisplayChangePassScreen() – Метод за показване на панел от контроли за промяна на паролата
* DisplayAddStudentScreen() - Метод за показване на панел от контроли за добавяне на ученик
* ShowAsModal() – Метод за визуализиране на прозореца за настройки като модален

В конструктора на EditForm е упоменато къде да се построи модалния прозорец на екрана заедно с неговите палени от контроли. Важна особеност на модалния прозорец е , че е изграден от панели /колекции/ от контроли.

Това са panelMain който включва следните контроли:

* Бутон за промяна на име
* Бутон за промяна на парола
* Бутон за добавяне на ученик

panelNames:

* Текстова кутия за име
* Текстова кутия за фамилия

panelPass:

* Текстова кутия за стара парола
* Текстова кутия за нова парола
* Текстова кутия за повтаряне на нова парола

panelStudent:

* Текстова кутия за въвеждане на ЕГН на ученик.
* Текстова кутия за въвеждане на уникален ученически номер.

Пропъртитата получават своите стойности :

* FirstNameTextBoxText – Получава стойност в textBoxFirstName.Text
* LastNameTextBoxText – Получава стойност в textBoxLastName.Text
* OldPassTextBoxText – Получава стойност в textBoxOldPass.Text
* NewPassTextBoxText – Получава стойност в textBoxPass1.Text
* NewPassAgainTextBoxText – Получава стойност в textBoxPass2.Text
* StudentPinTextBoxText – Получава стойност в textBoxStudentPin.Text
* StudentCodeTextBoxText – Получава стойност в textBoxCode.Text

Булевите пропъртита получават стойност:

* MainScreenVisible – получава стойност при видимост на panelMain.
* ChangeNameScreenVisible – получава стойност при видимост на panelNames.
* ChangePassScreenVisible – получава стойност при видимост на panelPass.
* AddStudentScreenVisible – получава стойност при видимост на panelStudent.
* AddStudentButtonVisible – получава стойност от btnAddStudentScreen , когато е видим.

Панелите от контроли се визуализират в потребителския интерфейс чрез имплементацията на следните методи във View:

DisplayAddStudentScreen():

* Скрива всички панели
* Изчиства текста по контролите
* Прави panelStudent видим
* Извлича го пред останалите колекции от контроли

DisplayChangeNameScreen():

* Скрива всички панели.
* Изчиства текста от контролите.
* Прави panelNames видим.
* Извлича го пред останалите колекции от контроли.

DisplayChangePassScreen():

* Скрива всички панели.
* Изчиства текста от контролите.
* Прави panelPass видим.
* Извлича го пред останалите колекции от контроли.

DisplayMainScreen():

* Скрива всички панели.
* Изчиства текста от контролите.
* Прави panelMail видим.
* Извлича го пред останалите колекции от контроли.

Във формата се имплементира метод за визуализация на прозорец със съобщение, но какво точно ще бъде съобщението отново се крие в Presenter.

Методът ShowAsModal() имплементира в себе си метода ShowDialog() , чието предназначение е да покаже прозореца за настройки като модален прозорец.

HideAllPanels() първоначално скрива всички панели от контроли преди Presenter да инструктира View какво да покаже на потребителя.

Методът ClearText() първоначално скрива текста от всички контроли от тип textbox преди Presenter да инструктира View какво да покаже на потребителя.

В EditForm са генерирани следните методи при натискане на контроли от различните панели:

* goBackPic\_Click – при натискане на картина бутон , който се обръща към метода OnBackButtonClick() от EditPresenter за изпълнение на своята функционалност.
* btnChangeNames\_Click – при натискане на бутон за зареждане на панел panelName, който се обръща към метода OnChangeNamesButtonClick() от EditPresenter .
* btnChangePass\_Click – при натискане на бутон за зареждане на panelPass , който се обръща към метода OnChangePassButtonClick() от EditPresenter за изпълнение на своята функционалност.
* btnAddStudent\_Click – при натискане на бутон за зареждане на panelStudent ,който се обръща към метода OnAddStudentButtonClick() от EditPresenter
* btnSaveNameChanges\_Click – при натискане на бутон за запазване на име , който се обръща към метода OnSaveNameChangesButtonClick().
* btnSavePassChanges\_Click – при натискане бутон за запазване на парола , който се обръща към метода OnSavePassChangesButtonClick() от EditPresenter за изпълнение на своята функционалност
* btnSaveStudent\_Click – при натискане на бутон за добавяне на нов ученик , който се обръща към OnSaveStudentButtonClick().

