ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СОФИЯ

**Факултет „Компютърни системи и управление“**

Техническа Документация

“Questions”

Задание: Тестова система. Да се проектира и реализира програма позволяваща провеждането на тестове за проверяване на знания в дадена област. Да се предостави възможност за добавянето на нови въпроси. Да се включат и въпроси с отворен отговор (например посочване на отговор конкретно число). Като опции на програмата да се включат режими на тестване с време и без. Възможност за печат на тест, печат или запис на справка с резултати от положени тестове.

Екип: Димитър Киряков фак.№:501214020,

Денис Заимов фак.№:501214031,

Дата: 10.01.2018 г.

# Съдържание

[1 Увод 4](#_Toc503215544)

[2 Съществуващи разработки 4](#_Toc503215545)

[2.1 Question Generator 4](#_Toc503215546)

[2.2 Random Questions Generator 4](#_Toc503215547)

[2.3 Date Question Generator 4](#_Toc503215548)

[3 Проектиране 5](#_Toc503215549)

[3.1 Предназначение 5](#_Toc503215550)

[3.2 Системни изисквания 5](#_Toc503215551)

[3.2.1 Функционални изисквания 5](#_Toc503215552)

[3.2.2 Нефункционални изисквания 6](#_Toc503215553)

[3.3 Изследване на потребителските изисквания 6](#_Toc503215554)

[3.3.1 Потребителски истории 6](#_Toc503215555)

[3.3.2 Критерии за приемане на потребителски истории 7](#_Toc503215556)

[3.4 Използвани данни 7](#_Toc503215557)

[3.5 Комуникация между изгледите 9](#_Toc503215558)

[3.6 Използван модел за реализацията на приложението 10](#_Toc503215559)

[4 Реализация 11](#_Toc503215560)

[4.1 Бази Данни 11](#_Toc503215561)

[4.2 Реализация на Views 11](#_Toc503215562)

[5 Потребителско ръководство 21](#_Toc503215563)

[5.1 Регистрация на потребител 21](#_Toc503215564)

[5.2 Главен прозорец 22](#_Toc503215565)

[5.3 Добавяне на въпрос 22](#_Toc503215566)

[5.4 Премахване на въпрос 23](#_Toc503215567)

[5.5 Решаване на тест 23](#_Toc503215568)

[6 Заключение 24](#_Toc503215569)

[7 Използвана литература 25](#_Toc503215570)

# Увод

Главната цел на проекта е създаване на приложение, чрез което лесно и бързо да се съставя тест от въпроси. То предоставя възможност за правене на тест на място, както и за проверка на резултати от тест.

# Съществуващи разработки

## Question Generator

Това е приложение създадено от фирмата Hook Education Ltd. То предоставя възможност на учители и ученици да го използват за задаване на въпроси по различни предмети с опции, като тази да export-тират любими за тях въпроси.

Предимствата му са:

* Лесен за употреба
* Възможност за export на въпроси
* Въпросите са предифинирани

Недостатъците му са:

* Няма възможност за избиране на конктретни въпроси, а чакаш да видиш кой ще се падне
* Няма възможност за добавяне на собствени въпроси
* Няма възможност за триене на въпроси
* Няма възможност за създаване на собствен тест от въпроси
* Няма възможност за директно правене на тест в приложението

## Random Questions Generator

Това приложение е създадено от фирмата Apptomotion. То е предназначено за забаление и предоставя възможност за задаване на random въпроси на потребителя.

Предимствата му са:

* Лесен за употреба
* Въпросите са предифинирани

Недостатъците му са:

* Няма възможност за създаване на собствени въпроси
* Няма възможност за правене на тест в приложението на избрана тематика
* Няма възможност за създаване на собствен тест от въпроси
* Няма възможност за exporti-ране на въпроси

## Date Question Generator

Това приложение е създадено от фирмата Kitty Pup Games. То е предназначено за забавление и по – конктретно за въпроси свързани с опознаването на даден човек.

Предимствата му са:

* Лесен за употреба
* Въпросите са предифинирани

Недостатъците му са:

* Въпросите са само в една категория, няма възможност за дефиниране на различни категории.
* Няма възможност за експортиране на въпроси.
* Няма възможност за дефиниране на собствени въпроси.
* Няма възможност за създаване на собствен тест.

# Проектиране

## Предназначение

Системата обхваща двете главни звена, участващи в гореописаната дейност – потребител и приложение. По-конкретно, системата ще се грижи за целия процес, от регистрация на потребители, вписване на потребители, следване на потребителски резултати, добавяне на нови въпроси от потребител в определена секция, премахване на въпроси, възможност за решаване на тест от определена категория на място с време и без време или печат на тест от различна категория.

