**Оглавление**

[1.Разработка программных модулей в среде MS Visual Studio 2019 1](#_Toc100693867)

[1.1 Изучение постановки задачи 1](#_Toc100693868)

[1.2. Описание области применения и пользователей приложения 1](#_Toc100693869)

[1.3. Проектирование интерфейса приложения 1](#_Toc100693870)

[1.4. Стандарты оформления кода языка C# 1](#_Toc100693871)

[1.5. Создание и использование стилей в технологии WPF 1](#_Toc100693872)

[1.6. Организация доступа к данным из приложения 1](#_Toc100693873)

[2. Разработка программных модулей в среде Python 1](#_Toc100693874)

[2.1. Установки и настройка среды PyCharm 1](#_Toc100693875)

[2.2. Особенности задания имен переменных и правила организации кода в Python 1](#_Toc100693876)

[2.3. Работа с циклами, строками и списками 1](#_Toc100693877)

[2.4. Именованные массивы – словари 1](#_Toc100693878)

[2.5. Картежи. Работа с файлами в Python 1](#_Toc100693879)

[3. Разработка мобильных приложений. 1](#_Toc100693880)

[3.1 Разработка дизайна мобильного приложения в среде Figma 1](#_Toc100693881)

[3.2 Разработка класса на языке Java 1](#_Toc100693882)

[3.3 Работа в среде Android 1](#_Toc100693883)

[4. Тестирование программных модулей 1](#_Toc100693884)

[4.1. План тестирования 1](#_Toc100693885)

[4.2. Сценарии тестирования с результатами 1](#_Toc100693886)

[Test case #1 1](#_Toc100693887)

[Test case #2 1](#_Toc100693888)

[Test case #3: 1](#_Toc100693889)

[Test case #4: 1](#_Toc100693890)

[Test case #5: 1](#_Toc100693891)

[4.3. Предложения по улучшению функциональности 1](#_Toc100693892)

[**5. Разработка технической документации** 1](#_Toc100693893)

[**5.1. Техническое задание** 1](#_Toc100693894)

[5.1.1 Введение 1](#_Toc100693895)

[5.1.2 Назначение разработки 1](#_Toc100693896)

[5.1.3 Требования к программе или программному изделию 1](#_Toc100693897)

[5.1.4 Требования к программной документации 1](#_Toc100693898)

[**5.2. Руководство программиста** 1](#_Toc100693899)

[5.2.1 Назначения разработки 1](#_Toc100693900)

[5.2.2 Характеристики программы 1](#_Toc100693901)

[5.2.3 Обращения к программе 1](#_Toc100693902)

[5.2.4 Входные и выходные данные 1](#_Toc100693903)

[5.2.5 Разработка базы данных 1](#_Toc100693904)

[5.2.6 Разработка программного модуля 1](#_Toc100693905)

[**5.3. Руководство пользователя** 1](#_Toc100693906)

[5.3.1 Назначение программы 1](#_Toc100693907)

[5.3.2 Условия выполнения программы 1](#_Toc100693908)

[5.3.3 Выполнение программы 1](#_Toc100693909)

[5.3.4 Выполнение программного модуля 1](#_Toc100693910)

[5.3.5 Сообщения пользователю 1](#_Toc100693911)

[Заключение 1](#_Toc100693912)

[Список литературы 1](#_Toc100693913)

[Приложения 1](#_Toc100693914)

[Приложение1 1](#_Toc100693915)

[Приложение2 1](#_Toc100693916)

# 1.Разработка программных модулей в среде MS Visual Studio 2019

## **1.1 Изучение постановки задачи**

Программный продукт «Victorina» – программа, созданная для работы с базой данных по предметной области.

Автоматизация продукта «Victorina» - процесс внедрения программно-аппаратных комплексов для облегчения и автоматизации.

Приложение должно выполнять следующие функции:

• Вход в программу при запуске

• Просмотр информации из базы данных

• Обеспечить идентицифированным пользователям функционал по их ролям

• Корректно выводить данные из базы данных.

• Добавление новой информации

• Удаление информации

• Изменение информации

• Сортировка

• Переходы по страницам

## **1.2. Описание области применения и пользователей приложения**

**Пользователи:**

1. **Студент**

Роль «Студент» в приложении имеет доступ ко следующим функциям: «Прохождение тестов», «Просмотр своих результатов», «Управление личной информацией»

1. **Админ**

Роль «Админ» в приложении имеет доступ ко следующим функциям: «Просмотр чужих результатов», «Удаление/редактирование/добавление новых пользователе».

Рис №1. Сase-диаграмма

## **1.3. Проектирование интерфейса приложения**

**1.3.1. Правила и принципы разработки интерфейса (теория)**

**Главная задача — ясность**

Ясность — это первая и самая главная задача любого интерфейса. Чтобы интерфейс эффективно помогал людям добиваться своих целей, он должен обладать следующими характеристиками.

* Во-первых, он должен быть узнаваемым, а его назначение — очевидным для пользователя.
* Во-вторых, люди должны понимать, с чем они взаимодействуют через интерфейс.
* Наконец, процесс взаимодействия с интерфейсом должен быть предсказуемым. В интерфейс можно внести некую загадочность или элементы игры, но вот путаницы быть не должно.

Ясность рождает в пользователях уверенность и готовность продолжать работу с интерфейсом. Сто понятных экранов лучше, чем один беспорядочный.

**Ключевое назначение интерфейсов — взаимодействие**

Интерфейсы существуют, чтобы люди могли взаимодействовать с нашим миром. Через интерфейс мы можем прояснить, проиллюстрировать, дать возможность, показать взаимосвязь, объединить людей или разделить, управлять ожиданиями и давать доступ к услугам. Процесс проектирования интерфейсов — это не Искусство.

*Интерфейсы — это не памятники самим себе. Интерфейсы выполняют конкретные задачи, их эффективность измерима.*

Однако, они могут выходить за рамки чисто практического применения. Лучшие интерфейсы — те, которые вдохновляют, пробуждают чувства, удивляют и усиливают наш опыт общения с миром.

**Удержать внимание любой ценой**

Мы живем в мире отвлекающих факторов. В наше время даже почитать спокойно не получится — кто-то или что-то всегда будет отвлекать нас, перетягивать наше внимание. Внимание — это большая ценность.

**1.3.2. Разработка макета приложения**

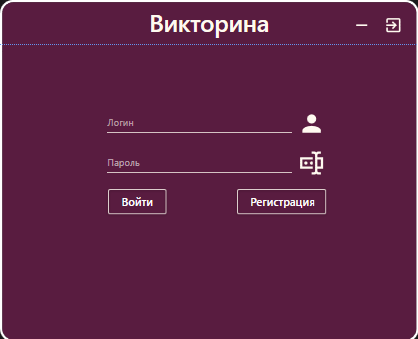


Рис №2 Страница авторизации



Рис №3 Страница пользовательского интерфейса



Рис №4 Страница админского интерфейса

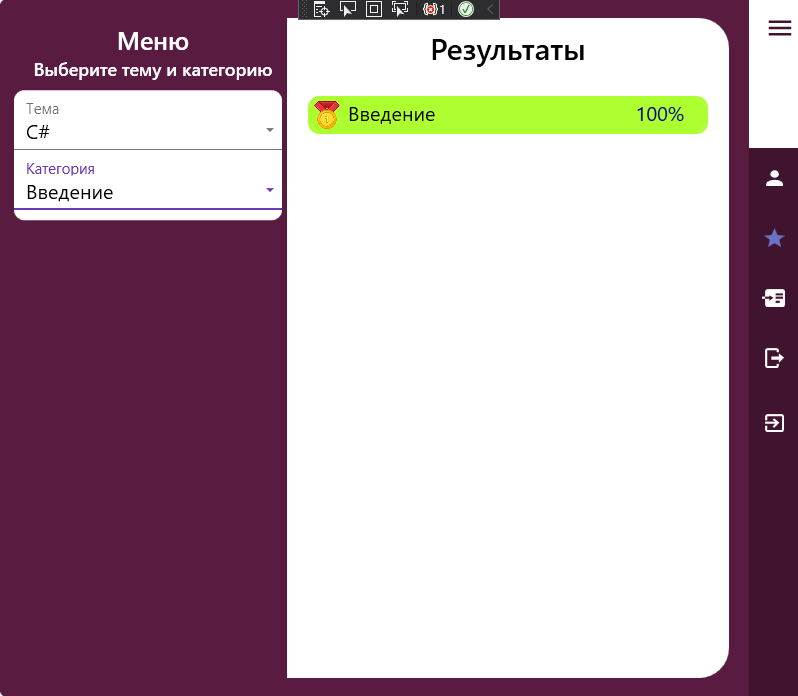


Рис №5 Страница результатов(Пользователь)

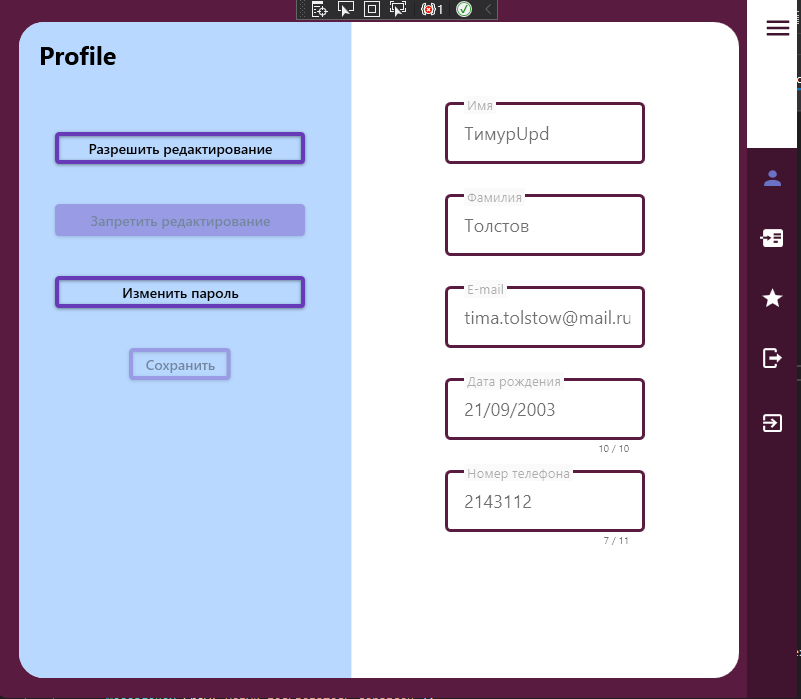


Рис №6 Страница профиля

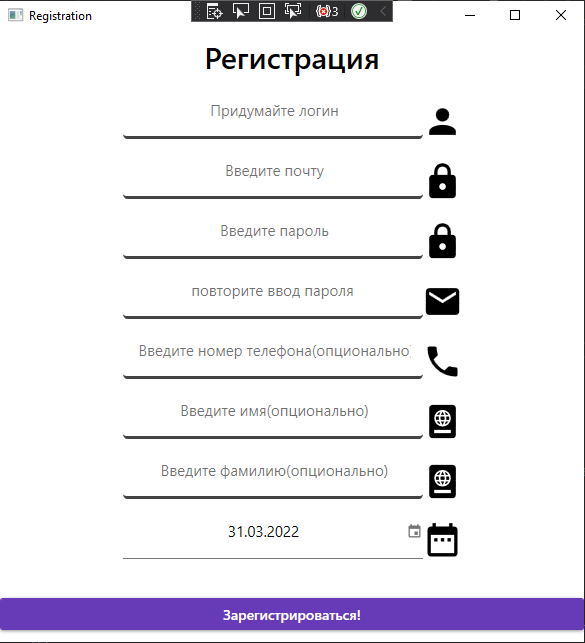


Рис №7 Страница регистрации

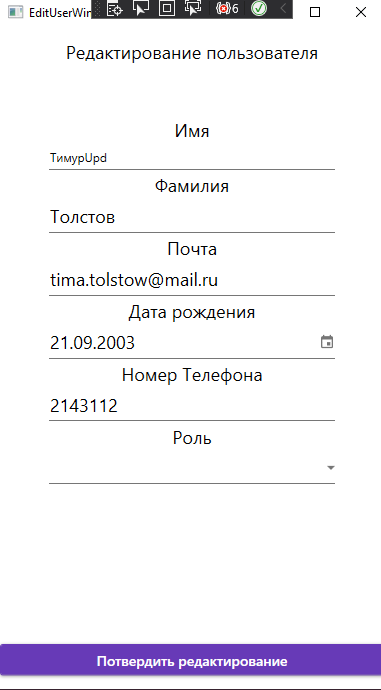


Рис №8 окно редактирование пользователя

**Окно добавления пользователя(Админ)**

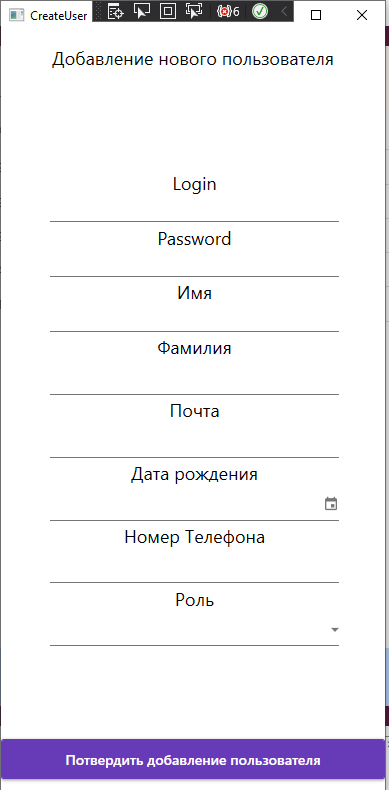


Рис №9 окно добавления пользователя

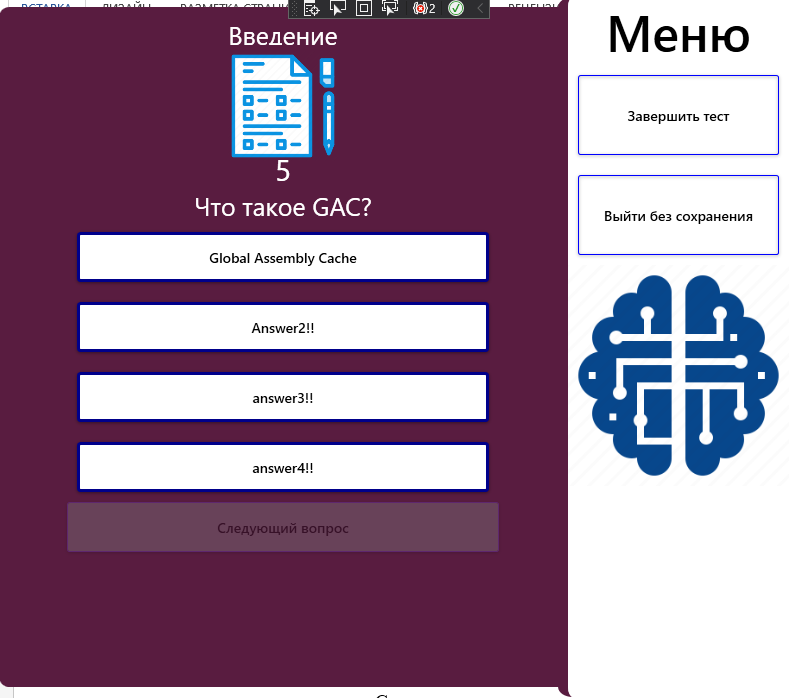


Рис №10 Страница прохождения теста

## **1.4. Стандарты оформления кода языка C#**

**1.4.1 Соглашения об именовании**

Camel case (иногда стилизованный под camelCase или camelCase, также известный как camel caps или более формально как медиальные столицы) - это практика написания фраз без пробелов или знаков препинания, указывающая на разделение слов с одной заглавной буквой и первое слово, начинающееся с любого случая. Распространенные примеры включают "iPhone" и "eBay". Он также иногда используется в онлайн-именах пользователей, таких как "JohnSmith", и для того, чтобы сделать многословные доменные имена более разборчивыми, например, при продвижении "EasyWidgetCompany.com.