##### 2. IEditPresenter и EditPresenter.

В Presenter се намира основната логика на модалния прозорец EditView. Той съдържа интерфейс IEditPresenter и имплементиращият го клас EditPresenter. В тях са декларирани и имплементирани следните методи, които посочват поведението на приложението след потребителско взаимодействие в потребителския интерфейс:

* Start()
* OnChangeNamesButtonClick();
* OnChangePassButtonClick();
* OnAddStudentButtonClick();
* OnBackButtonClick();
* OnSaveNameChangesButtonClick();
* OnSavePassChangesButtonClick();
* OnSaveStudentButtonClick();

Класа EditPresenter се асоциира с View и Model, чрез полетата view и data и се извличат данни за текущия потребител в полето user. В конструктора му е посочено , че ако текущия потребител е учител във прозореца от View трябва да се скрие бутона за добавяне на нов ученик , защото това е опция само на родителя.

3.Процеси при потребителско взаимодействие

При избор на потребителя да промени име и фамилия :

* Потребителят натиска бутона за промяна на име и фамилия , който се обръща при настъпване на събитието към метода btnChangeNames\_Click.
* EditForm отчита потребителското взаимодействие и изпраща сигнал към EditPresenter.
* Изпълнява се методът OnChangeNamesButtonClick().
* Presenter изпраща заявка до EditForm да изпълни метода DisplayChangeNameScreen(), като при визуализирането на контролите от panelName текстовите полета се изпълват с текущите име и фамилия , които са записани в базата данни.

При избор на потребителя да промени текущата си парола:

* Потребителят натиска бутона за промяна на парола който се обръща при настъпване на събитието към метода btnChangePass\_Click.
* EditForm отчита потребителското взаимодействие и изпраща сигнал към EditPresenter.
* Изпълнява се методът DisplayChangePassScreen().
* Presenter изпраща заявка до EditForm да покаже на екрана контролите от panelPass.

При избор на родителя да добави нов ученик към профила си :

* Потребителят натиска бутона за добавяне на нов ученик ,който се обръща при настъпване на събитието към метода btnAddStudent\_Click.
* EditForm отчита потребителското взаимодействие и изпраща сигнал към EditPresenter.
* Изпълнява се метода OnAddStudentButtonClick()
* EditPresenter изпраща заявка до EditForm да покаже на екрана контролите от panelStudent.

При въвеждане на ново име и фамилия :

* Потребителят натиска бутона за запазване на име и фамилия, който се обръща при настъпване на събитието към метода btnSaveNameChanges\_Click .
* EditForm отчита потребителското взаимодействие и изпраща сигнал към EditPresenter.
* Изпълнява се методът OnSaveNameChangesButtonClick(), който има задача да запази настъпилите промени , след като провери дари да те коректно въведени , чрез If – проверки.
* При случай на не попълнени от потребителя име или фамилия EditPresenter заявява на EditForm да покаже съобщение за настъпилата грешка.
* Данните не се запазват.
* При случай на попълване на име и фамилия с латински символи EditPresenter заявява на EditForm да покаже съобщение за настъпилата грешка.
* Данните не се запазват.
* При случай на коректно въведени име и фамилия новите данни заменят старите в базата данни.
* EditPresenter заявява на EditForm да покаже съобщение за успешно запазване.
* При случай на неуспешно запазване в базата данни EditPresenter заявява на EditForm да покаже съобщение за настъпилата грешка.
* При успешно запазване на данните Presenter заявява на View да покаже контролите от panelMain.