## Системни изисквания

### Функционални изисквания

| **Номер** | **Изисквания** | **Приоритет** |
| --- | --- | --- |
| 1 | При избиране на полето „Sign up“ от главното меню на системата, трябва да се зареди нов прозорец който да визуализира полета с необходимата информация за регистриране на потребител и след това автоматично да се логва като този потребител | M |
| 2 | При избиране на полето „Sign in“ от главното меню на системата, трябва да се зареди нов прозорец, който да визуализира полета ,в които да се въведе необходимата информация да се логне потребител | M |
| 3 | При зареждане на началния прозорец след като даден потребител се е логнал трябва да се вижда неговото потребителско име. | L |
| 4 | При показване на началния прозорец, трябва да има възможност за започване на тест. | H |
| 5 | След като се визуализира списъкът от категории да може да се избира всяка една от тях | H |
| 6 | В началния прозорец трябва да има опция за изтриване на въпрос от категория. | M |
| 7 | В началния прозорец трябва да има опция за добавяне на нов въпрос към съответна категория. | M |
| 8 | В главното меню, трябва да има бутон за правене на тест по избрана категория с и без време. | H |
| 9 | В главното меню, трябва да има опция за потребителя с която той да се отпише от профила си. | L |
| 10 | В главния прозорец трябва да има опция да се виждат всички резултати от потребителя от изминали тестове и тяхните категории. | M |

### Нефункционални изисквания

| **Номер** | **Изисквания** | **Приоритет** |
| --- | --- | --- |
| 1 | Системата трябва да бъде реализирана с обектно-ориентиран език за програмиране – C# | H |
| 2 | Предпочитани технологии MVVM pattern | H |
| 3 | Системата трябва да бъде реализирана като WPF приложение | H |
| 4 | Системата трябва да ползва релационна база от данни | H |

## Изследване на потребителските изисквания

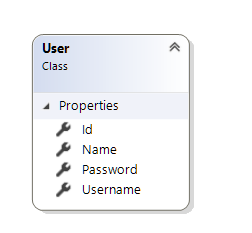
### Потребителски истории

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Номер на потр. историята** | **Като** | **аз искам да** | **така, че** | **точки на историята** |
| 1 | Потребител | мога да създавам въпроси, | при недостатъчно въпроси в определен раздел, да мога да добавя нови. | 50 |
| 2 | Потребител | мога да влизам в системата с потребителско име | да мога да виждам резултатите от изминалите ми тестове | 20 |
| 3 | Потребител | мога да избирам категории, | да стесня кръга от задаваните ми въпроси | 50 |
| 4 | Потребител | мога да правя тестове | да подобря моите познания в дадена област | 50 |
| 5 | Потребител | мога да определям време за правене на даден тест | да имам възможност за по задълбочено мислене по дадени въпроси | 50 |
| 6 | Потребител | мога да се отписвам от системата | друг да не вижда моите резултати | 50 |
| 7 | Потребите | мога да извадя тестовите въпроси във файл | да предоставя теста и на други хора да го направят | 50 |
| 8 | Потребител | създава и въпроси с отворен отговор | да увелича трудността на въпроса | 20 |

### Критерии за приемане на потребителски истории

|  |  |
| --- | --- |
| **Номер на потр. История** | Критерии за приемане |
| 1 | Потребителят да има възможност да създава въпроси. |
| 2 | Потребителят да има възможност да се логва в системата и да вижда своите резултати |
| 3 | Потребителят да има възможност за определяне на категории на даден тест |
| 4 | Потребителят да има възможност да прави тестове в приложението |
| 5 | Потребителят да има възможност да определя дали теста да е с време или без |
| 6 | Потребителят да има възможност да се отписва от системата. |
| 7 | Потребителят да има възможност да export-не теста в текстов файл |
| 8 | Потребителят да има възможност да определя дали въпроса, който създава да е с отворен или затворен отговор |
| 9 | Потребителят да има възможност да export-не теста |

## Използвани данни

Всеки потребител, който използва приложението бива описан от класа User който съдържа:

- идентификатор на потребителя,

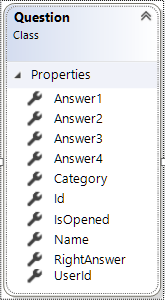
- потр.име,

- парола,

- име на потребителя.

След като се регистрира, като попълни своите данни, той може да се вписва с неговото потр. име и парола, тъй като всеки потребител има уникално потр.име.

Когато един потребител се впише в системата на приложението, той има право да избира категория и да решава тест на въпроси от избраната категория, или да принтира тест от тази категория. Също така, може да добавя въпроси от избрана категория, или да премахва, но само тези които той е добавил от избраната категория.

Всеки въпрос се описва от класа Question който съдържа:

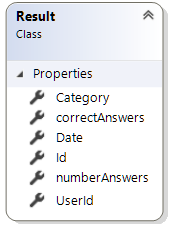
- категория,

- име на въпроса,

- дали той е с отворен отговор, или с избираем отговор,  
- отговорите от които потребителя може да избира, ако е с избираем отговор,

- идентификатор на потребителя който го е създал, за да може да го премахне ако пожелае,

- верния отговор на въпроса.