Camel case часто используется в качестве соглашения об именах в компьютерном программировании, но является неоднозначным определением из-за необязательной капитализации первой буквы. Некоторые стили программирования предпочитают camel case с заглавной буквой, другие нет. Для ясности в этой статье называются две альтернативы верхний регистр верблюда (начальная заглавная буква, также известная как Начальные заглавные буквы, Начальные заглавные буквы, InitCaps или Pascal case) и нижний регистр верблюда (начальная строчная буква, также известная как падеж верблюда). Некоторые люди и организации, в частности Microsoft, используют этот термин camel case только для нижнего camel case, обозначая Pascal case для верхнего camel case.

**1.4.2. Таблица идентификаторов программных объектов**

|  |  |  |
| --- | --- | --- |
| Название обьекта | Тип обьекта | Назначение |
| Pages | | |
| AdminInterface | Страница | Интерфейс админа |
| AuthorizationPage | Страница | страница для авторизации |
| CategoryWnd | Окно | Окно категории |
| CreateNewUserWnd | Страница | Окно создание нового пользователя(админ) |
| EditUserWnd | Окно | Окно редактирования пользователя |
| ProfileUser | Страница | Страница профиля пользователя |
| RegistrationWnd | Окно | Окно регистрации |
| ResultPage | Страница | Страница результатов |
| TestForm | Страница | Страница прохождения тестов |
| UserInterface | Страница | Страница пользовательского интефейса |
| Services | | |
| СategoryServices | Класс | Запросы для коллекци категории |
| QuestionsServices | Класс | Запросы для коллекции вопросов |
| ResultsServices | Класс | Запросы для коллекции резульататов |
| TemaServices | Класс | Запросы для коллекции тем |
| TestItemServices | Класс | Запросы для коллекии тестов |
| UserService | Класс | Запросы для коллекции юзеров |

## **1.5. Создание и использование стилей в технологии WPF**

**1.5.1. Понятие и описание стиля**

Стили позволяют определить набор некоторых свойств и их значений, которые потом могут применяться к элементам в xaml. Стили хранятся в ресурсах и отделяют значения свойств элементов от пользовательского интерфейса. Также стили могут задавать некоторые аспекты поведения элементов с помощью триггеров. Аналогом стилей могут служить каскадные таблицы стилей (CSS), которые применяются в коде html на веб-страницах.

**1.5.2. Разработка стилей для элементов управления**

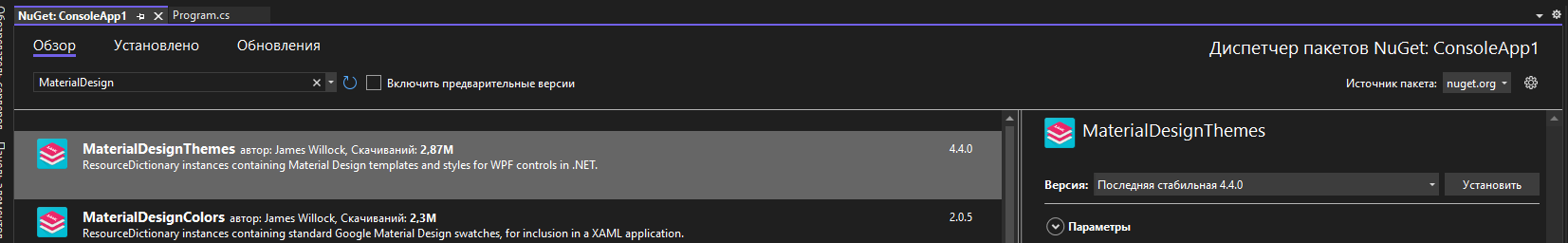


**1.5.3. Применение стилей для оформления интерфейса приложения**

Стили применяются автоматически при создании объектов в коде xaml. Вид и цвет объектов задается в app.xaml для всех объектов используя nu-get пакет MaterilDesign.

Подключение библиотеки MaterialdDesign

* 1. пкм, по названию проекта в обозревателе решений
  2. «Управление Nu-get пакетами»
  3. В поиске вводим MaterialDesign и справа нажимаем установить



* 1. Установка завершена

**1.5.4. Использование страниц Page в приложении**

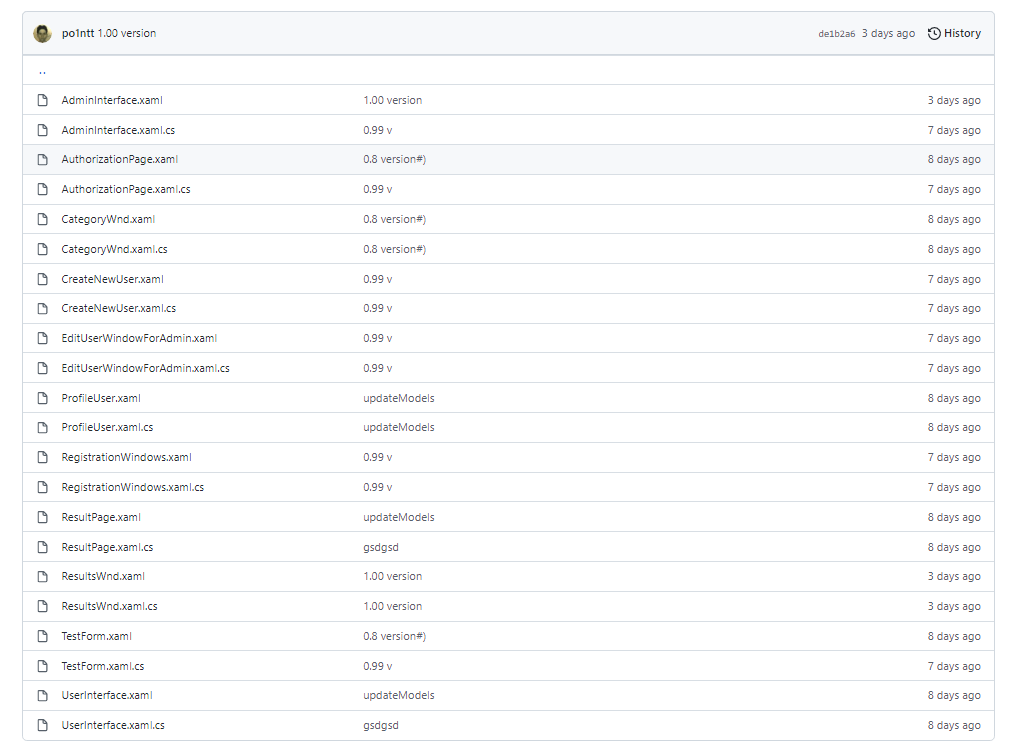


Рис №11. Папка Pages(В проекте)

**Класс навигации:**

class Frame

{

public static System.Windows.Controls.Frame FrameOBJ { get; set; }

}

**MainPage.cs:**

public MainWindow()

{

InitializeComponent();

ObjClas.Frame.FrameOBJ = MainFrame;

ObjClas.Frame.FrameOBJ.Navigate(new AuthorizationPage());

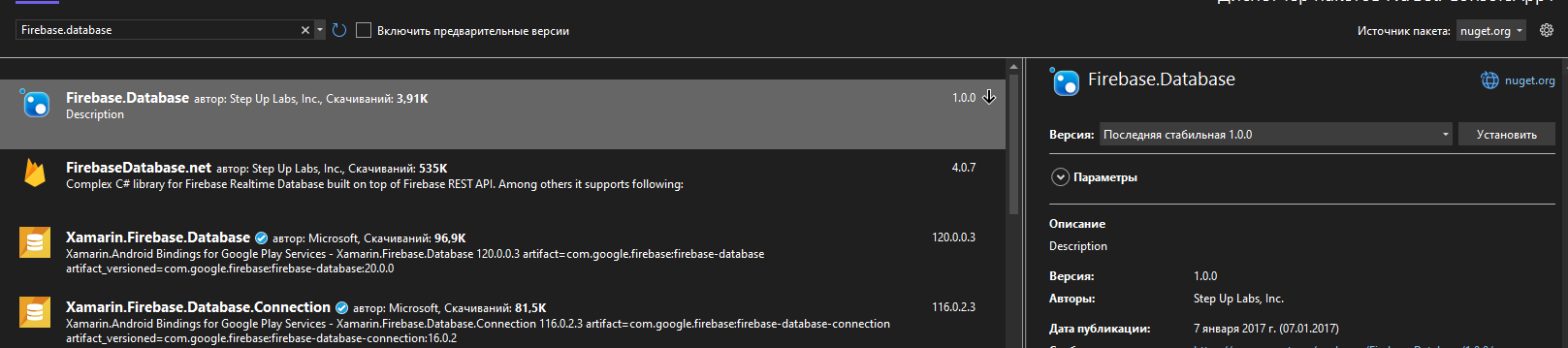
}

## **1.6. Организация доступа к данным из приложения**

**1.6.1. Использование Nu-get библиотеки FireBase.Database для связи с базой данных(теория)**

Для установки этой библиотеки нужно проделать следующие действия:

* 1. пкм, по названию проекта в обозревателе решений
  2. «Управление Nu-get пакетами»
  3. В поиске вводим firebase.database и справа нажимаем установить



* 1. Установка завершена.

Как обращаться к базе данных через эту библиотеку описано ниже.

**1.6.2. Создание приложения с применением Entity Framework** (подключение к БД, модули отображения данных, добавления, редактирования, поиска, фильтрации)

Разработка базы данных проходила в веб-сервисе google. Firebase.Console.

База данных имеет следующие коллекции:

* **Сategories:**

1. Cathegoryid;
2. СategoryName
3. TemaId
4. ImageUrl

* **ItemLearnCategory**

1. ImageUrlItem
2. NameItemLearn
3. Url\_toLearningSite
4. Id\_LearnCategory
5. Id\_ItemLearn

* **LearningCategory**

1. ImageLearnCat
2. NameLearnCategory
3. TemaId
4. IdLearnCat

* **Questions**

1. TextQuest
2. Id\_quest
3. Id\_test
4. Quest\_Answer1
5. Quest\_answer2
6. Quest\_answer3
7. QuestAnaswer4
8. Quest\_rightanswer

* **Results**

1. CategoryId
2. MedalImage
3. NameTestDone
4. ScorePercennt
5. TestID
6. User\_Login

* **Role**

1. Id\_role
2. NameRole

* **Tema**

1. NameTema
2. TemaId

* **Tests**

1. CategoryId
2. Description
3. Name
4. RightAnswerId
5. TestId

* **Users**

1. BirthDay
2. Email
3. Login
4. Name
5. Phone
6. Password
7. RoleId
8. SurName

В веб-сервисе база данных выглядит следующим образом:



Подключения в базе данных осуществляется следующим программным кодом:

Using Firebase.Database;

FirebaseClient client;

public СathegoryServices()

{

client = new FirebaseClient("https://victorinaproject-default-rtdb.firebaseio.com/");

}

Примеры программное реализации функции Добавления/Редактирование/Удаления/Просмотр на примере одной из модели данных(Questions)

**1. Удаление:**

public async Task<bool> DeleteQuest(int Id\_Quest)

{

var keytodelete = (await client.Child("Questions").OnceAsync<Questions>()).FirstOrDefault(a => a.Object.id\_quest == Id\_Quest);

await client.Child("Questions").Child(keytodelete.Key).DeleteAsync();

return true;

}

**2. Добавление**

public async Task<bool> AddQuest(string textquest, string questanswer1, string questanswer2, string questanswer3, string questanswer4, string questrightanswer, TestsModel tests)

{

var questions = await GetQuestionsAsync();

await client.Child("Questions").PostAsync(new Questions()

{

TextQuest = textquest,

id\_quest = questions.Count + 1,

quest\_answer1 = questanswer1,

quest\_answer2 = questanswer2,

quest\_answer3 = questanswer3,

quest\_answer4 = questanswer4,

quest\_rightanswer = questrightanswer,

id\_test = tests.TestId

});

return true;

}

**3. Редактирование**

public async Task<bool> UpdateQuestions(int id\_Quest,string textquest, string questanswer1, string questanswer2, string questanswer3, string questanswer4, string questrightanswer, TestsModel tests)

{

var keyquest = (await client.Child("Questions")

.OnceAsync<Questions>())

.FirstOrDefault

(a => a.Object.id\_quest == id\_Quest);

Questions quest = new Questions() {TextQuest = textquest, id\_quest = id\_Quest, id\_test = tests.TestId, quest\_answer1 = questanswer1, quest\_answer2 = questanswer2, quest\_answer3 = questanswer3, quest\_answer4 = questanswer4, quest\_rightanswer = questrightanswer };

await client.Child("Questions")

.Child(keyquest.Key)

.PutAsync(quest);

return true;

}

**4. Просмотр**

public async Task<List<Questions>> GetQuestionsAsync()

{

var questions = (await client.Child("Questions").OnceAsync<Questions>()).Select(f => new Questions

{

id\_quest= f.Object.id\_quest,

id\_test = f.Object.id\_test,

quest\_answer1 = f.Object.quest\_answer1,

quest\_answer2 = f.Object.quest\_answer2,

quest\_answer3 = f.Object.quest\_answer3,

quest\_answer4 = f.Object.quest\_answer4,

TextQuest = f.Object.TextQuest,

quest\_rightanswer =f.Object.quest\_rightanswer

}).ToList();

return questions;

}

public async Task<ObservableCollection<Questions>> GetQuestionsAsyncBYTest(int test\_id)

{

var QuestionsByTest = new ObservableCollection<Questions>();

var items = (await GetQuestionsAsync()).Where(p => p.id\_test == test\_id).ToList();

foreach (var item in items)

{

QuestionsByTest.Add(item);

}

return QuestionsByTest;

}

Работа с другими коллекциями данных происходит аналогичным образом.

# 2. Разработка программных модулей в среде Python

## **2.1. Установки и настройка среды PyCharm**

Для установки PyCharm сперва нужно установить интерпретатор Python.

После выполнения предыдущего пункта, перейдите на сайт JetBrains https://www.jetbrains.com/pycharm/. PyCharm – это проект JetBrains.

Нажмите «Download». Произойдет перенаправление на страницу загрузки PyCharm.

Дальше необходимо выбрать версию PyCharm: Community или Professional. Professional — это платная версия с полным набором функций. Она идеально подходит для профессиональной разработки. Community бесплатная. Ею можно пользоваться благодаря набору базовых возможностей. После нажатия на кнопку «Загрузить» скачивание начнется автоматически.

Теперь необходимо запустить установку. Нажмите «Next».

Оставьте папку установки по умолчанию. Это папка C:\Program files(x86) \JetBrains\PyCharm Community Edition 2019.2. Нажмите «Next».

Поставьте галочки напротив обоих пунктов, если нужен ярлык для рабочего стола. Нажмите «Next».

Далее установщик попросит указать имя для отображения в стартовом меню. По умолчанию это JetBrains. Нажмите «Install».

Установка начнется автоматически. Дождитесь завершения.

После завершения установки поставьте галочку напротив «Run PyCharm Community Edition» для запуска, а затем «Finish».

## **2.2. Особенности задания имен переменных и правила организации кода в Python**

PEP8 — это стилевые правила для кода на языке Python.