При въвеждане на нова парола:

* Потребителят натиска бутона за запазване на парола , който се обръща при настъпване на събитието към метода btnSavePassChanges\_Click.
* EditForm отчита потребителското взаимодействие и изпраща сигнал към EditPresenter.
* Изпълнява се методът OnSavePassChangesButtonClick(), който има задача да запази настъпилите промени , след като провери дари да те коректно въведени , чрез If – проверки.
* При случай на не попълнени от потребителя текстови полета за парола , нова парола и повтаряне на нова парола EditPresenter заявява на EditForm да покаже съобщение за настъпилата грешка.
* Данните не се запазват.
* При случай грешно въведена стара парола EditPresenter заявява на EditForm да покаже съобщение за настъпилата грешка.
* Данните не се запазват.
* При случай на различно въвеждане в полетата за нова парола и нейното потвърждаване EditPresenter заявява на EditForm да покаже съобщение за настъпилата грешка.
* Данните не се запазват.
* При случай , че новата парола съвпада със старата EditPresenter заявява на EditForm да покаже съобщение за настъпилата грешка.
* Данните не се запазват.
* При случай на нова парола извън ограничението от 8 до 25 символа , EditPresenter заявява на EditForm да покаже съобщение за настъпилата грешка.
* Данните не се запазват.
* При случай на коректно въвеждане на нова парола тя се хешира и се добавя в базата данни за лицето , на което принадлежи.
* EditPresenter заявява на EditForm да покаже съобщение за успешното обновяване на паролата
* EditPresenter заявява на EditForm да покаже контролите от panelMain.

При добавяне на ученик в родителски профил:

* Потребителят натиска бутона за запазване на асоциирания нов , който се обръща при настъпване на събитието към метода btnSaveStudent\_Click.
* EditForm отчита потребителското взаимодействие и изпраща сигнал към EditPresenter.
* Изпълнява се методът OnSaveStudentButtonClick(), който има задача да запази настъпилите промени , след като провери дари да те коректно въведени , чрез If – проверки.
* При случай на не коректно въвеждане на ЕГН от потребителя , EditPresenter заявява на EditForm да покаже съобщение за настъпилата грешка.
* При случай на невалидно въвеждане на секретен код , EditPresenter заявява на EditForm да покаже съобщение за настъпилата грешка.
* Ученикът , който трябва да се асоциира от родителя е предварително добавен от администратора в базата данни.
* Presenter изпраща заявка към Model да провери дали има данни за лицето с това ЕГН.
* При случай , че ученикът не съществува в базата данни EditPresenter заявява на EditForm да покаже съобщение за настъпилата грешка.
* При случай , че ученикът е вече асоцииран с родителя в базата данни EditPresenter заявява на EditForm да покаже съобщение за настъпилата грешка.
* При случай , че лицето е успешно асоциирано с родителския профил EditPresenter заявява на EditForm да покаже съобщение за успешното добавяне.
* При случай на неуспешно асоцииране на ученик с родител в базата данни EditPresenter заявява на EditForm да покаже съобщение за възникналата грешка.
* След извършване на операциите EditPresenter заявява на EditForm да покаже контролите от panelMain.

### 2.8.Администраторски профил

Администраторът получава достъп до профила си в EduWatch след коректно въведени и приети от системата входни данни (потребителско име и парола).

##### 1.IAdminView и AdminForm

Интерфейсът IView се имплементира от този на администраторския профил IAdminView. Методите, декларирани в IAdminView, дефинират какъв тип поведение демонстрира обектът Admin при взаимодействие с дизайна на администраторския профил. Същите тези методи са имплементирани в AdminForm:

* Presenters.AdminPresenter Presenter { set; } – пропърти за връзка на IAdminView с AdminPresenter. Чрез него IAdminView се обръща към методите , които се изпълняват в AdminPresenter.
* AdminLabelText { get; set; } – пропърти, чрез което adminLabelText получава стойност имената на администратора, импортирани от базата данни;
* ComboBoxProfile { get; set; } – пропърти, чрез което comboBoxProfile получава стойности за избор на вид профил (учителски, родителски или ученически);
* FirstNTextBox { get; set; } – пропърти за попълване във firstNTextBox на първите имена на обектите, чиито профили ще добавя/изтрива потребителят;
* LastNTextBox { get; set; } - пропърти за попълване в lastNTextBox на фамилиите на обектите, чиито профили ще добавя/изтрива потребителят;
* UserNameTextBox { get; set; } – пропърти за попълване в usernameTextBox на потребителско име на учителя при добавяне на такъв профил;
* ComboBoxUserName { get; set; } – пропърти, чрез което СomboBoxUserName за потребителско име получава стойности при изтриване на родителски или учителски профил;
* TextBoxSubject { get; set; } – пропърти за попълване на subjectTextBox с предмета, по който преподава учителят, при добавяне на учителски профил;
* ComboBoxGrade { get; set; } – пропърти чрез което comboBoxGrade за класовете, в които може да се обучава ученикът, получава стойности. Използва се при добавяне на ученически профил;
* PINTextBox { get; set; } – пропърти за попълване на PINTextBox с ЕГН на ученика при добавяне на ученически профил;
* PasswordTextBox { get; set; } – пропърти за попълване на passwordTextBoxс парола при добавяне на учителски профил;
* SelectedUserNameID { get; } - връща номер, който е различен за всеки елемент, който може да бъде избран от combobox-а. Така в presenter-a може да се разбере кой точно е избран.
* string typeOfProfile { get;} – пропърти, което взема стойността на вида профил от comboBoxProfile и я преобразува в string;
* IsbtnDeleteVisible { get; } – пропърти, което прави бутонът за изтриване btnDelete видим в администраторската форма;
* IsbtnAddVisible { get; } - пропърти, което прави бутонът за добавяне btnAdd на профил видим в администраторската форма;
* IsUserNameTextBoxVisible { get; } - пропърти, което прави userNameTextBox видим в администраторската форма;
* IsUserNameComboBoxVisible { get; } - пропърти, което прави userNameComboBox видим в администраторската форма;
* IsPINTextBoxVisible { get; } - пропърти, което прави PINTextBox видим в администраторската форма;
* IsBTNAddisible { get; } - пропърти, което прави addInfoBTN – бутонът за добавяне на профил след попълнена нужната информация за ученик и учител видим в администраторската форма;
* IsBTNDeleteVisible { get; set; } - пропърти, което прави deleteInfoBTN – бутонът за изтриване на профил след попълнена нужната информация за ученик, учител и родител видим в администраторската форма;
* IsFirstNTextBoxVisible { get; } - пропърти, което прави firstNTextBox – място за автоматично попълване на име при изтриване на трите вида профили и за попълване при добавяне на учителски/ученически профил, видим в администраторската форма;
* IsLastNTextBoxVisible { get; } - пропърти, което прави lastNTextBox – място за автоматично попълване на фамилия при изтриване на трите вида профили и за попълване при добавяне на учителски/ученически профил видим в администраторската форма;
* IsSubjectGradeLabelVisible { get; } - пропърти, което прави subjectGradeLabel видим в администраторската форма;
* IsStudentNumberTextBoxVisivle { get; } - пропърти, което прави studentNumberTextBox видим в администраторската форма;
* IsPINLabelVisible { get; } - пропърти, което прави PINLabel видим в администраторската форма;
* IsPasswordTextBoxVisible { get; } - пропърти, което прави passwordTextBox видим в администраторската форма;
* IsPasswordLabelVisible { get; } - пропърти, което прави passwordLabel видим в администраторската форма;
* IsCheckBtnVisible { get; } - пропърти, което прави checkBtn видим в администраторската форма;
* IsSubjectTextBoxVisible { get; } - пропърти, което прави checkBtn видим в администраторската форма;
* FillInProfile(string[] typeOfProfiles) – метод, който дава стойности „родител“, „учител“ и „ученик“ на comboBoxProfile, падащото меню с профилите;
* void FillInWhichGrade(string[] grade) – метод, който дава стойности от 8 до 12 вкл. на comboBoxGrade при добавяне на ученик в системата ;
* FillInCorrespondingUserNames(string[] usernames) – метод, който дава стойности под формата на потребителски имена на comboBoxUsername при изтриване на родителски и ученически профил;
* AdminStartView() – метод, който показва администраторската форма във вид на диалогов прозорец с контроли ;
* HideAll() – метод, който прави всички контроли за взаимодействие с администраторския изглед невидими в случай на рестарт на изгледа;