След решаването на тест от потребителя, той вижда резултат си, който се описва от класа Result който съдържа:

- идентификатор,

- категория на тест който е направен,

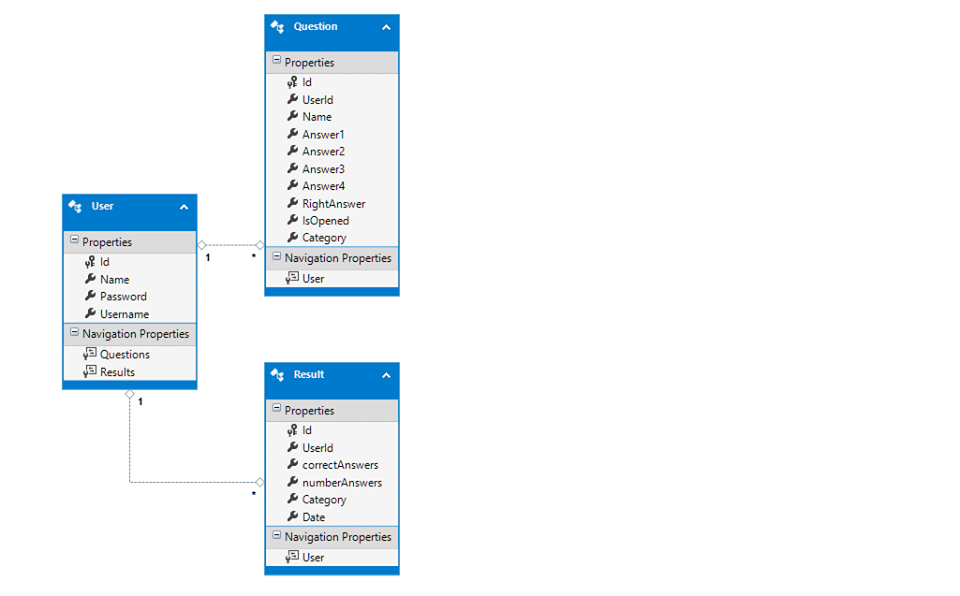
- броя на правилните отговори,

- датата на провеждане на теста,

- броя на отговорите на теста,

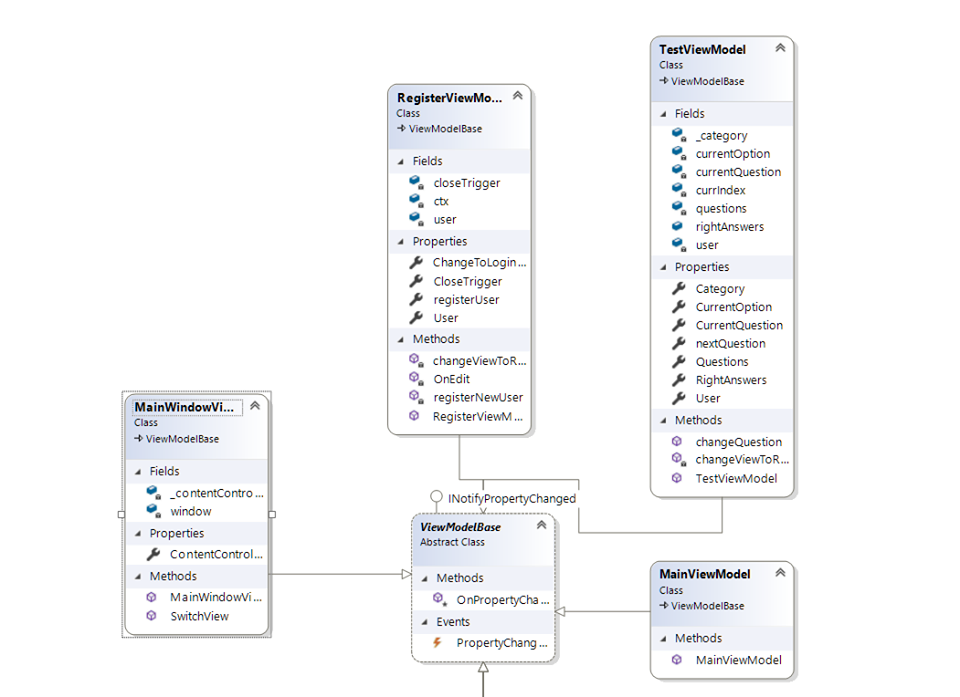
- идентификатор на потребителя.