* используйте 4 пробела для отступа и не используйте Tab, не смешивайте их.
* максимальная длина строки 79 символов; для разрыва строки используйте отступы или “бэкслеш”
* функции верхнего уровня и определения классов отделяйте двумя пустыми строками
* определения методов внутри класса отделяйте одной пустой строкой
* дополнительные пустые строки используйте для логической группировки методов
* кодировка файлов должна быть Latin-1, а в Питоне 3 и выше - utf-
* каждый импортируемый модуль с новой строки
* порядок импортов: стандартные модули, third-party модули, локальные модули приложения
* пустая строка между каждой группой импортов
* избегайте лишних пробелов внутри скобок; перед запятыми, точкой с запятой и двоеточиями
* избегайте лишних пробелов перед скобками с аргументами функций и скобками с индексами
* избегайте больше, чем одного пробела между операторами (= - + отд.)
* используйте одиночный пробел для выделения операторов
* избегайте пробелов вокруг ‘=’, который используется для параметров по-умолчанию и keyword аргументов
* обновляйте комментарии вместе с кодом
* пишите комментарии по-английски
* избегайте имён из одиночных l, O, I
* для модулей и пакетов используйте короткие все строчные имена
* для классов используйте Pascal Case стандарт
* для функций используйте слова\_с\_подчёркиваниями (или смешанныйРегистр если так получилось исторически)
* для методов используйте слова\_с\_подчёркиваниями имена
* для внутренних методов и переменных \_подчёркивание\_перед\_словом
* для проверки на None используйте is или is not, не используйте операторы сравнения
* используйте исключения, основанные на классах – наследованные от Exception
* используйте методы класса string вместо строковых функций
* используйте ‘‘. startswith () и ‘‘. endswith () вместо вырезки из строк для проверки префиксов и суффиксов
* используйте isinstance () для проверки типа
* не сравнивайте булевы переменные с True и False

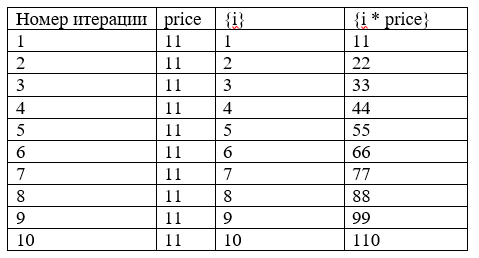
## **2.3. Работа с циклами, строками и списками**

**2.3.1 Выполнений задач на ветвление.**

Код:

price = float(input('Введите цену конфет: '))  
for i in range(1,11):  
 print(f'Цена за {i}кг конфет {i \* price}')

Результат:



**2.3.2 Отладка программного кода встроенными методами IDE (теория)**

**PyCharm и отладка**

В то время как многие из нас в первую очередь боятся писать коды, некоторые другие находят отладку более сложной задачей. В Python это требует больше времени, поскольку вы не знаете, в чем ошибка. К счастью, с PyCharm и его значительными функциями отладки пользователи могут получить уникальный опыт отладки при запуске своих скриптов Python.

**Точки останова и как их разместить**

Точки останова - это маркеры, которые помогают вам приостановить выполнение вашей программы в любой конкретной точке, чтобы вы могли изучить ошибки и поведение этой конкретной строки. После отметки точка останова останется в вашем коде, если вы явно не удалите ее. Чтобы разместить их, щелкните левый желоб рядом со строкой, на которой вы хотите приостановить приложение.

**Как начать отладку?**

Когда вы закончите добавлять точки останова, ваш код готов к отладке. Отладчик запустит всю программу, за исключением строк, которые вы явно не отметили. Чтобы начать сеанс отладчика, щелкните значок ‘зеленыйигратьикона' слева. Когда появится всплывающее меню, выберите параметр ‘ОтлаживатьРешатель. (Решатель, потому что это имя файла.)

Когда запускается отладчик, ваша программа начинает выполняться на вкладке консоли окна отладки. Поскольку программе требуются входные значения, экран отладчика попросит вас сделать это ниже:

В первой точке останова отладчик приостанавливает выполнение программы, выделяя строку синим цветом:

Чтобы возобновить отладку, щелкните «зеленый значок воспроизведения» в верхней части панели инструментов вкладки «Отладчик».

**Концепция встроенной отладки**

Встроенная отладка позволяет просматривать значение каждой переменной, чтобы сделать процесс отладки более простым и понятным. Эта функция по умолчанию присутствует в PyCharm.

**Наблюдение за вашими переменными в PyCharm**

Для более глубокого понимания вашего кода PyCharm также позволяет вам наблюдать за переменными. Нажмите кнопку в верхней части панели инструментов вкладки «Переменные». Затем введите имя переменной, за которой вы хотите следить во время выполнения кода.

**2.3.3 Выполнение задач на строки**

Код:

str = input('Введите строку: ')  
str1 = ''  
if(len(str) > 10):  
 str1 += str[-1]  
 str1 += str[0]  
 str1 += str[-3]  
 str1 += str[2]  
 print(str1)  
else:  
 print('Длина ст роки меньше 10')

Результат:

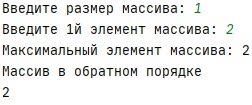


**2.3.4 Выполнение задач на списки**

Код:

mas = []  
n = int(input('Введите размер массива: '))  
for i in range(n):  
 mas.append(int(input(f'Введите {i + 1}й элемент массива: ')))  
print(f'Максимальный элемент массива: {max(mas)}')  
print('Массив в обратном порядке')  
print(\*reversed(mas))

Результат:



## **2.4. Именованные массивы – словари**

**2.4.1 Понятие и назначение словарей**

Словарь — неупорядоченная структура данных, которая позволяет хранить пары «ключ — значение».

Вот пример словаря на Python:

dictionary = {'персона': 'человек',

'марафон': 'гонка бегунов длиной около 26 миль',

'противостоять': 'оставаться сильным, несмотря на давление',

'бежать': 'двигаться со скоростью'}

Данный словарь использует строки в качестве ключей, однако ключом может являться в принципе любой неизменяемый тип данных. Значением же конкретного ключа может быть что угодно. Вот ещё один пример словаря, где ключами являются числа, а значениями — строки:

gender\_dict = {0: 'муж',

1: 'жен'}

Важное уточнение: если вы попробуете использовать изменяемый тип данных в качестве ключа, то получите ошибку:

dictionary = {(1, 2.0): 'кортежи могут быть ключами',

1: 'целые числа могут быть ключами',

'бежать': 'строки тоже',

['носок', 1, 2.0]: 'а списки не могут'}

Прим. перев. На самом деле проблема не с изменяемыми, а с некэшируемыми типами данных, но обычно это одно и то же.

**2.4.2 Использование методов**

#Создаем словарь  
company = {'Ит-отдел': 12, 'Бухгалтерия': 5, 'Менеджеры': 20, 'Директора': 4, 'Уборщики': 10}  
print("-------------------------------------------")  
  
#Выводим число сотрудников по названию подразделения  
print("Выводим число сотрудников по названию подразделения")  
key = input('Введите название отдела: ')  
if key in company.keys():  
 print(f'Кол-во сотрудников: {company[key]}')  
else:  
 print('Такого отдела нет!')  
print("-------------------------------------------")  
  
#Добавляем новые подразделения  
#Изменяем число  
company['Директора'] = 1  
company['Инженеры'] = 9  
company['Чертежники'] = 6  
  
#Выводим данные о подразделениях  
print("Выводим данные о подразделениях")  
print(company)  
print("-------------------------------------------")  
  
#Преобразовываем словарь в список  
print("Преобразовываем словарь в список")  
company\_list = list(company.items())  
for M in company\_list:  
 print(\*M)

**2.4.2 Разработка программы по обработке коллекций**

Необходимо написать программу, которая будет хранить информацию о товарах на складе.

Ключом словаря должен являться id записи.

Реализуйте следующий функционал:

• Добавление новой записи (id должно быть уникальным).

• Удаление записи по её id (учитывать, что такой номер может отсутствовать в словаре).

• Вывести все записи о товаре на указанном складе.

• Вывести всю информацию о товаре с ценой меньше указанной.

• Вывести всю информацию о товаре с указанной ед. изм.

• Вывести среднюю цену товары по складам в формате: «название склада: сред. Цена»

• Вывести всю информацию об указанном товаре.

• Вывести все данные записи и сумму по каждой записи. Сумма = Цена\*кол-во

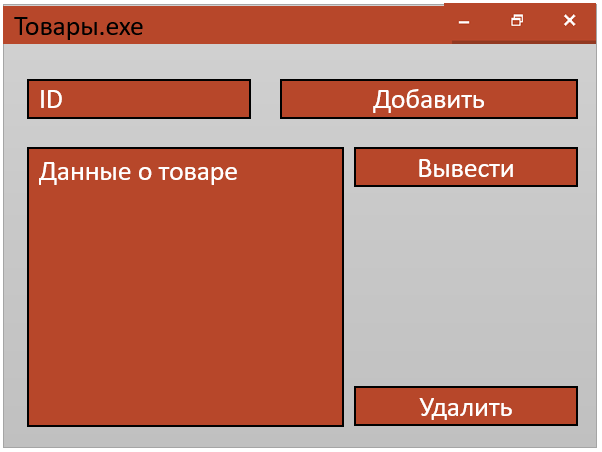
• Любая ваша функция.

При реализации данного функционала необходимо реализовать диалог с пользователем. (например, каждой функции можно присвоить номер, когда пользователь его введет, функция будет выполнена).

Код: [См.Приложение1](#_Приложение1)

**2.4.3 Разработка интерфейса программы**

По указанной предметной области был разработан интерфейс приложения, позволяющего делать совершать учет склада. Рисунок 46 демонстрирует возможный интерфейс приложения, а именно страницу добавления товара на определенный склад. Слева можно видеть меню приложения, а справа рабочую область, которая изменяется в зависимости от выбора пункта меню. На данном рисунке изображена форма добавления, на которой пользователю будет нужно заполнить поля, назначение которых указано в блоке слева.



## **2.5. Картежи. Работа с файлами в Python**

**2.5.1 Понятие картежей и их назначение**

Отличие кортежа от списка заключается в том, что кортеж защищен от изменений, как намеренных, так и случайных.

**Меньший размер. К примеру:**

>>> a = (1, 2, 3, 4, 5, 6)

>>> b = [1, 2, 3, 4, 5, 6]

>>> a.\_\_sizeof\_\_()

36

>>> b.\_\_sizeof\_\_()

44

**Возможность использовать кортежи в качестве ключей словаря:**

>>> d = {(1, 1, 1) : 1}

>>> d

{(1, 1, 1): 1}

>>> d = {[1, 1, 1] : 1}

Traceback (most recent call last):

File "", line 1, in

d = {[1, 1, 1] : 1}

TypeError: unhashable type: 'list'

**Как работать с кортежами?**

Работа с кортежами не сильно отличается от работы со списками.

Создаем пустой кортеж:

>>>

>>> a = tuple() # С помощью встроенной функции tuple()

>>> a

()

>>> a = () # С помощью литерала кортежа

>>> a

()

>>>

Создаем кортеж из одного элемента:

>>>

>>> a = ('s')

>>> a

's'

Получилась строка. Получить кортеж можно следующим образом:

>>>

>>> a = ('s', )

>>> a

('s',)

Все дело - в запятой. Сами по себе скобки ничего не значат, точнее, значат то, что внутри них находится одна инструкция, которая может быть отделена пробелами, переносом строк и т.д. Также, кортеж можно создать и так:

>>> a = 's',

>>> a

('s',)

Создать кортеж из итерируемого объекта можно с помощью все той же пресловутой функции tuple()

>>> a = tuple('hello, world!')

>>> a

('h', 'e', 'l', 'l', 'o', ',', ' ', 'w', 'o', 'r', 'l', 'd', '!')

**Операции с кортежами**

Все операции над списками, не изменяющие список (сложение, умножение на число, методы index () и count () и некоторые другие операции). Можно также по-разному менять элементы местами и так далее.

**2.5.2 Основы работы с внешними файлами в Python**

CSV (comma-separated value) — это формат представления табличных данных (например, это могут быть данные из таблицы или данные из БД).

В этом формате каждая строка файла — это строка таблицы. Несмотря на название формата, разделителем может быть не только запятая.

И хотя у форматов с другим разделителем может быть и собственное название, например, TSV (tab, separated, values), тем не менее, под форматом CSV понимают, как правило, любые разделители.

Пример файла в формате CSV (sw\_data.csv):

hostname, vendor, model, location

sw1, Cisco,3750, London

sw2, Cisco,3850, Liverpool

sw3, Cisco,3650, Liverpool

sw4, Cisco,3650, London

В стандартной библиотеке Python есть модуль csv, который позволяет работать с файлами в CSV формате.

**Чтение**

Пример чтения файла в формате CSV (файл csv\_read.py):

Вывод будет таким:

$ python csv\_read.py

['hostname', 'vendor', 'model', 'location']

['sw1', 'Cisco', '3750', 'London']

['sw2', 'Cisco', '3850', 'Liverpool']

['sw3', 'Cisco', '3650', 'Liverpool']

['sw4', 'Cisco', '3650', 'London']

В первом списке находятся названия столбцов, а в остальных соответствующие значения.

Обратите внимание, что сам csv.reader возвращает итератор:

In [1]: import csv

In [2]: with open('sw\_data.csv') as f:

...: reader = csv.reader(f)

...: print(reader)

...:

<\_csv.reader object at 0x10385b050>

При необходимости его можно превратить в список таким образом:

In [3]: with open('sw\_data.csv') as f:

...: reader = csv.reader(f)

...: print(list(reader))

...:

[['hostname', 'vendor', 'model', 'location'], ['sw1', 'Cisco', '3750', 'London'], ['sw2', 'Cisco', '3850', 'Liverpool'], ['sw3', 'Cisco', '3650', 'Liverpool'], ['sw4', 'Cisco', '3650', 'London']]

Чаще всего заголовки столбцов удобней получить отдельным объектом. Это можно сделать таким образом (файл csv\_read\_headers.py):

Иногда в результате обработки гораздо удобней получить словари, в которых ключи — это названия столбцов, а значения - значения столбцов.

Для этого в модуле есть DictReader (файл csv\_read\_dict.py):

Вывод будет таким:

$ python csv\_read\_dict.py

OrderedDict([('hostname', 'sw1'), ('vendor', 'Cisco'), ('model', '3750'), ('location', 'London')])

sw1 3750

OrderedDict([('hostname', 'sw2'), ('vendor', 'Cisco'), ('model', '3850'), ('location', 'Liverpool')])

sw2 3850

OrderedDict([('hostname', 'sw3'), ('vendor', 'Cisco'), ('model', '3650'), ('location', 'Liverpool')])

sw3 3650

OrderedDict([('hostname', 'sw4'), ('vendor', 'Cisco'), ('model', '3650'), ('location', 'London')])

sw4 3650

DictReader создает не стандартные словари Python, а упорядоченные словари. За счет этого порядок элементов соответствует порядку столбцов в CSV-файле.

Примечание

До Python 3.6 возвращались обычные словари, а не упорядоченные.

В остальном с упорядоченными словарями можно работать, используя те же методы, что и в обычных словарях.

Запись

Аналогичным образом с помощью модуля csv можно и записать файл в формате CSV (файл csv\_write.py):

В примере выше строки из списка сначала записываются в файл, а затем содержимое файла выводится на стандартный поток вывода.

Вывод будет таким:

$ python csv\_write.py

hostname,vendor,model,location

sw1,Cisco,3750,"London, Best str"

sw2,Cisco,3850,"Liverpool, Better str"

sw3,Cisco,3650,"Liverpool, Better str"

sw4,Cisco,3650,"London, Best str"

Обратите внимание на интересную особенность: строки в последнем столбце взяты в кавычки, а остальные значения - нет.

Так получилось из-за того, что во всех строках последнего столбца есть запятая. И кавычки указывают на то, что именно является целой строкой. Когда запятая находится в кавычках, модуль csv не воспринимает её как разделитель.