Чрез изброените свойства Presenter (AdminPresenter) прави връзка с View (AdminForm).

Дизайнът на администраторската форма включва два бутона съответно за изтриване и добавяне на профил, още два за изтриване и добавяне след въведени конкретни данни; TextBox за потр.име, номер на ученика, име и фамилия, парола, ЕГН, предмет; ComboBox за избор на потр.име, за избор на клас и лейбъли, уточняващи местата за попълване на информацията.

##### 2.IAdminPresenter и AdminPresenter

Класът AdminPresenter осъществява връзка с View чрез полето view, с базата данни – чрез полето data, с Login формата – чрез loginPresenter. Чрез полето user се дава ролята администратор на потребителя.

В интерфейса IAdminPresenter са декларирани методите, свързани със събитията, предизвикани от администратора в администраторския изглед. AdminPresenter имплементира и пренаписва тези методи.

* Start() – пропърти, чрез което, през полето view се стартира диалоговия прозорец на администраторската форма.
* OnExitButtonClick() – метод, изпълняващ събитие, предизвикано с натискане на бутона „Изход“. Затваря администраторския изглед и визуализира LoginForm-ата.
* OnSaveClick() – метод, който се активира с натискане на бутона „Добави“при събитието добавяне на ученически/родителски профил. Съдържа switch блок със случаи за ученик и учител с вложени if-проверки и try-catch блок за реакция на приложението, ако промените са, или съответно не са запазени;
* OnRemoveClick() - метод, който се активира с натискане на бутона „Изтрий“при събитието изтриване на ученически/родителски профил. Съдържа switch блок със случаи за ученик и учител с вложени if-проверки и try-catch блок за реакция на приложението, ако промените са, или съответно не са запазени;
* OnTeacherRemove() – метод, който при събитието изтриване на учителски профил, чрез полето data достъпва базата данни и извлича потр.имена на учителите, чрез полето view и метода FillInCorrespondingUserNames попълва потр.имена в comboBoxUserName;
* OnParentRemove() - метод, който при събитието изтриване на родителски профил, чрез полето data достъпва базата данни и извлича потр.имена на родителите, чрез полето view и метода FillInCorrespondingUserNames попълва потр.имена в comboBoxUserName;;
* OnTeacherSelection() – метод, който при избор за изтриване на учителски профил, чрез полето data прави връзка с БД в Model и взема всички потр.имена на учителите, чрез view данните се показват в сomboBoxUserName. Въз основа на потр.име, името и фамилията на избрания учител се генерират автоматично съответно във firstNTextBox и lastNTextBox, чрез полето view се визуализират в администраторския изглед ;
* OnParentSelection() - метод, който при избор за изтриване на родителски профил, чрез полето data прави връзка с БД в Model и взема всички потр.имена на родителите, чрез view данните се показват в сomboBoxUserName. Въз основа на потр.име, името и фамилията на избрания учител се генерират автоматично съответно във firstNTextBox и lastNTextBox, чрез полето view се визуализират в администраторския изглед ;
* OnStudentSelection() – метод с try-catch блок. Активира се при иницииране на действие изтриване на ученически профил. Проверява дали въведеното от админа ЕГН в PINTextBox отговаря на ограниченията. Ако не – чрез view връща съобщение за грешка. Ако да – на база ЕГН на ученика, името и фамилията на избрания ученик се генерират автоматично съответно във firstNTextBox и lastNTextBox, чрез полето view се визуализират в администраторския изглед;

##### 3.Процеси при потребителско взаимодействие

Администраторът може да трие учителски, родителски и ученически профил, но може да добавя само ученически и учителски профил.