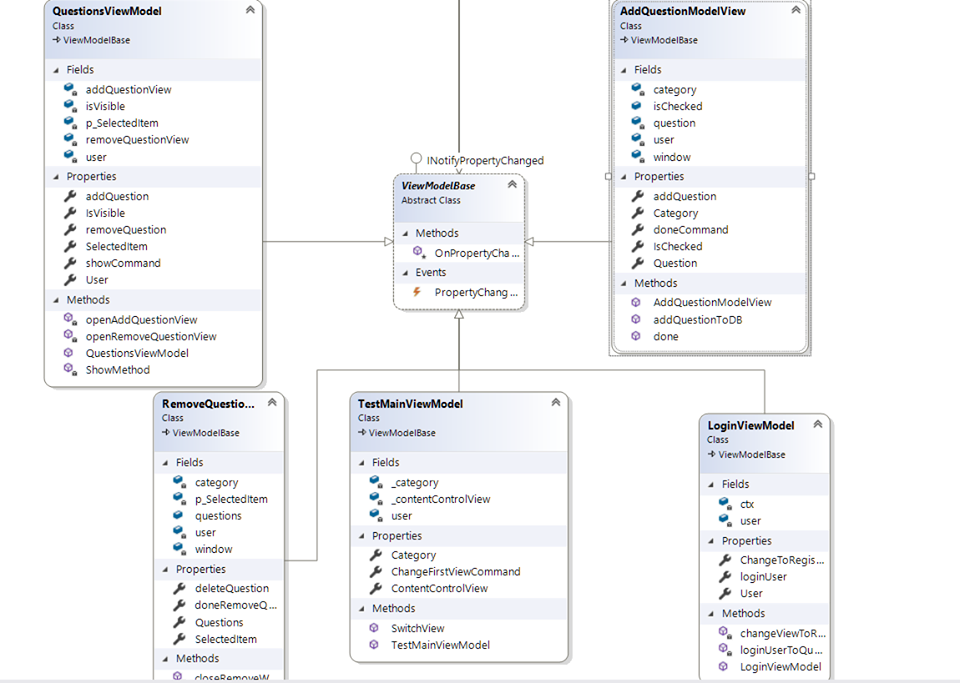
Тези обекти се съхраняват в релационна база от данни със следните връзки:



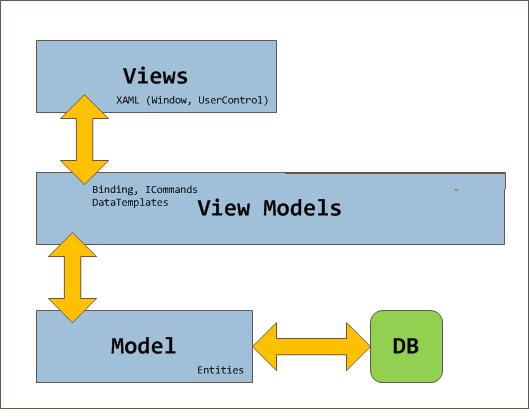
## Комуникация между изгледите

Описаните по горе обекти се предват посредсвтом различни изгледи. Тяхната комуникация може да я проследим от следната диаграма:





## Използван модел за реализацията на приложението



# Реализация

## Бази Данни

Всяка една от таблици е създадена директно през visual studio. Ето и скрипта за всяка една от тях:

CREATE TABLE [dbo].[User] (

[ Id] INT NOT NULL,

[Name] NVARCHAR (50) NOT NULL,

[Password] NVARCHAR (50) NOT NULL,

[Username] NVARCHAR (50) NOT NULL,

PRIMARY KEY CLUSTERED ([Id] ASC)

);

Това представлява User таблицата. Тя съдържа информацията за потребителите на приложението, които ще се извършват всички операции в него.

CREATE TABLE [dbo].[Results] (

[Id] INT NOT NULL,

[UserId] INT NOT NULL,

[correctAnswers] INT NULL,

[numberAnswers] INT NULL,

[Category] NVARCHAR (100) NULL,

[Date] NVARCHAR (50) NULL,

PRIMARY KEY CLUSTERED ([Id] ASC),

FOREIGN KEY ([UserId]) REFERENCES [dbo].[User] ([Id])

);

В тази таблица ще се пазят резултатите от положените тестове на всеки от потребителите.

CREATE TABLE [dbo].[Question] (

[Id] INT NOT NULL,

[UserId] INT NOT NULL,

[Name] NVARCHAR (200) NOT NULL,

[Answer1] NVARCHAR (200) NULL,

[Answer2] NVARCHAR (200) NULL,

[Answer3] NVARCHAR (200) NULL,

[Answer4] NVARCHAR (200) NULL,

[RightAnswer] NVARCHAR (200) NOT NULL,

[IsOpened] SMALLINT NOT NULL,

[Category] NVARCHAR (200) NOT NULL,

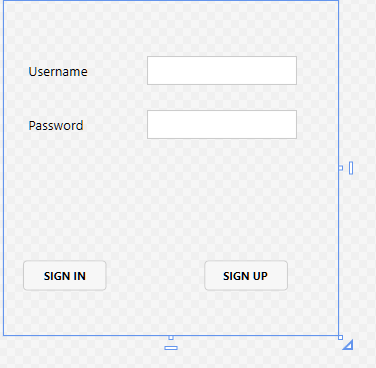
PRIMARY KEY CLUSTERED ([Id] ASC),

CONSTRAINT [userID] FOREIGN KEY ([UserId]) REFERENCES [dbo].[User] ([Id])

);

Тази таблица е основната за въвеждането на въпроси и провеждането на тестовете. От нея ще се извлича цялата информация за генерирането на определен тест. Тази таблица е главната единица за осъществяването на основната идея на приложението.