Иногда лучше, чтобы все строки были в кавычках. Конечно, в данном случае достаточно простой пример, но когда в строках больше значений, то кавычки позволяют указать, где начинается и заканчивается значение.

Модуль csv позволяет управлять этим. Для того, чтобы все строки записывались в CSV-файл с кавычками, надо изменить скрипт таким образом (файл csv\_write\_quoting.py):

Теперь вывод будет таким:

$ python csv\_write\_quoting.py

"hostname","vendor","model","location"

"sw1","Cisco","3750","London, Best str"

"sw2","Cisco","3850","Liverpool, Better str"

"sw3","Cisco","3650","Liverpool, Better str"

"sw4","Cisco","3650","London, Best str"

Теперь все значения с кавычками. И поскольку номер модели задан как строка в изначальном списке, тут он тоже в кавычках.

Кроме метода writerow, поддерживается метод writerows. Ему можно передать любой итерируемый объект.

Например, предыдущий пример можно записать таким образом (файл csv\_writerows.py):

DictWriter

С помощью DictWriter можно записать словари в формат CSV.

В целом DictWriter работает так же, как writer, но так как словари не упорядочены, надо указывать явно в каком порядке будут идти столбцы в файле. Для этого используется параметр fieldnames (файл csv\_write\_dict.py):

Указание разделителя

Иногда в качестве разделителя используются другие значения. В таком случае должна быть возможность подсказать модулю, какой именно разделитель использовать.

Например, если в файле используется разделитель; (файл sw\_data2.csv):

hostname; vendor; model; location

sw1; Cisco;3750; London

sw2; Cisco;3850; Liverpool

sw3; Cisco;3650; Liverpool

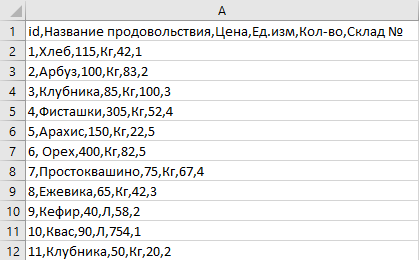
sw4; Cisco;3650; London

Достаточно просто указать, какой разделитель используется в reader (файл csv\_read\_delimiter.py)

**2.5.2 Добавление в программу функционала по работе с csv файлами**

Код: [См.Приложения2](#_Приложение2)

Файл csv:



# 3. Разработка мобильных приложений.

## **3.1 Разработка дизайна мобильного приложения в среде Figma**

## **3.2 Разработка класса на языке Java**

## **3.3 Работа в среде Android**

**3.3.1 Структура проекта Android**

**3.3.2 Создание проекта по заданию**

# 4. Тестирование программных модулей

## **4.1. План тестирования**

* Тестирование авторизации, и правильности переходов на страницу после авторизации (в зависимости от роли пользователя).
* Тестирование функционала «Тестирования» получение результатов тестирования
* Изменение личных данных самим пользователем на странице «Профиль»
* Авторизация с неверными данными.

## **4.2. Сценарии тестирования с результатами**

## **Test case #1**

|  |  |
| --- | --- |
| **Тестовый пример #** | 1 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Авторизация обычного пользователя |
| **Краткое изложение теста** | После ввода данных пользователь должен оказаться на странице пользовательского интерфейса |
| **Этапы теста** | 1. Запуск приложения 2. Ввод данных в textbox-ы(Пароль,Логин) 3. Потверждение данных |
| **Тестовые данные** | Login – TestLog, Password-TestPas |
| **Ожидаемый результат** | Пользователь успешно оказался на странице пользовательского интерфейса |
| **Фактический результат** | Пользователь успешно оказался на странице пользовательского интерфейса |
| **Статус** | Зачет |
| **Предварительное условие** | - |
| **Постусловие** | - |
| **Примечания/комментарии** | Тест пройден успешно |

## **Test case #2**

|  |  |
| --- | --- |
| **Тестовый пример #** | 2 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Авторизация пользователя с ролью администратора |
| **Краткое изложение теста** | После ввода данных пользователь должен оказаться на странице админского интерфейса. |
| **Этапы теста** | 1. Запуск приложения 2. Ввод данных для авторизации(Аккаунт админа) 3. Подтверждение данных |
| **Тестовые данные** | Login – LogAdmin, Password - Testgg |
| **Ожидаемый результат** | Пользователь успешно оказался на странице админского интерфейса |
| **Фактический результат** | Пользователь успешно оказался на странице админского интерфейса |
| **Статус** | Зачет |
| **Предварительное условие** | - |
| **Постусловие** | - |
| **Примечания/комментарии** | Тест пройден успешно |

## **Test case #3:**

|  |  |
| --- | --- |
| Тестовый пример **#** | 3 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Получение результатов после прохождения тестирования |
| **Краткое изложение теста** | После прохождения тестирования пользователю должна быть предоставлена информация о результатах прохождения(процент правильных ответов) |
| **Этапы теста** | 1. Запуск приложения 2. Авторизация 3. Выбор категории 4. Переход на определенный тест 5. Прохождение теста 6. Получение результата(в messagebox) |
| **Тестовые данные** | Login –TestLog, Password - TestPas |
| **Ожидаемый результат** | После прохождения теста, высвечивается всплывающее окно с результатом тестирования. |
| **Фактический результат** | После прохождения теста, высвечивается всплывающее окно с результатом тестирования. |
| **Статус** | Зачет |
| **Предварительное условие** | - |
| **Постусловие** | - |
| **Примечания/комментарии** | Тест пройден успешно. |

## **Test case #4:**

|  |  |
| --- | --- |
| Тестовый пример **#** | 4 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Личное изменение данных пользователем |
| **Краткое изложение теста** | На странице «Профиль» пользователь используя интерфейс может изменить личные данные введенные при регистрации. |
| **Этапы теста** | 1. Запуск приложения 2. Авторизация 3. Переход на страницу профиля 4. Клик на кнопку «Разрешить редактирование» 5. Изменить данные Email и Name 6. Нажать «сохранить» |
| **Тестовые данные** | Login –TestLog, Password – TestPas, новый email – [Shahanov@mail.ru](mailto:Shahanov@mail.ru), Name- Никита |
| **Ожидаемый результат** | Пользователь успешно изменил свои данные и высветился всплывающее окно с уведомление о изменение данных |
| **Фактический результат** | Пользователь успешно изменил свои данные и высветился всплывающее окно с уведомление о изменение данных |
| **Статус** | Зачет |
| **Предварительное условие** | - |
| **Постусловие** | - |
| **Примечания/комментарии** | Тест пройден успешно |

## **Test case #5:**

|  |  |
| --- | --- |
| **Test Case #** | 5 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Ввод неверных данных при авторизациия |
| **Краткое изложение теста** | После ввода неверных данных при авторизации пользователю должен высветится попап, сообщающий о вводе неверных данных |
| **Этапы теста** | 1. Запуск приложения 2. На окне авторизации ввести неверные данные 3. Подтвердить |
| **Тестовые данные** | Login – 12341 Password -153 |
| **Ожидаемый результат** | Пользователь не смог авторизоваться, пользователя высветился всплывающее окно, сообщающий о вводе неверных данных |
| **Фактический результат** | Пользователь не смог авторизоваться, пользователя высветился всплывающее окно, сообщающий о вводе неверных данных |
| **Статус** | Зачет |
| **Предварительное условие** | - |
| **Постусловие** | - |
| **Примечания/комментарии** | Тест пройден успешно. |

## **4.3. Предложения по улучшению функциональности**

1. Добавление функционала получения наград за завершения категорий на 100%(Достижения)

2. возможно отправки репортов/рекомендаций/комментариев пользователем

# 5. Разработка технической документации

## **5.1. Техническое задание**

## **5.1.1 Введение**

Платформа для прохождения тестирование «Victorina». Предназначена удобного прохождения тестов, на разных платформах desktop/mobile. Разработка проводилась в WPF(Windows Forms Presentation, система для построения клиентских приложений).

Цели приложения:

**обучающая:** расширение кругозора, активизация интеллектуальной деятельности; формирование внимательного отношения к дисциплинам.

**развивающая:** формирование умений обдумывать и принимать решения, развитие мышления, памяти, эрудиции.

**воспитательная:** развитие познавательного интереса, воспитание у обучающихся самостоятельности как черты личности, без которой невозможна деятельность современного специалиста.

## **5.1.2 Назначение разработки**

Разработка проводится на основании приказа о курсовом проектировании.

Программный продукт «Victorina» – программа, созданная для работы с базой данных по предметной области.

Автоматизация продукта «Victorina» - процесс внедрения программно-аппаратных комплексов для облегчения и автоматизации.

## **5.1.3 Требования к программе или программному изделию**

**Требования к составу выполняемых функций:**

* Вход в программу при запуске
* Просмотр информации из базы данных
* Обеспечить идентифицированным пользователям функционал по их ролям
* Корректно выводить данные из базы данных.
* Добавление новой информации
* Удаление информации
* Изменение информации
* Сортировка
* Переходы по страницам

**Требования к организации входных данных:**

Входные данные программы организованы в виде ввода в текстовые поля или выбором: Пароль, Логин, Email, Телефон, Фамилия, Имя, Категория, Тест, Тема, Ответ.

**Требования к организации выходных данных:**

Выходные данные организованы в виде таблиц (Список пользователей) и объектов (Категорий, тестов, тем, вопросов)

**Требования к надежности.**

Программа должна быть в достаточной степени надёжна от сбоев. На такой случай в программе предусмотрены следующие функции:

* Парольная защита. Для работы с программой нужно ввести логин и пароль.
* Маска ввода на затраты. Программа не дает возможности вписать различные символы, поставить две запятые, или написать больше двух цифр после запятой в поле затраты
* Сообщения пользователю
* Приведено описание сообщений пользователю и возможные действия пользователя в ответ на эти сообщения.

**Условия эксплуатации**

Программа не требует специального обслуживания. Для ознакомления с полным функционалом пользователь должен прочесть Руководство пользователя. Для работы с программой требуются хотя бы малейшие навыки работы похожими продуктами. Программа запускается с помощью исполняющего файла Transmash.exe.

Климатические условия эксплуатации:

Климатические условия эксплуатации, при которых должны обеспечиваться заданные характеристики, должны удовлетворять требованиям, предъявляемым к техническим средствам в части условий их эксплуатации.

**Требования к составу и параметрам технических средств**

Минимальные характеристики компьютера

|  |  |
| --- | --- |
| Процессор | AMD Athlon |
| Память | От 4 гб |
| Разрешение экрана | От 1280x1024 |
| Устройства ввода | Клавиатура, мышь |
| Дисковое пространство | От 1 Гб |
|  |  |

Оптимальный характеристики компьютера

|  |  |
| --- | --- |
| Процессор | Intel Core i5 3ГГц |
| Память | От 8 гб |
| Разрешение экрана | От 1920x1080 |
| Устройства ввода | Клавиатура, мышь |
| Дисковое пространство | От 1 Гб |

**Требования к информационной и программной совместимости**

Для корректной работы программы необходимо: ОС Windows 7-10;

**Специальные требования**

Программа должна обеспечивать взаимодействие с пользователем (клиентом) посредством графического пользовательского интерфейса.

## **5.1.4 Требования к программной документации**

В ходе разработки программы должны быть подготовлены следующие программные документы: текст программы, описание программы, программа и методика испытаний, руководство пользователя, руководство программиста, технико-экономическое обоснование.

## **5.2. Руководство программиста**

## **5.2.1 Назначения разработки**

Разработка проводится на основании приказа о курсовом проектировании.

Программный продукт «Victorina» – программа, созданная для работы с базой данных по предметной области.

Автоматизация продукта «Victorina» - процесс внедрения программно-аппаратных комплексов для облегчения и автоматизации.

## **5.2.2 Характеристики программы**

Программа должна обеспечивать следующие функции

* Вход в программу при запуске
* Просмотр информации из базы данных
* Обеспечить идентифицированным пользователям функционал по их ролям
* Корректно выводить данные из базы данных.
* Добавление новой информации
* Удаление информации
* Изменение информации
* Сортировка
* Переходы по страницам

## **5.2.3 Обращения к программе**

С# — объектно-ориентированный язык программирования. Разработан в 1998—2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык разработки приложений для платформы Microsoft .NET Framework. Впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270

С помощью конструктора Windows Presentation Foundation (WPF) – система для построения клиентских приложений Windows

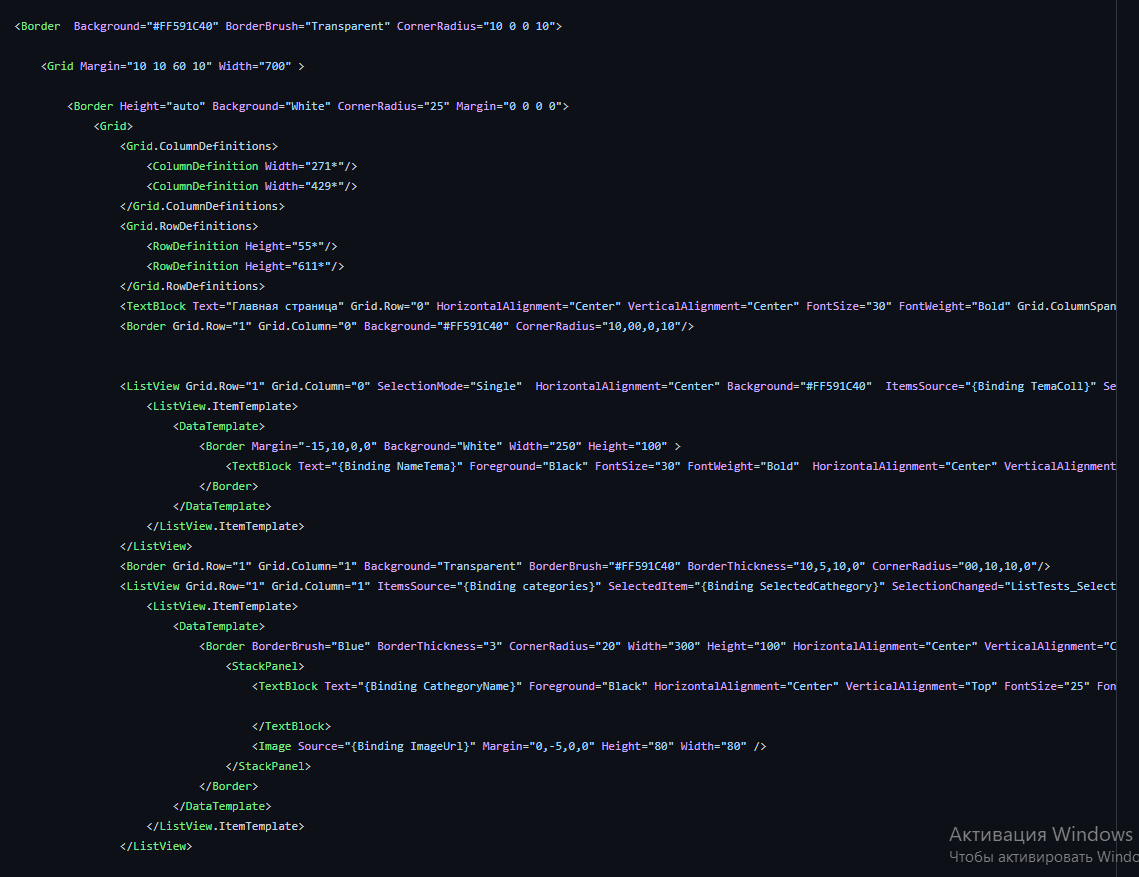


Рис. №1 Страница пользовательского интерфейса

## **5.2.4 Входные и выходные данные**

**Входные данные:**

|  |  |
| --- | --- |
| **Поля** | **Тип данных** |
| Пароль | Строковый |
| Логин | Строковый |
| Email | Строковый |
| Телефон | Строковый |
| Фамилия | Строковый |
| Имя | Строковый |