Инициализацията на изгледа става в конструктора AdminPresenter. Админ се асоциира с ролята си като локалната променлива admin приема стойността на параметъра user – осъществява се връзка с БД в Model. аdminLabelText получава стойности от БД под формата на име и фамилия. Чрез метода FillInProfile, в comboBoxProfile се зареждат видовете профили, чрез полето view става визуализацията им.

Изтриване на родителски профил

* View регистрира събитието чрез метода deleteBtn\_Click. firstNTextBox и lastNTextBox не са активни до избора на потр.име на родител от comboBoxUsername. Чрез else-if проверка за тип профил View подготвя изгледът за действието изтриване на родителски профил. View прави връзка с Presenter чрез метода OnParentRemove.
* След избор на конкретно потр.име на родител от comboBoxUsername, възниква ново събитие в Presenter,което OnParentSelection изпълнява. firstNTextBox и lastNTextBox са автоматично попълнени на база потр.име на родителя.
* Последното събитие се случва с натискане на бутона „Изтрий“. Методът DeleteInfoBTN\_Click (вика OnRemoveClick, изчиства текущия избор, премахва конкретния избор на потр.име в comboBoxUsername и го връща в начално състояние) в ParentForm прави връзка с Presenter. OnRemoveClick има switch блок. В case за Родител полето data достъпва БД в Model и, в зависимост от направения през ComboBoxUserName избор на потр.име, информацията за родителя се премахва от базата данни. Ако действието е било успешно, try-catch блокът генерира съобщение за успех, ако не- такова за грешка.

Изтриване на учителски профил

* View регистрира събитието чрез метода deleteBtn\_Click. firstNTextBox и lastNTextBox не са активни до избора на потр.име на учител от comboBoxUsername. Чрез if проверка за тип профил View подготвя изгледът за действието изтриване на учителски профил. View прави връзка с Presenter чрез метода OnTeacherRemove.
* След избор на конкретно потр.име на учител от comboBoxUsername, възниква ново събитие в Presenter,което OnTeacherSelection изпълнява. firstNTextBox и lastNTextBox са автоматично попълнени на база потр.име на учителя.
* Последното събитие се случва с натискане на бутона „Изтрий“. Методът DeleteInfoBTN\_Click в ParentForm прави връзка с Presenter. OnRemoveClick има switch блок. В case за Учител полето data достъпва БД в Model и, в зависимост от направения през ComboBoxUserName избор на потр.име, информацията за учителя се премахва от базата данни. Ако действието е било успешно, try-catch блокът генерира съобщение за успех, ако не- такова за грешка.

Изтриване на ученически профил

* View регистрира събитието чрез метода deleteBtn\_Click. Чрез else-if проверка за тип профил View подготвя изгледът за действието изтриване на ученически профил.В PINTextBox админът трябва да въведе ЕГН. С натискане на червеното тикче възниква събитие във View, което checkPictureBox\_Click изпълнява. Методът прави връзка с Presenter и метода OnStudentSelection.
* OnStudentSelection представлява try-catch блок. Прави се проверка дали въведеното ЕГН отговаря на условията. Ако не – View изпраща съобщение за невалидно ЕГН. Ако да – чрез полето data се достъпва БД и въз основа на ЕГН-то firstNTextBox и lastNTextBox получават стойности съответно име и фамилия на ученика. Ако въведеното ЕГН отговаря на условията, но за него няма ученик в БД, методът изпраща съответното съобщение.
* С натискане на бутона „Изтрий“ възниква последно събитие. Presenter реагира с метода OnRemoveClick. В case за Ученик полето data достъпва БД в Model и, въз основа на въведеното ЕГН, информацията за ученика се премахва от базата данни. Ако действието е било успешно, try-catch блокът генерира съобщение за успех, ако не- такова за грешка.