## Реализация на Views

 **LoginView**

User Control Login View ще се зарежда в MainWindowView. То е реализирано от два бутона и две полета. Това е първоначалната контрола, която ще се визуализира след стартирането на приложението.

Ето и по – важната част от реализирания код на неговия datasource:

public LoginViewModel()

{

ctx = new UserEntities();

user = new User();

}

public ICommand ChangeToRegisterViewCommand

{

get

{

return new RelayCommand(changeViewToRegister,"registerView");

}

}

public ICommand loginUser

{

get

{

return new RegisterLoginCommand(loginUserToQuestions);

}

}

private void changeViewToRegister(string view)

{

Messenger.Default.Send<SwitchViewMessage>(new SwitchViewMessage { ViewName = view });

}

private void loginUserToQuestions(PasswordBox pwBox)

{

user.Password = pwBox.Password;

List<User> listOfUsers = (from a in ctx.Users select a).ToList();

for(int i = 0; i < listOfUsers.Count; i++)

{

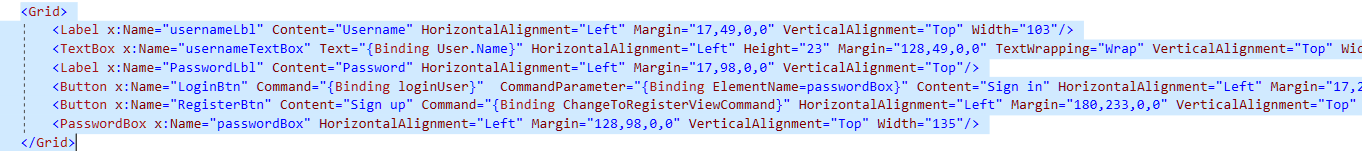
if(user.Name.Equals(listOfUsers.ElementAt(i).Name) && user.Password.Equals(listOfUsers.ElementAt(i).Password) ){

Messenger.Default.Send<SwitchViewMessage>(new SwitchViewMessage { ViewName = "TestView" , User = listOfUsers.ElementAt(i)});

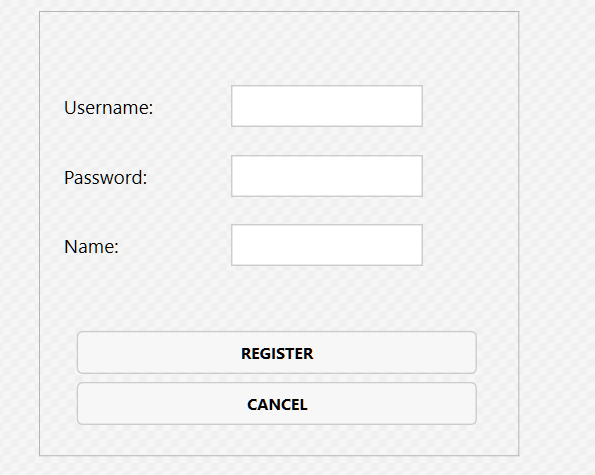
}

}

}

 Ето и част от xaml файла:

**RegisterView**



Този User Control ще се използва за регистрация на потребителите. То ще се зарежда в MainWindowView и ще бъде управлявано от RegisterViewModel класа.

Ето и по – важната част от реализирания код на неговия datasource:

public ICommand registerUser

{

get

{

return new RegisterLoginCommand(registerNewUser);

}

}

public ICommand ChangeToLoginViewCommand

{

get

{

return new RelayCommand(changeViewToRegister, "loginView");

}

}

private void OnEdit(PasswordBox itemToEdit)

{

}

private void changeViewToRegister(string view)

{

Messenger.Default.Send<SwitchViewMessage>(new SwitchViewMessage { ViewName = view });

}

private void registerNewUser(PasswordBox pwBox)

{

user.Password = pwBox.Password;

List<User> listOfUsers = (from a in ctx.Users select a).ToList();

user.Id = listOfUsers.Count;