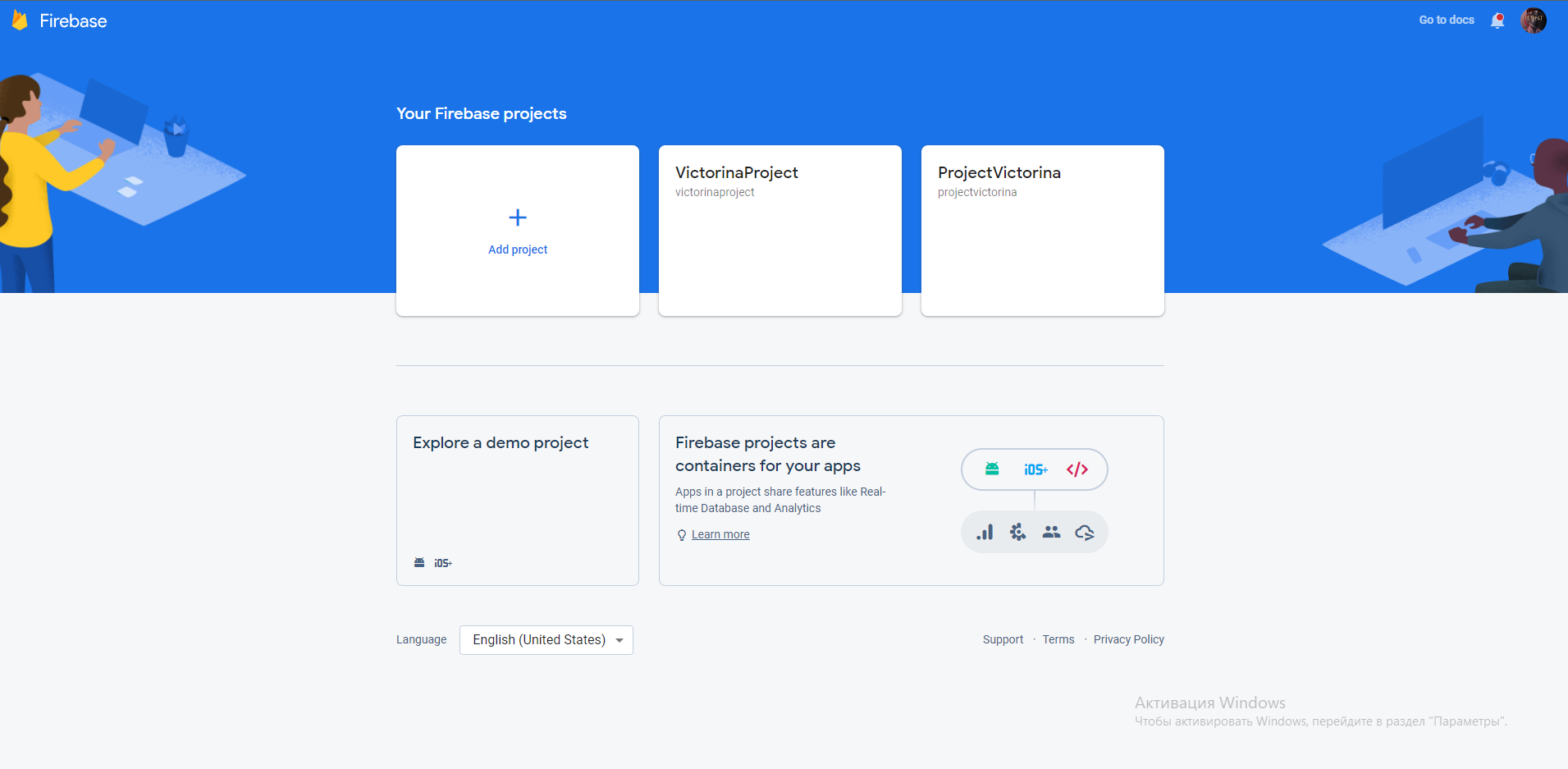
**Выходная информация**

Выходные данные организованы в виде таблиц (Список пользователей) и объектов (Категорий, тестов, тем, вопросов)

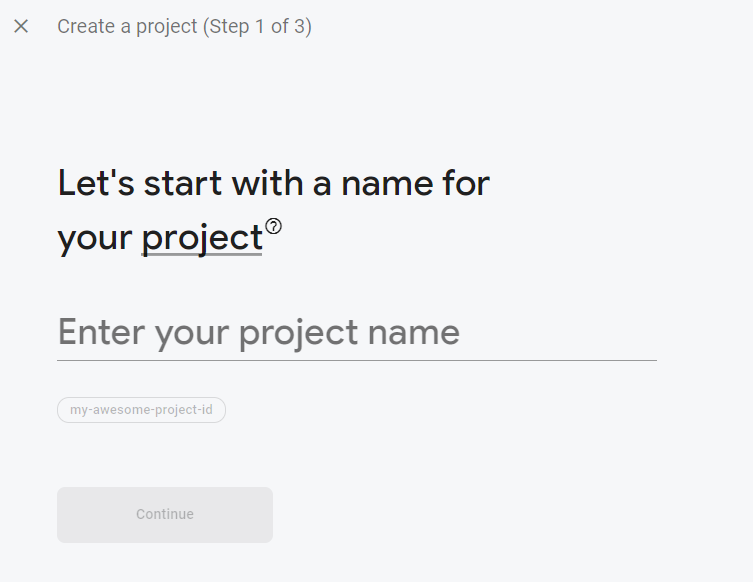
## **5.2.5 Разработка базы данных**

**Создание базы данных:**

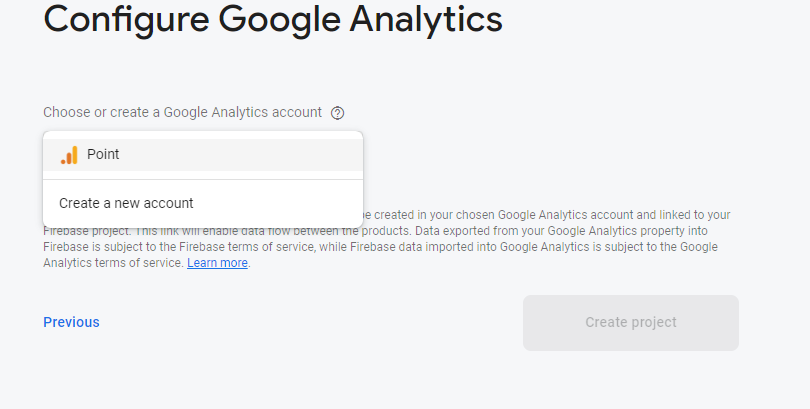
1. Переход на сайт <https://console.firebase.google.com/> и авторизация.



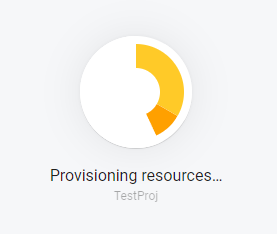
1. Нажимаем на кнопку Add Project. Появляется окно с вводом имени проекта вводим имя проекта жмем «Continue».



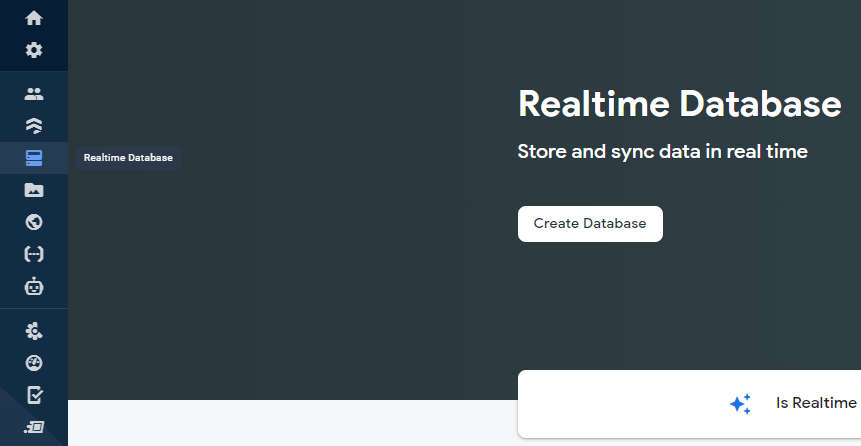
1. Жмем «Continue»



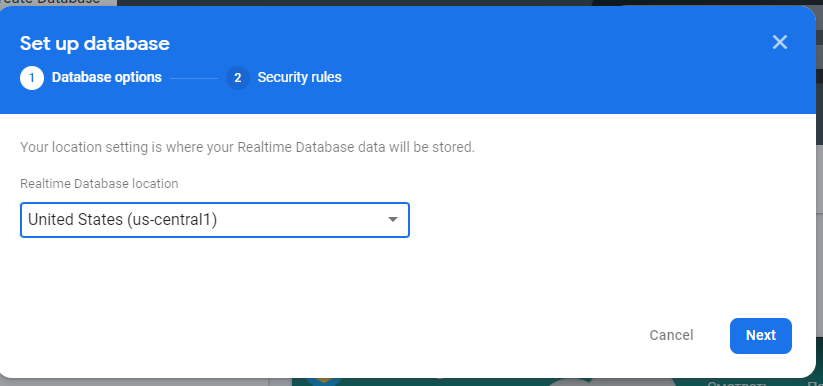
1. Выбираем аккаунт, на который создадим проект и жмем «Create project» и ждем загрузку.



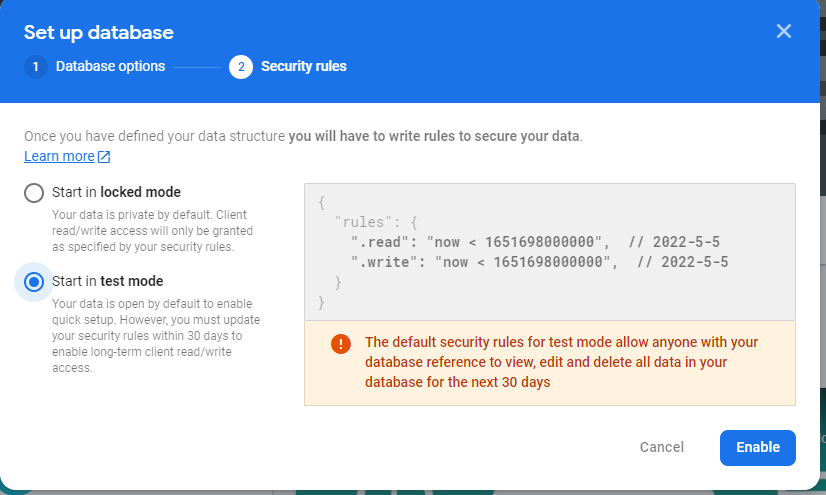
1. Заходим во вкладку RealTime DB и создаем базу данных(«Create DataBase»)



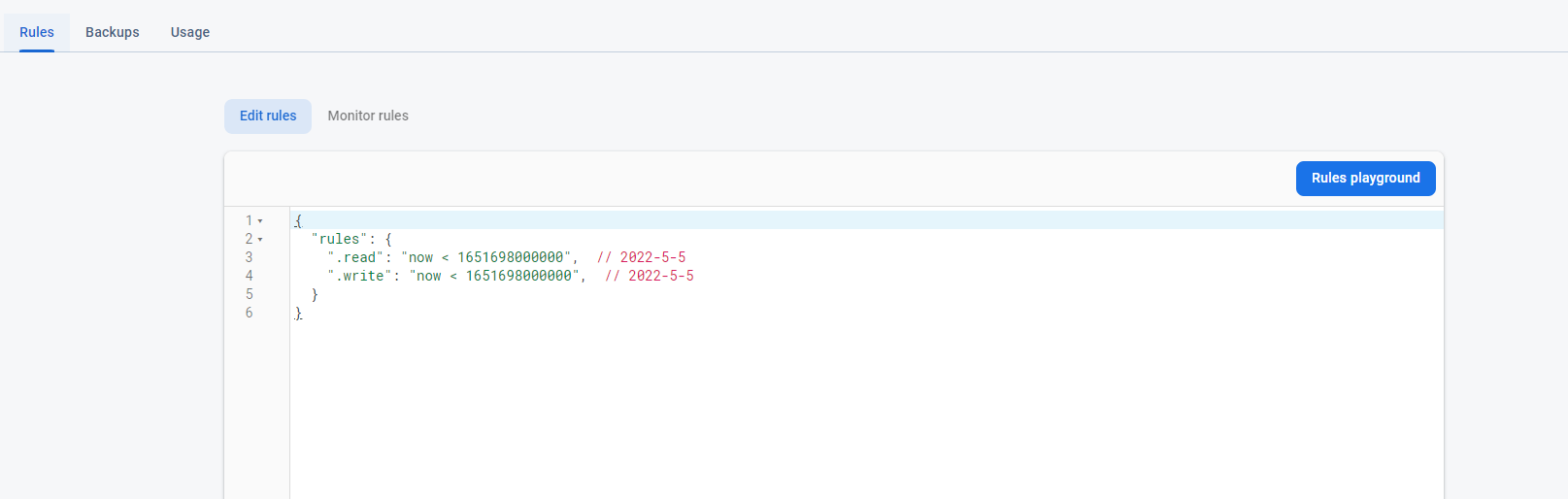
1. Выбираем страну и жмем Next.



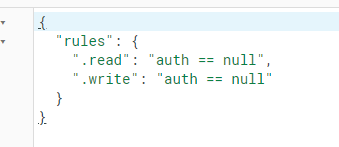
1. Ставим галочку на test-mode и жмем «Enable»



1. База данных успешно создана! Заходим во вкладку rules.



1. Изменяем правила Read и Write на «auth == null». Подтверждаем изменение. База данных готова к работе.



Разработка базы данных проходила в одном из веб-сервисом google. Firebase.Console.

База данных имеет следующие коллекции:

•Сategories:

1. Cathegoryid;

2.СategoryName

3. TemaId

4. ImageUrl

•ItemLearnCategory

1. ImageUrlItem

2. NameItemLearn

3. Url\_toLearningSite

4. Id\_LearnCategory

5. Id\_ItemLearn

•LearningCategory

1. ImageLearnCat

2. NameLearnCategory

3. TemaId

4. IdLearnCat

•Questions

1. TextQuest

2. Id\_quest

3. Id\_test

4. Quest\_Answer1

5. Quest\_answer2

6. Quest\_answer3

7. QuestAnaswer4

8. Quest\_rightanswer

•Results

1. CategoryId

2. MedalImage

3. NameTestDone

4. ScorePercennt

5. TestID

6. User\_Login

•Role

1. Id\_role

2. NameRole

• Tema

1. NameTema

2. TemaId

•Tests

1. CategoryId

2. Description

3. Name

4. RightAnswerId

5. TestId

•Users

1. BirthDay

2. Email

3. Login

4. Name

5. Phone

6. Password

7. RoleId

8. SurName

В веб-сервисе база данных выглядит следующим образом:



## **5.2.6 Разработка программного модуля**

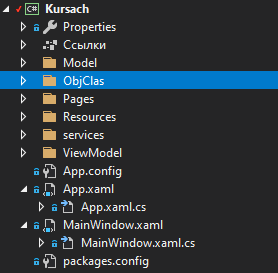
Программа была разработана на Visual Studio 2019

1. **Определение структуры проекта (Создание нужных папок, классов, страниц, моделей данных)**

Структура проекта основывается на паттерне MVVM(Models-Views-ViewsModels)

**MVVM** - это архитектурный шаблон программного обеспечения, который облегчает разделение разработки графического пользовательского интерфейса (представления) — будь то с помощью языка разметки или кода GUI — от разработки бизнес-логики или обратной связи. конечная логика (модель), чтобы представление не зависело от какой-либо конкретной платформы модели.