Добавяне на учителски профил

* View регистрира събитието чрез метода addBtn\_Click. Чрез if- проверка за тип профил View подготвя изгледът за действието добавяне на учителски профил.
* Администраторът въвежда необходимите данни в определените за целта полета.
* С натискане на бутона „Добави“ възниква събитие. Presenter реагира с метода OnSaveClick, в който има switch-блок. В case за учител има вложени if-проверки. Ако необходимите полета с данни са попълнени, се правят две нови проверки – за вече съществуващ учител в БД(чрез полето data Presenter се свързва с БД и проверява) и за коректно въведена парола (между 8 и 25 символа).
* Ако предложената парола е коректно въведена и учителят е нов за системата, PasswordHash хешира паролата, учителят получава своето ID в БД. БД получава учителските данни и предмета от полетата, попълнени от админа, чрез view. Data добавя данните в БД.

Добавяне на ученически профил

* View регистрира събитието чрез метода addBtn\_Click. Чрез else if- проверка за тип профил View подготвя изгледа за действието добавяне на ученически профил.
* Администраторът въвежда необходимите данни в определените за целта полета и избира клас от падащото меню (comboBoxGrade).
* С натискане на бутона „Добави“ възниква събитие. Presenter реагира с метода OnSaveClick. В case за Ученик if-else проверката проверява дали всички полета за данни са попълнени. Ако да, информацията за новия уеник се взима от полетата, попълнени от админа, чрез view. Data добавя данните в БД. Показва се съобщение , че е генериран уникален секретен номер на ученика , с който родителят може да го асоциира към себе си.Създава се текстови файл на устройството , в който се записват ЕГН , име , фамилия и секретния код на ученика. Родителя взима трябва за вземе номера от самото училище.

### 2.9. Допълнителни елементи и становища.

**Парола:**

В „EduWatch“ е създаден класа PasswordHash , който имплементира интерфейс IPasswordHash. Функцията на този клас е да хешира паролите на потребителите и да провери дали парола съответства на даден хеш.

За защита на паролите се използва криптографския метод осоляване /Salt/. Това е концепция , която се отнася до хеширането на паролата. По същество солта е уникална стойност , която се добавя в края на паролата .Целта е процеса на хеширане да бъде защитен и да добави сигурност към информацията в базата данни. Посолявайки паролите се скриват истинските им хеш стойности. Това прави системата защитена от атаки с груба сила.

**База данни:**

Приложението е налично в две конфигурации на базата данни. При едната се използва SQL Server (в частност по време на разработката - LocalDB), а при другата MySQL. Могат да бъдат утилизирани и други алтернативи, поддържани от Entity Framework.

Начинът на имплементация на използваните по време на разработката технологии не е подходящ за употреба в продукционна среда, тъй като би довел до сериозни рискове за сигурността.

С цел улесняване на работата по време на разработка информацията за връзка към базата данни се съхранява в явен вид, което я прави уязвима.

За подобряване на сигурността е необходимо да бъдат взети редица мерки като осъществяване на криптирана връзка, защита на низа за връзка към източника на данните и криптиране на конфигурационните файлове. Съхраняване на информация за връзката в явен вид може да доведе до компрометиране на цялата система.

За да бъде сигурността на системата още по-голяма, е необходимо текстовият низ за връзка с базата данни да не бъде достъпен до крайния потребител, дори в криптиран вид.

Добър вариант е разширяване на системата с посредник като уеб услуга от типа „REST API“. Тя е отделена от приложението, използвано от потребителите и е под контрола на хората, занимаващи се с поддръжката на продукта. Негатив тук е увеличаване на сложността и затрудняване на поддръжката.

Друга алтернатива е изграждането на виртуална частна мрежа (VPN), с която да се осигури сигурна връзка между потребителите и необходимите им информационни ресурси. За целта е нужно и осигуряване на техническа поддръжка при първоначалното свързване на даден потребител към системата.