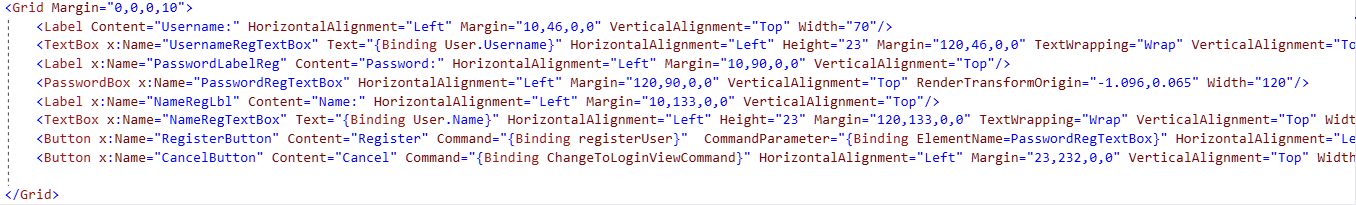
ctx.Users.Add(user);

ctx.SaveChanges();

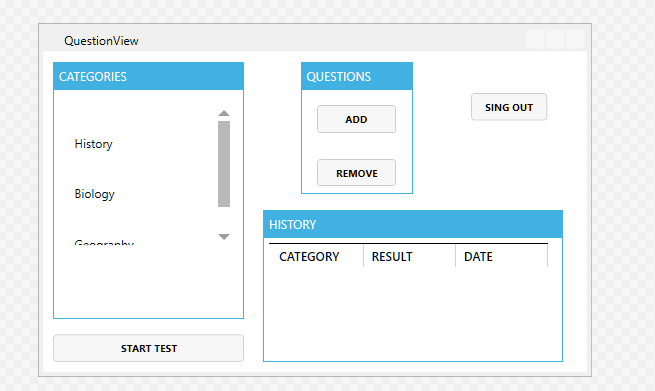
Messenger.Default.Send<SwitchViewMessage>(new SwitchViewMessage { ViewName = "TestView" , User = user });

}

Част от xaml реализацията:



**QuestionView**



Това представлява основното view на приложението. Същинската информация ще бъде визуализирана в този прозорец и ще олицетворява главното меню на приложението.

Състои се от няколко части:

* Categories – съдържа категориите, за които ще могат да се правят тестове и да се създават нови въпроси. За изпълнението на който и да е от бутоните се изисква преди това да бъде избрана категория
* History – този GroupBox представлява историята на направените тестове от вписалия се потребител
* Start Test button – бутона, от който ще стартира тест
* Add button – бутон за създаване на въпрос по избрана категория
* Remove button – бутон за премахване на въпрос по избрана категория
* Sign Out button – бутон за излизане от главното меню

Ето и по – важната част от реализирания код на неговия datasource:

private Boolean isVisible = false;

public Boolean IsVisible

{

get

{

return isVisible;

}

set

{

isVisible = value;

OnPropertyChanged(nameof(IsVisible));

}

}

public User User

{

get

{

return user;

}

set

{

user = value;

OnPropertyChanged(nameof(User));

}

}

public QuestionsViewModel(User user)

{

addQuestionView = new AddQuestion();

removeQuestionView = new RemoveQuestion();

User = user;

}

private ListBoxItem p\_SelectedItem;

public ListBoxItem SelectedItem

{

get { return p\_SelectedItem; }

set

{

IsVisible = true;

p\_SelectedItem = value;

OnPropertyChanged("SelectedItem");

}

}

public ICommand addQuestion

{

get

{

return new QuestionCommand(openAddQuestionView);

}

}

public ICommand removeQuestion

{

get

{

return new QuestionCommand(openRemoveQuestionView);

}

}

private void openAddQuestionView()

{

addQuestionView.Show();

addQuestionView.DataContext = new AddQuestionModelView(SelectedItem, addQuestionView, user);

}

private void openRemoveQuestionView()

{

removeQuestionView.Show();

removeQuestionView.DataContext = new RemoveQuestionViewModel(user, (string)SelectedItem.Content, removeQuestionView);

}

public ICommand showCommand

{

get

{

return new QuestionCommand(ShowMethod);

}

}

private void ShowMethod()

{

TestMainWindow testView = new TestMainWindow();

testView.DataContext = new TestMainViewModel(SelectedItem, user);

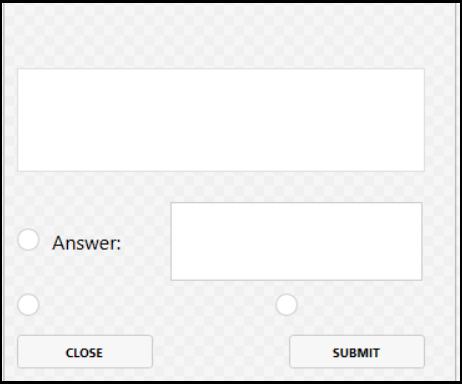
testView.ShowDialog();

}

Част от xaml файла:



**TestView**



Този User Control ще се визуализира в TestMainWindow. Чрез него ще се извършва изпълнението на теста. В зависимост от вида на въпроса ще се показват или 4 възможни отговора, или поле за отворен отговор. След всеки въпрос ще се визуализира поле с текущият брой верни отговори.