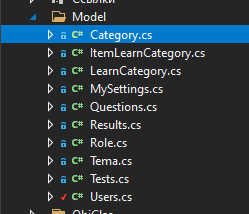
Структура проекта выглядит следующим образом:



1. Написание моделей данных для связи с базой данных.

Модели данных нужны для написания запросов к базе данных в данном случае firebase.

Директория с ними выглядит так:



Пример программного кода одной из них:

public class Category

{

public int CathegoryId { get; set; }

public string CathegoryName { get; set; }

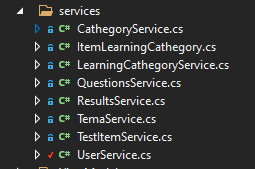
public string ImageUrl { get; set; }

public int TemaID { get; set; }

}

1. Написание запросов к коллекциям базы данных.

Запросы находятся в директории «Services». В ней хранятся Задачи(Task<T>), которые в свою очередь находятся в классах, данные задачи нужны для получения/внесения/удаления информации из базы данных.



1. Написание страниц(Views) и моделей к ним(ViewsModel)

Страницы в себе хранят Xaml и cs файл. В Xaml файлах пишут верстку самой страницы, а в cs – некоторый функционал.

Модели же хранят в себе свойства и некоторые команды данной страницы. Имеют интерфейс INotifyPropertyChanged.

**INotifyPropertyChanged** - используется для уведомления представления об изменениях свойств объекта. Реализация классом интерфейса предполагает генерацию события PropertyChanged каждый раз, когда значение свойства объекта изменяется.

**2.** **Подключение БД**

Подключения в базе данных осуществляется следующим программным кодом:

Using Firebase.Database;

FirebaseClient client;

public СathegoryServices()

{

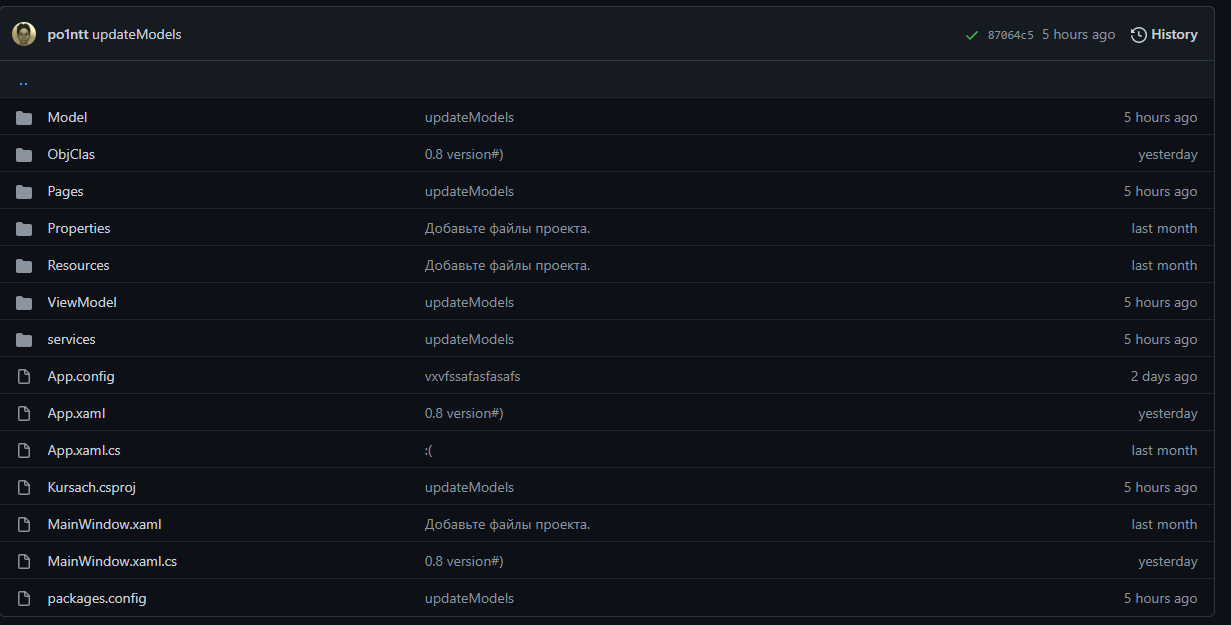
client = new FirebaseClient("https://victorinaproject-default-rtdb.firebaseio.com/");

}

* Using – ссылка на nu-get пакет.
* FirebaseClient – определение клиента firebase, и вставка ссылки на базу данных.

**3. Текст программы**

Текст программы можно найти на ресурсе GitHub - [**https://github.com/po1ntt/CourseWorkMain**](https://github.com/po1ntt/CourseWorkMain)

****

Описание страниц:

* **AdminIterfase**

Страница содержит в себе таблицу c выводом информации о пользователях. При правом клике по объекту в таблице, появляется контекстное меню с действиями– «Удаление», «Редактирование», «Добавление нового пользователя», «Просмотр результатов». **Удаление** – удаляет из базы данных выбранного пользователя. **Редактирование** – открывает всплывающее окно где можно изменять данные пользователя и сохранить изменения в базу данных. **Добавление нового пользователя** – открывает всплывающее окно, где можно добавить нового пользователя в базу данных.

**Просмотр результатов –** открывает всплывающее окно, где можно посмотреть результаты выбранного пользователя.

Классы относящиеся к этой страницы:

1. Viewmodels

AdminViewModel.cs – класс, имеющий интерфейс InotifyPropertyChanged. Содержит в себе коллекции и свойства.

Коллекции нужны для вывода информации в таблицу или list view, свойства, для получения некоторых данных при взаимодействии с объектами на странице

1. Services

UserService.cs – класс, который содержит в себе запросы к базе данных к коллекции «Users». На данной странице используется следующий запрос: SelectUsers (), выводящий список всех пользователей, существующих в базе данных.

* **AutorizationPage**

Страница в которой пользователей авторизовывается в приложении, путем ввода своего логина и пароля. Также при клике на кнопку «Регистрация» открывается окно регистрации. При авторизации осуществляется проверка корректности ввода данных.

Классы относящиеся к этой страницы:

1. Viewmodels

AuthorizationViewModel.cs – класс, имеющий интерфейс InotifyPropertyChanged. Содержит свойства, в которые передаются данные при обновлении интерфейса (то есть ввода данных в текст боксы). Эти свойства используются при проверки данных.

1. Services

UserService.cs – класс, который содержит в себе запросы к базе данных к коллекции «Users». На данной странице используются такие запросы – IsUserExists, LoginUser. IsUserExists - проверяет наличия пользователя в базе данных. LoginUser – проверяет корректность ввода данных.

* **CategoryWnd**

Данное окно открывается при взаимодействии со страницей **UserInterface (**При нажатии на выведенную категории). В нем отображается список тестов, которые содержит выбранная категория.

При клике на один их тестов, произойдет переход на страницу **TestForm**, где можно будет пройти тест

Классы относящиеся к этой страницы:

1. Viewmodels

СategoryVM.cs – класс, имеющий интерфейс InotifyPropertyChanged, содержит в себе коллекцию – TestLists и метод GetTestsItems(int cathegoryid). Метод записывает в коллекцию список тестов, который относится к выбранной категории.

1. Services

CathegoryServices.cs – класс, содержит в себе запросы к базе данных к коллекции «Cathegories». В данном окне используются следующий запрос: GetCathegoryAsunc(), который возвращает все категории которые есть в базе данных.

* **CreateNewUser**

Окно для добавление нового пользователя (Для админа). Данное окно вызывается в странице AdminInterface, при вызове контекстного меню. При нажатии кнопки «добавить» в базу данных добавляется новый пользователь.

Классы относящиеся к этой страницы:

1. Viewmodels

AdminViewModel.cs - класс, имеющий интерфейс InotifyPropertyChanged. Содержит в себе коллекции и свойства.

Коллекции нужны для вывода информации в таблицу или list view, свойства, для получения некоторых данных при взаимодействии с объектами на странице.

1. Services

UserService.cs – класс, который содержит в себе запросы к базе данных к коллекции «Users». На данной странице используются такие запросы – IsUserExists, RegisterUser.

IsUserExists - проверяет наличия пользователя в базе данных.

RegisterUser – добавляет пользователя в базу данных.

* **EditUserWindowForAdmin**

Окно редактирования существующего пользователя(Админа). Данное окно вызывается в странице AdminInterface, при вызове контекстного меню. В данном окне выводится информация о выбранном пользователя в текст боксы, ее можно изменить. При нажатии на кнопку «Изменить» данные выбранного пользователя обновляются в базе данных.

Классы относящиеся к этой страницы:

1. ViewModels

AdminViewModel.cs - класс, имеющий интерфейс InotifyPropertyChanged. Содержит в себе коллекции и свойства.

Коллекции нужны для вывода информации в таблицу или list view, свойства, для получения некоторых данных при взаимодействии с объектами на странице.

1. Services

UserService.cs – класс, который содержит в себе запросы к базе данных к коллекции «Users». На данной странице используется следующий запрос UpdateUser.

UpdateUser – обновляет выбранного пользователя в базу данных.

* **ProfileUser**

Страница профиля пользователя. На данной странице отображаются личные данные пользователя, пользователь может изменять эти данные, путем взаимодействия с кнопками («Разрешить редактирование» -> Редактировать -> «Сохранить»).

Классы относящиеся к этой страницы:

1. Viewmodels

ProfileVm.cs – Класс, имеющий интерфейс InotifyPropertyChanged, содержит в себе свойства и методы. Методы – GetUserBylogin(login) – ищет информацию об авторизованном пользователе.

1. Services

UserServices.cs – класс, который содержит в себе запросы к базе данных к коллекции «Users». На данной странице используется следующие запросы UpdateUser, SelectUsers.

* **RegistrationWindows**

Окно регистрации. Данное окно открывается при взаимодействии со страницей AuthorizationPage (при нажатии кнопки регистрации). В данном окне пользователь проходит регистрацию.

Классы относящиеся к этой страницы:

1. Viewmodels

-

1. Services

UserServices.cs – класс, который содержит в себе запросы к базе данных к коллекции «Users». На данной странице используется следующие запрос IsUserExists, RegisterUser.

* **ResultsPage**

Страница результатов, при выборе в combobox-ах, темы и категории, пользователю выводит его результаты в правом окне.

Классы относящиеся к этой страницы:

1. Viewmodels

ResultsVM.cs – Класс, имеющий интерфейс InotifyPropertyChanged, содержит в себе коллекции, свойства, методы. Методы: GetTemaList () – заполняет коллекцию темами, GetCategories () – заполняет коллекцию категориями, GetItemsCategory () – заполняет коллекцию результатами тестов определенного пользователя;

1. Services

UserServices.cs – Класс, который содержит в себе запросы к базе данных к коллекции «Users». Запросы, используемые на этой странице SelectUsers ()

ResultsServices.cs - Класс, который содержит в себе запросы к базе данных к коллекции «Results». Запросы, используемые на этой странице GetResulstsByLogin ()

* **TestForm**

Страница которая отображает вопросы выбранного теста.

Классы относящиеся к этой страницы:

1. Viewmodels

TestFormVm.cs – Класс, имеющий интерфейс InotifyPropertyChanged, содержит коллекции и свойства, метод. Метод – GetQuestInfo. Данный метод получает информацию о вопросе, который будет показан пользователю, при определении информации о вопросе он удаляет из коллекции данный вопрос, и передает этот вопрос в значение свойства CurrentQuest, после из этого свойства интерфейс получает информацию об вопросе и выводит это пользователю.

1. Services

TestServices.cs – Класс, который содержит в себе запросы к базе данных к коллекции «Tests». Запросы, используемые на этой странице GetQuestionByTestAsync, GetQuestions.

ResultServices.cs – Класс, который содержит в себе запросы к базе данных к коллекции «Results». Запросы, используемые на этой странице IsResultsExists, RegisterResults, UpdateResults

UserServices.cs – Класс, который содержит в себе запросы к базе данных к коллекции «Users». Запросы, используемые на этой странице SelectUsers()

* **UserInterface**

Страница пользовательского интерфейса, в ней пользователь выбирает тему и переходит по одной из категорий выбранной темы.

Классы относящиеся к этой страницы:

1. Viewmodels

UserInterfacesViewModel.cs – Класс, имеющий интерфейс InotifyPropertyChanged. Содержит в себе свойства, методы, коллекции. Методы: GetCategories (int temaid) – заполняет коллекцию категорий, GetTema () – заполняет коллекцию тем.

1. Services

UserServices.cs - Класс, который содержит в себе запросы к базе данных к коллекции «Users». Запросы, используемые на этой странице SelectUsers ()

TemaServices.cs – Класс, который содержит в себе запросы к базе данных к коллекции «Tema». Запросы, используемые на этой странице GetTema ()

CategoryServices.cs – Класс, который содержит в себе запросы к базе данных к коллекции «Cathegories». Запросы, используемые на этой странице GetCategoriesByTema()

* **ResultsWnd**

Окно результатов. Данное окно открывается при взаимодействии со страницей AdminInterface. В данной странице админ выбирает в combobox-ах тему и категорию, сверху над этими combobox-ами располагается текстовое поле с логином выбранного пользователя. После выбора темы и категории, админ получит информацию о результатах выбранного пользователя.

Классы относящиеся к этой страницы:

1. ViewModels

ResultsVM - Класс, имеющий интерфейс InotifyPropertyChanged, содержит в себе коллекции, свойства, методы. Методы: GetTemaList () – заполняет коллекцию темами, GetCategories () – заполняет коллекцию категориями, GetItemsCategory () – заполняет коллекцию результатами тестов определенного пользователя;

1. Services

ResultsServices.cs - Класс, который содержит в себе запросы к базе данных к коллекции «Results». Запросы, используемые на этой странице GetResulstsByLogin ()

## **5.3. Руководство пользователя**

## **5.3.1 Назначение программы**

Разработка проводится на основании приказа о курсовом проектировании.

Программный продукт «Victorina» – программа, созданная для работы с базой данных по предметной области.

Автоматизация продукта «Victorina» - процесс внедрения программно-аппаратных комплексов для облегчения и автоматизации.

## **5.3.2 Условия выполнения программы**

Минимальные характеристики компьютера

|  |  |
| --- | --- |
| Процессор | AMD Athlon |
| Память | От 4 гб |
| Разрешение экрана | От 1280x1024 |
| Устройства ввода | Клавиатура, мышь |
| Дисковое пространство | От 1 Гб |
|  |  |

Оптимальный характеристики компьютера

|  |  |
| --- | --- |
| Процессор | Intel Core i5 3ГГц |
| Память | От 8 гб |
| Разрешение экрана | От 1920x1080 |
| Устройства ввода | Клавиатура, мышь |
| Дисковое пространство | От 1 Гб |

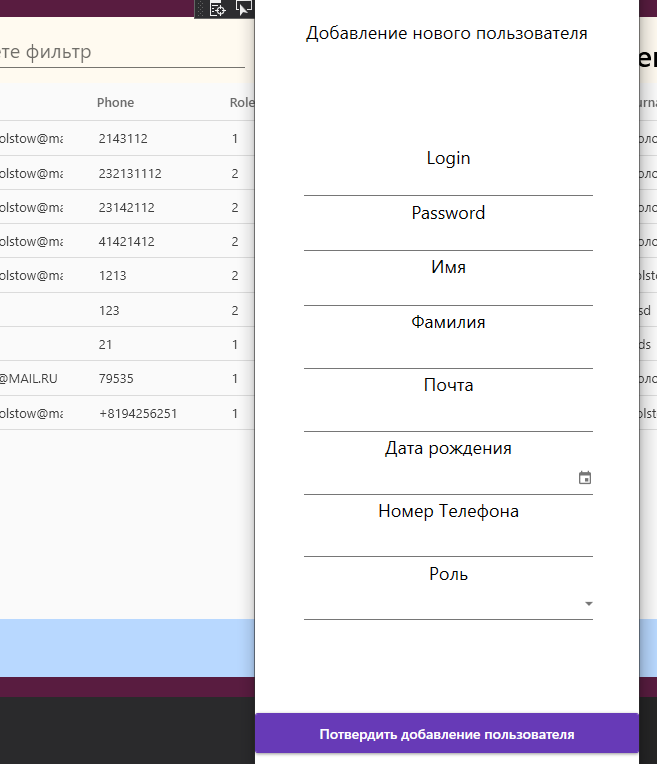
Для корректной работы программы необходимо: ОС Windows 7-10;

## **5.3.3 Выполнение программы**

При открытии программы (исполняемый «.exe» файл) появится окно авторизации, где необходимо ввести пароль и логин. Если введённый пароль не правильный – пользователя уведомит об этом диалоговое окно с ошибкой.