Ето и по – важната част от реализирания код на неговия datasource:

public Question CurrentQuestion

{

get { return currentQuestion; }

set

{

this.currentQuestion = value;

UiQCounter = (currIndex+1) + "/" + Questions.Count; ;

if(this.currentQuestion.IsOpened == 1)

{

ClosedQ = "Hidden";

OpenQ = "Visible";

} else

{

ClosedQ = "Visible";

OpenQ = "Hidden";

}

OnPropertyChanged();

}

}

public int RightAnswers

{

get { return rightAnswers; }

set

{

this.rightAnswers = value;

OnPropertyChanged(nameof(RightAnswers));

}

}

public string CurrentOption

{

get { return currentOption; }

set

{

this.currentOption = value;

OnPropertyChanged();

}

}

public String Category

{

get { return \_category; }

set

{

\_category = value;

OnPropertyChanged(nameof(Category));

}

}

public TestViewModel(string category, User user)

{

Category = category;

User = user;

QuestionEntities ctx = new QuestionEntities();

questions = new List<Question>();

questions = (from a in ctx.Questions select a).ToList();

CurrentQuestion = questions.ElementAt(currIndex);

currIndex++;

RightAnswers = 0;

}

public void changeQuestion()

{

if (currentOption != null)

{

if(CurrentQuestion.IsOpened == 0)

{

currentOption = OpenQAnswer;

}

else

{

switch (currentOption)

{

case "1":

currentOption = CurrentQuestion.Answer1;

break;

case "2":

currentOption = CurrentQuestion.Answer2;

break;

case "3":

currentOption = CurrentQuestion.Answer3;

break;

case "4":

currentOption = CurrentQuestion.Answer4;

break;

}

}

if (currentOption.Equals(currentQuestion.RightAnswer))

{

RightAnswers = RightAnswers + 1;

}

}

if (currIndex < questions.Count)

{

int index = currIndex;

currIndex++;

CurrentQuestion = questions.ElementAt(index);

} else

{

changeViewToResult("resultView");

}

}

private void changeViewToResult(string view)

{

Messenger.Default.Send<SwitchViewMessage>(new SwitchViewMessage { ViewName = view, CorrectAnswers = RightAnswers});

}

public ICommand nextQuestion

{

get

{

return new QuestionCommand(changeQuestion);

}

}

public List<Question> Questions

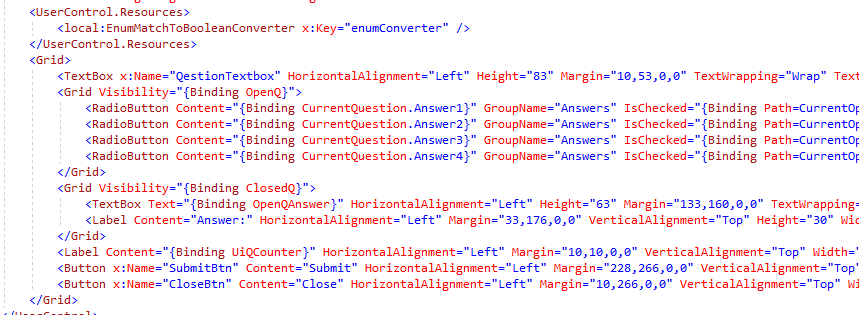
{

get { return questions; }

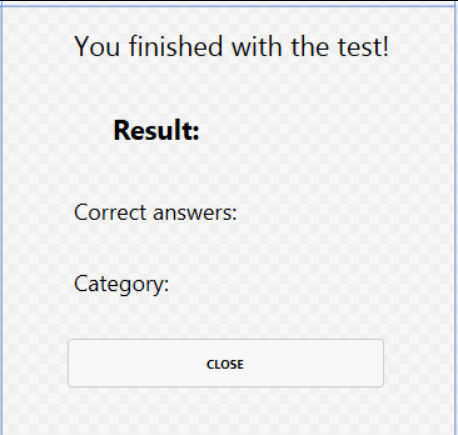
}

}

Част от xaml файла:



**ResultView**



Този User Control ще визуализира резултата след изпълнението на теста. Този Control се съдържа в TestMainWindow. Резултатът ще бъде записан в базата и ще съдържа датата на изпълнение, потребителя, броя на верните отговори и категорията на теста. След това потребителя ще може да види резултата си в историята на неговите тестове.

Ето и по – важната част от реализирания код на неговия datasource:

public ResultViewModel(int correctAnswers, string category, User user, int numberOfQuestions)

{

CorrectAnswers = correctAnswers;

Category = category;

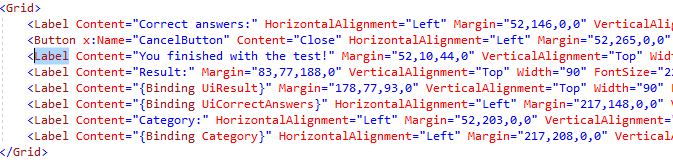
User = user;

UiCorrectAnswers = CorrectAnswers + " / " + numberOfQuestions;

UiResult = Result + "%";

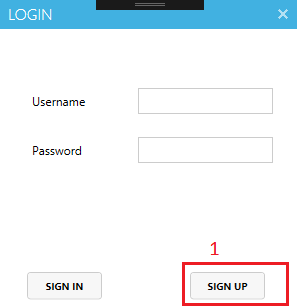
}

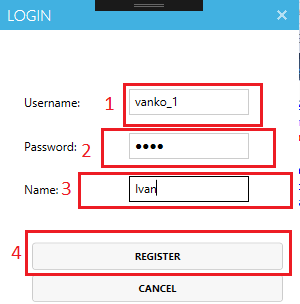
Част от xaml файла:



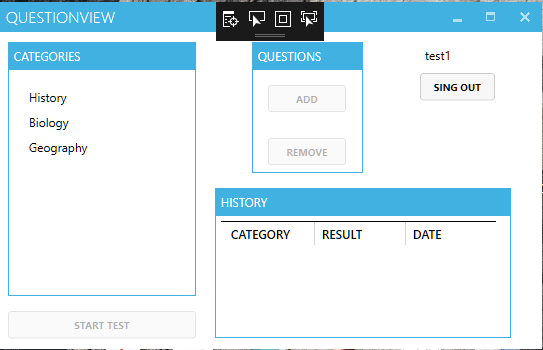
# Потребителско ръководство

## Регистрация на потребител

След стартиране на приложението, се зарежда прозореца за вписване на вече създадени потребители. Ние ще създадем нов потребител като натиснем бутона “Sign up”:

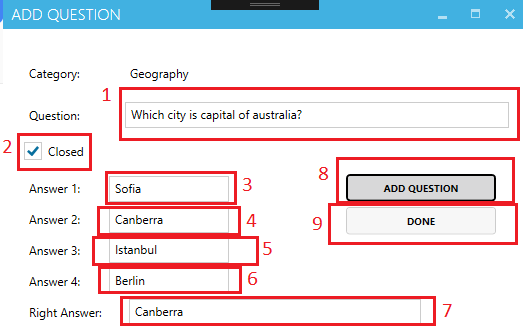
След това се зарежда прозореца за регистрация, там се попълват необходимите данни и се натиска бутона за регистрация:

## Главен прозорец



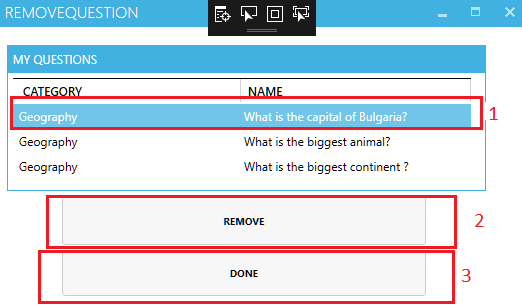
## Добавяне на въпрос

Може да добавяме въпроси като натиснем бутона “Add Question” от главния прозорец



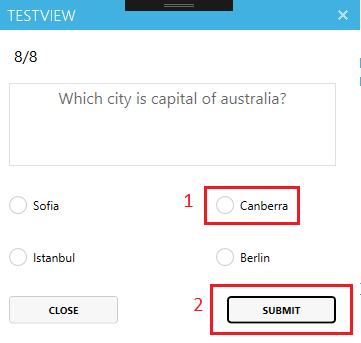
## Премахване на въпрос

Може да премахваме въпроси като натиснем бутона “Remove Question” от главния прозорец

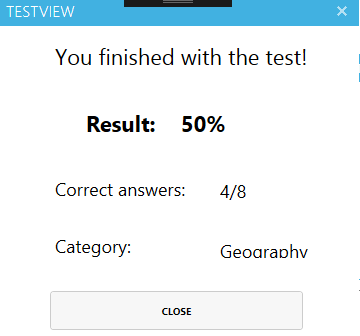


## Решаване на тест

Може да решаваме тест като изберем категория и натиснем бутона “Start Test” от главния прозорец. След въвеждане на отговор се натиска бутона “Submit”



След края на теста се визуализира резултата. С натискане на бутона “Close” се връщаме в главния прозорец



# Заключение

Приложението е годно за употреба от учители, ученици и дори просто за любителски нужди. Предоставя необходимите функционалности, за модифицирането на тестове според желанията и потребностите на потребителя.

# Използвана литература

<https://docs.microsoft.com/en-us/dotnet/framework/wpf/>

<https://msdn.microsoft.com/en-us/library/hh848246.aspx>

<https://www.objc.io/issues/13-architecture/mvvm/>