**Добавление пользователя(Админ)**

* + - 1. Пкм по свободному пространству в таблице и выбор в контекстном меню «Добавление нового пользователя»
      2. Ввод данных пользователя в поля
      3. Клик по кнопке «Добавить»
      4. Пользователь добавлен

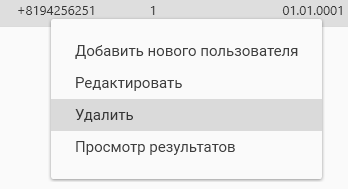


**Удаление пользователя(Админ)**

Пкм по пользователю в таблице

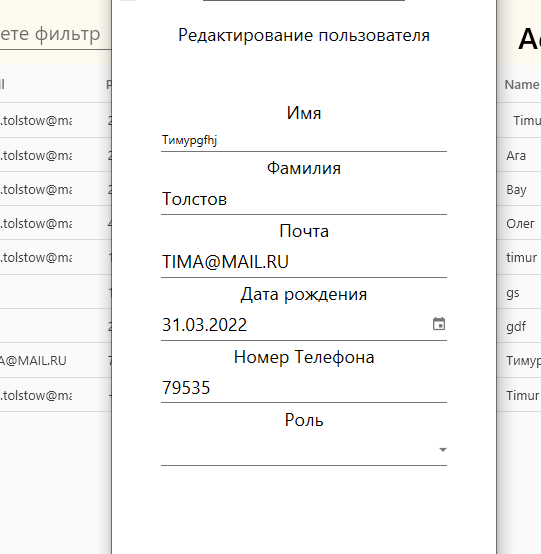
Выбор к контекстном меню «Удалить выбранного пользователя»

Пользователь удален.



**Редактирование пользователя(Админ)**

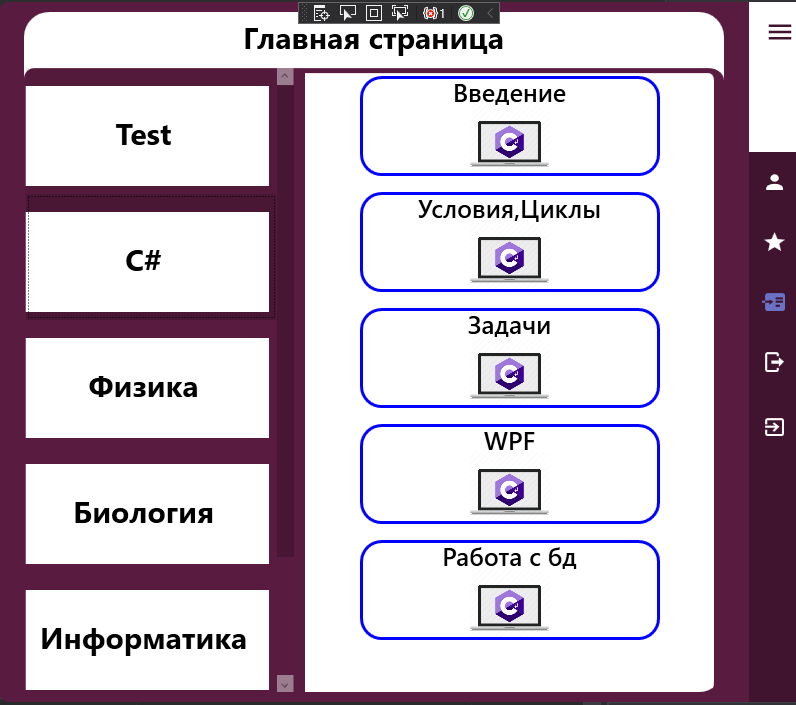
* + - 1. Пкм по пользователю в таблице и выбор в контекстном меню «Редактирование пользователя»
      2. Изменение данные выбранного пользователя в полях ввода
      3. Клик по кномке «Изменить»
      4. Информация о пользователи изменена.



**Просмотр категорий. (Студент)**

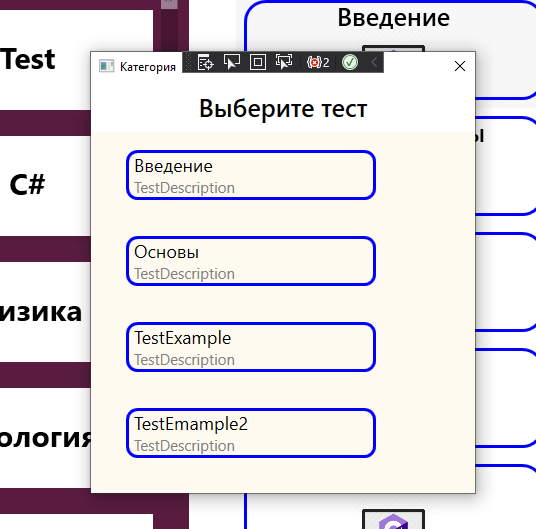
На странице «Пользовательского интерфейса выбрать тему в левом окне(Клик по ней)

В правом окне отобразятся категории



**Просмотр списка тестов(Студент)**

* + - 1. После выбора темы на странице «Пользовательского интерфейса, клик по категории.
      2. Появится всплывающее окно со списком тестов.

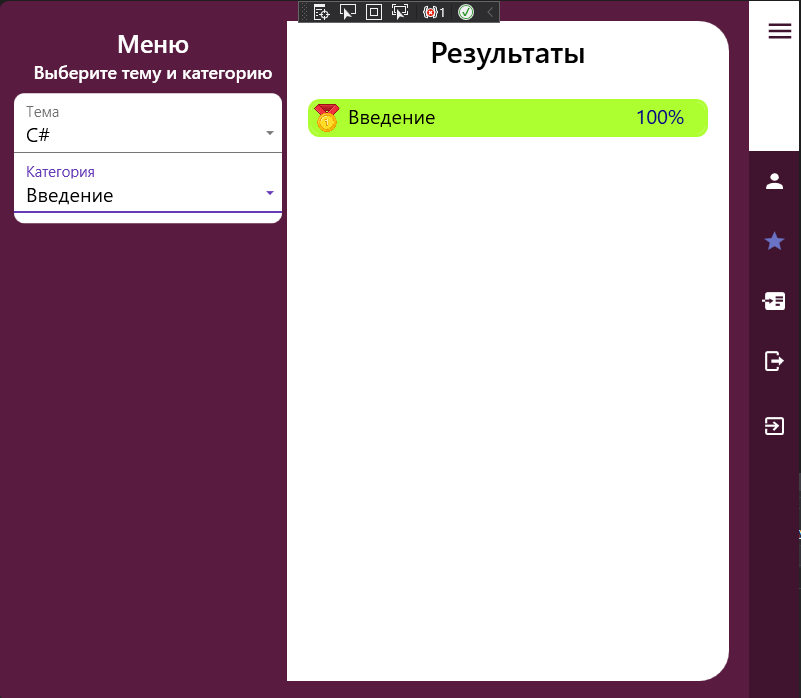


**Просмотр своих результатов(Студент)**

Переход на страницу результатов

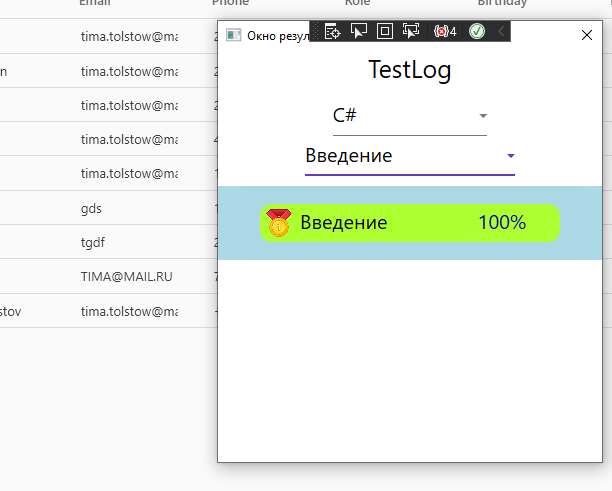
В правом меню выбрать тему и категорию по которой будут отображены результаты

Результаты появились в правом окне(Если они есть)



**Просмотр чужих результатов(Админ)**

* + - 1. Пкм по пользователю в таблице
      2. Клик в контекстном меню «Просмотр результатов»
      3. Появится в всплывающее окно с результатами выбранного пользователя.



## **5.3.4 Выполнение программного модуля**

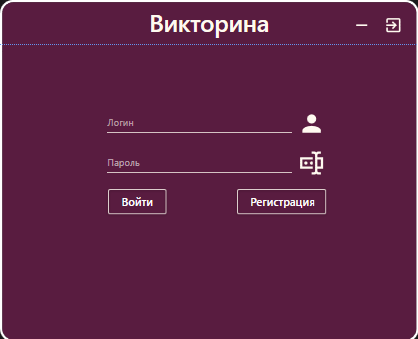
Для запуска программы надо запустить Victorina.exe, который находит по пути:

\*\Курсовая Толстов\Victorina \Victorina\bin\Debug\

Существуют 2 роли: Администратор – самый главный в системе и обычный пользователь – человек с определенным функционалом.

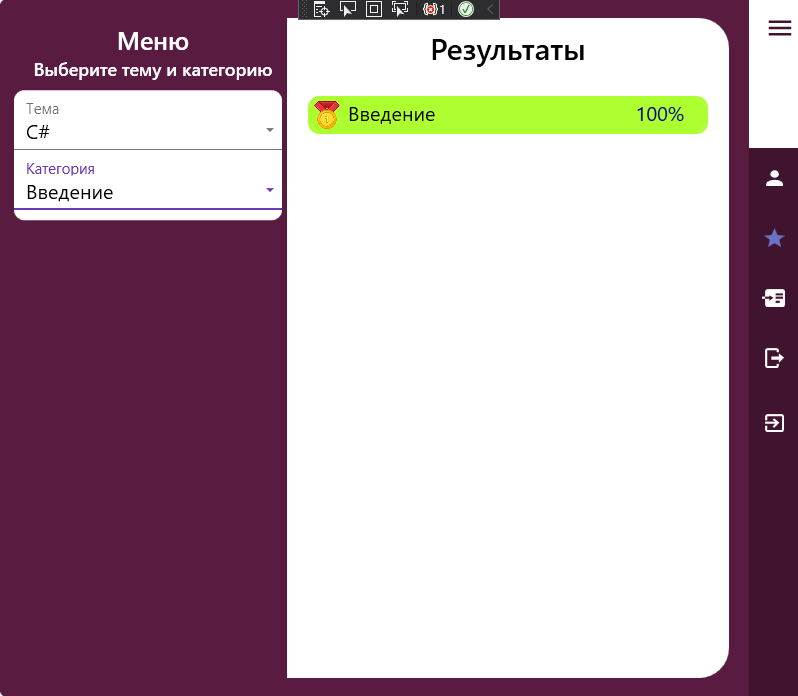
Администратор – может добавлять новых пользователей, и назначать роли, также редактировать информацию пользователей.

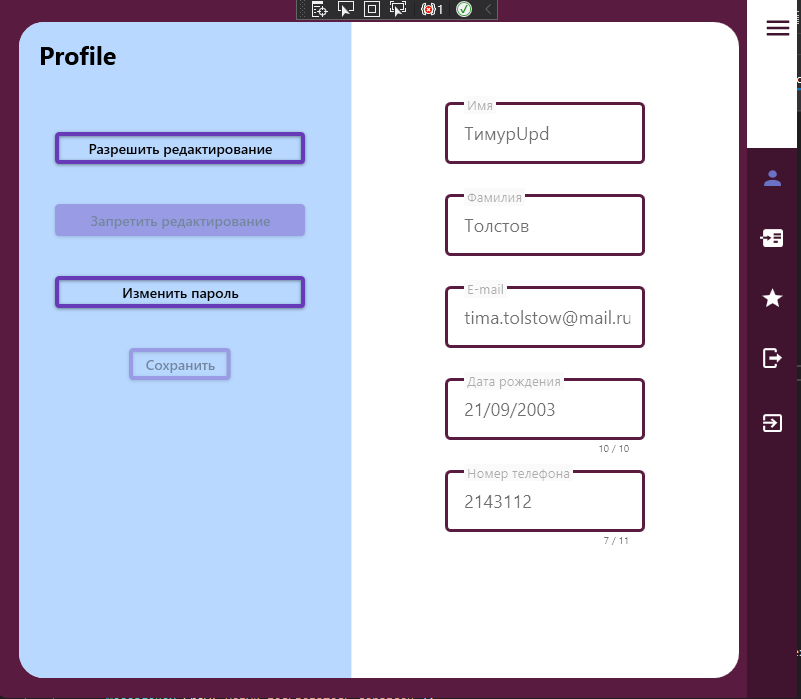
Обычный пользователь – может проходить тесты, изменять личную информацию на определенной странице, также приложение предоставляет результаты пройденных тестов.

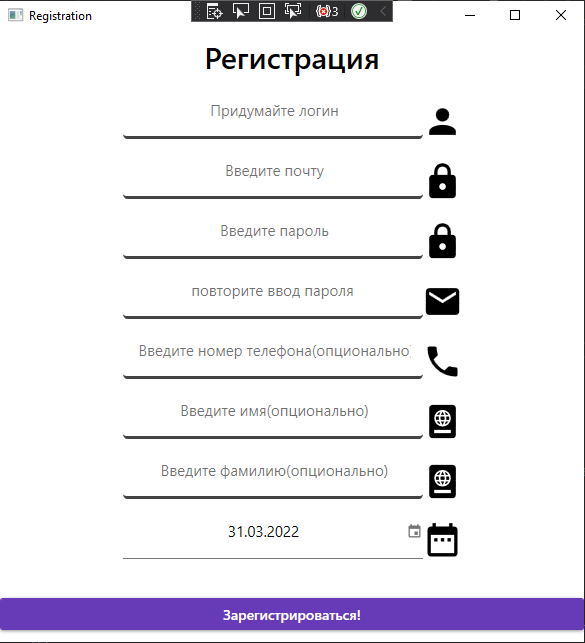


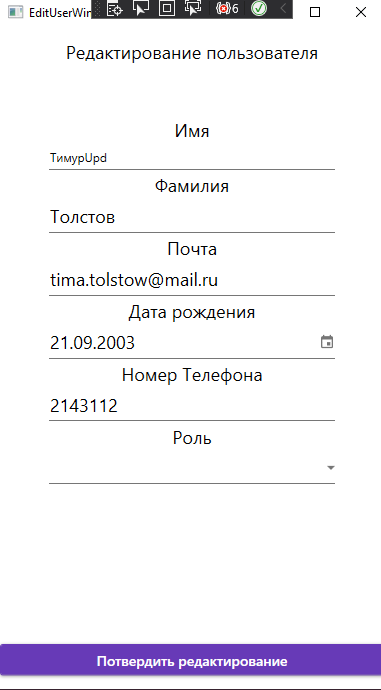


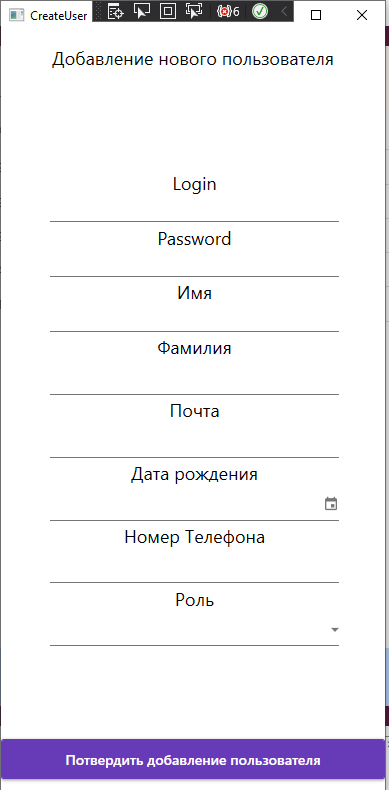


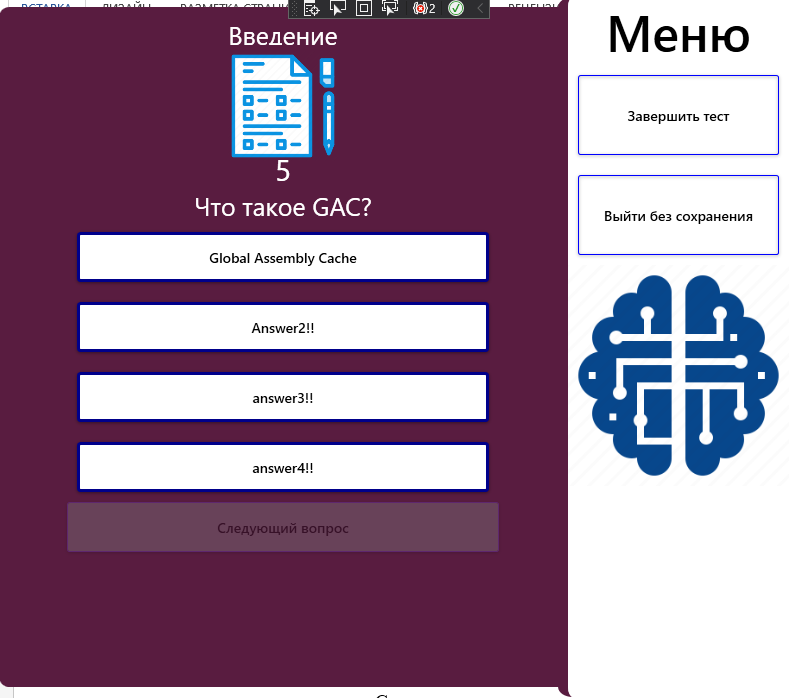








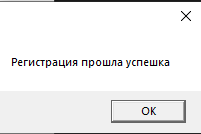




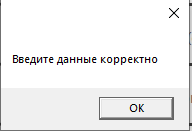
## **5.3.5 Сообщения пользователю**

* Сообщения при регистрации.

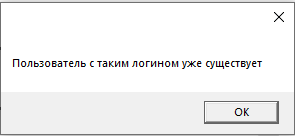
1. Успешная регистрация.



1. Некорректный ввод данных.

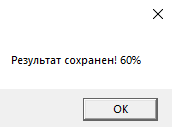


1. Пользователь уже существует.

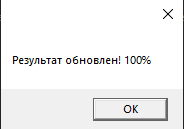


* Сообщения после прохождения тестов.

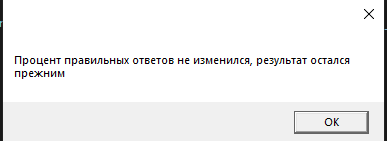
1. Новый результат.



1. Обновление существующего.

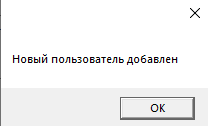


1. Результат не сохранен.

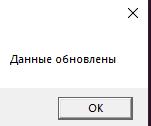


* Сообщения администратору.

1. При добавлении пользователя

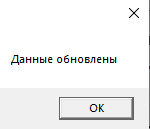


1. При редактировании пользователя.



* Сообщения пользователю при изменении личных данных.

1. Изменение личных данных прошло успешно.



# Заключение

Для информационной системы «Викторина» был разработан программный модуль.

В результате выполненного исследования в рамках курсового проектирования разработан программный модуль для информационной системы Викторина. В процессе разработки использовалась облачная база данных Firebase, применена технология Entity Framework, язык программирования C# Visual Studio. Приложение соответствует поставленным задачам, реализованы функции: просмотр, добавление, сортировка, выборка. Программный модуль может быть установлен на любую операционную систему, начиная с Windows 7 x32. В проекте используется язык запросов LINQ + добавление некоторых языковых особенностей с библиотеки FireBase.Database.

В дальнейшем в приложении могут быть реализованы функции:

**1. Возможность изменения данных**

1.1 Открыть профиль/админ

1.2 Открыть контекстное меню/нажать на кнопку разрешить редактирование.

1.3 В окне изменить нужные значения, подтвердить изменения.

**2. Возможность прохождения тестов**

2.1 Выбрать любую тему и любую категорию.

2.2 нажать на любой тест.

**3. Просмотр результатов тестов**

3.1 Просмотр как пользователь

3.1.1 зайти на страницу результатов.

3.1.2 выбрать тему и категорию.

3.2 Просмотр как админ

3.2.1 Выбрать пользователя в таблице(ПКМ)

3.2.2 В контекстном меню выбрать пункт «Просмотр результатов».

3.2.3 Выбрать тему и категорию.

# Список литературы

1. Федорова Г.Н. Разработка модулей программного обеспечения для компьютерных систем: учебник для студентов СПО /Г.Н. Федорова. 2-е изд., стер. - М.: Издательский центр «Академия»,2018. - 384с.
2. Федорова Г.Н. Осуществление интеграции программных модулей: учебник для студентов СПО / Г.Н. Федорова. 2-е изд., стер. - М.: Издательский центр «Академия»,2018. - 285 с.
3. Царёв, Р.Ю. Алгоритмы и структуры данных (CDIO): учебник /Р.Ю. Царёв, А.В. Прокопенко; Министерство образования и науки Российской Федерации, Сибирский Федеральный университет. - Красноярск: СФУ, 2016. - 204 с. : ил. - Библиогр. в кн. - ISBN 978-5-7638-3388-1; То же [Электронный ресурс]. - URL: <http://biblioclub.ru/index.php?page=book&id=497016>
4. Гниденко, И. Г. Технология разработки программного обеспечения : учебное пособие для среднего профессионального образования / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. — Москва: Издательство Юрайт, 2020. — 235 с.
5. C# 7.0. Справочник. Полное описание языка.: Пер. с англ. – СпБ.: ООО “Альфакнига”, 2018. – 1024 с. : ил. – Парал. тит. англ.
6. Microsoft Visual C#. Подробное руководство. 8-е изд. – СПб.: Питер, 2017. – 848 с.: ил. – (Серия «Библиотека программиста»).
7. Программирование на C# для начинающих. Основные сведения / Алексей Васильев. – Москва : Эксмо, 2018. – 592 с. – (Российский компьютерный бестселлер).
8. Язык программирования C# 7 и платформы .NET и .NET Core, 8-е изд. : Пер. с англ. – СПб. : ООО “Диалектика”, 2018 – 1328 с. : ил. – Парал. тит. англ.
9. Оппель, Эндрю Дж. SQL. Полное руководство / Оппель Эндрю Дж.. - М.: Диалектика / Вильямс, 2016. - 902 c.

# Приложения

## Приложение1

menu = {  
 '1': 'Добавить запись',  
 '2': 'Удалить запись',  
 '3': 'Вывести все записи о товаре на указанном складе: ',  
 '4': 'Вывести всю информацию о товаре с ценой меньше указанной: ',  
 '5': 'Вывести всю информацию о товаре с указанной ед. изм.: ',  
 '6': 'Вывести среднюю цену товара по складам в формате: «название склада: сред. Цена»: ',  
 '7': 'Вывести всю информацию об указанном товаре: ',  
 '8': 'Вывести все данные записи и сумму по каждой записи. Сумма = Цена\*кол-во: ',  
 '9': 'Ввывести все данные',  
 '0': ' <-Меню',  
 'exit': 'Выйти'  
}  
  
staff = [{'id': 1, 'Название продовольствия': 'Хлеб','Цена': 115, 'Ед.изм': 'Кг', 'Кол-во': 42, 'Склад №': 1},  
{'id': 2, 'Название продовольствия': 'Арбуз', 'Цена': 100, 'Ед.изм': 'Кг', 'Кол-во': 83, 'Склад №': 2},  
{'id': 3, 'Название продовольствия': 'Клубника', 'Цена': 85, 'Ед.изм': 'Кг', 'Кол-во': 100, 'Склад №': 3},  
{'id': 4, 'Название продовольствия': 'Фисташки', 'Цена': 305, 'Ед.изм': 'Кг', 'Кол-во': 52, 'Склад №': 4},  
{'id': 5, 'Название продовольствия': 'Арахис', 'Цена': 150, 'Ед.изм': 'Кг', 'Кол-во': 22, 'Склад №': 5},  
{'id': 6, 'Название продовольствия': 'Орех', 'Цена': 400, 'Ед.изм': 'Кг', 'Кол-во': 82, 'Склад №': 5},  
{'id': 7, 'Название продовольствия': 'Простоквашино', 'Цена': 75, 'Ед.изм': 'Кг', 'Кол-во': 67, 'Склад №': 4},  
{'id': 8, 'Название продовольствия': 'Ежевика', 'Цена': 65, 'Ед.изм': 'Кг', 'Кол-во': 42, 'Склад №': 3},  
{'id': 9, 'Название продовольствия': 'Кефир', 'Цена': 40, 'Ед.изм': 'Л', 'Кол-во': 58, 'Склад №': 2},  
{'id': 10,'Название продовольствия': 'Квас', 'Цена': 90, 'Ед.изм': 'Л', 'Кол-во': 754, 'Склад №': 1}  
 ]  
  
for key, val in menu.items():  
 print(key, '-', val)  
  
while True:  
 action = input('>\_ ')  
  
 if action == '1':  
 NazProd = input('Название продовольствия:')  
 price = int(input('Цена:'))  
 Edizmer = input('Ед.изм:')  
 kolvo = int(input('Кол-во:'))  
 sklad = input('Cклад №:')  
 lid = staff[-1]['id']  
 staff.append({'id': lid + 1, 'Название продовольствия': NazProd, 'Цена': price, 'Ед.изм': Edizmer, 'Кол-во': kolvo, 'Cклад №': sklad})  
 print('Данные добавлены!')  
  
 elif action == '2':  
 key\_del = input('Код продукции: ')  
 fl = False  
 for i in range(len(staff)):  
 if staff[i]['id'] == key\_del:  
 del staff[i]  
 print('Данные удалены!')  
 fl = True  
 break  
 if fl == False:  
 print('Такого кода нет в списке!')  
  
 elif action == '3':  
 sklad = int(input('Введите номер склада:'))  
 fl = False  
 for i in range(len(staff)):  
 if sklad == staff[i]['Склад №']:  
 print("{}\t{}\t{}\t{}\t{}".format(staff[i]['id'], staff[i]['Название продовольствия'],  
 staff[i]['Цена'], staff[i]['Ед.изм'], staff[i]['Кол-во']))  
 fl = True  
 if fl == False:  
 print("Склад №", sklad, "не найден!")  
  
 elif action == '4':  
 price = int(input('Введите цену: '))  
 fl = False  
 for i in range(len(staff)):  
 if price > staff[i]['Цена']:  
 print("{}\t{}\t{}\t{}\t{}".format(staff[i]['id'], staff[i]['Название продовольствия'],  
 staff[i]['Цена'], staff[i]['Ед.изм'], staff[i]['Кол-во']))  
 fl = True  
 if fl == False:  
 print("Ниже такой цены ничего нету")  
  
 elif action == '5':  
 Edizmer = input('Введите ед.измерения: ')  
 fl = False  
 for i in range(len(staff)):  
 if Edizmer == staff[i]['Ед.изм']:  
 print("{}\t{}\t{}\t{}\t{}".format(staff[i]['id'], staff[i]['Название продовольствия'],  
 staff[i]['Цена'], staff[i]['Ед.изм'], staff[i]['Кол-во']))  
 fl = True  
 if fl == False:  
 print("Такого единицы", Edizmer, "Нету")  
  
 elif action == '6':  
  
 src\_sklad5 = 0  
 src\_sklad4 = 0  
 src\_sklad3 = 0  
 src\_sklad2 = 0  
 src\_sklad1 = 0  
  
 delitel1 = 0  
 delitel2 = 0  
 delitel3 = 0  
 delitel4 = 0  
 delitel5 = 0  
  
 for i in range(len(staff)):  
  
 if 4 == staff[i]['Склад №']:  
 src\_sklad4 += staff[i]['Цена']  
 delitel4 += 1  
  
 elif 3 == staff[i]['Склад №']:  
 src\_sklad3 += staff[i]['Цена']  
 delitel3 += 1  
  
 elif 2 == staff[i]['Склад №']:  
 src\_sklad2 += staff[i]['Цена']  
 delitel2 += 1  
  
 elif 1 == staff[i]['Склад №']:  
 src\_sklad1 += staff[i]['Цена']  
 delitel1 += 1  
  
 elif 5 == staff[i]['Склад №']:  
 src\_sklad5 += staff[i]['Цена']  
 delitel5 += 1  
 src\_sklad5 /= delitel5  
 src\_sklad4 /= delitel4  
 src\_sklad3 /= delitel3  
 src\_sklad2 /= delitel2  
 src\_sklad1 /= delitel1  
 print("{}\t{}\t".format(1, round(src\_sklad1,3)))  
 print("{}\t{}\t".format(2, round(src\_sklad2,3)))  
 print("{}\t{}\t".format(3, round(src\_sklad3,3)))  
 print("{}\t{}\t".format(4, round(src\_sklad4,3)))  
 print("{}\t{}\t".format(5, round(src\_sklad5,3)))  
  
 elif action == '7':  
 NazProd = input('Введите продукцию:')  
 fl = False  
 for i in range(len(staff)):  
 if NazProd == staff[i]['Название продовольствия']:  
 print("{}\t{}\t{}\t{}\t{}".format(staff[i]['id'], staff[i]['Название продовольствия'],  
 staff[i]['Цена'], staff[i]['Ед.изм'], staff[i]['Кол-во']))  
 fl = True  
 if fl == False:  
 print("Продукция", NazProd, "не найден!")  
  
 elif action == '8':  
 sum = 0  
 for i in range(len(staff)):  
 sum = staff[i]['Цена'] \* staff[i]['Кол-во']  
 print("{}\t{}\t{}\t{}\t{}\tСумма: {}".format(staff[i]['id'], staff[i]['Название продовольствия'],  
 staff[i]['Цена'], staff[i]['Ед.изм'], staff[i]['Кол-во'], sum))  
  
 elif action == '9':  
 for i in range(len(staff)):  
 print("{}\tНазвание продовольствия: {:13}\tЦена: {:4}\tЕд.изм: {:3}\tКол-во: {:4}\tСклад №{:3}".format(staff[i]['id'], staff[i]['Название продовольствия'],  
 staff[i]['Цена'], staff[i]['Ед.изм'], staff[i]['Кол-во'],staff[i]['Склад №']))  
  
 elif action == '0':  
 for key, val in menu.items():  
 print(key, '-', val)  
 elif action == 'exit':  
 break

## Приложение2

import csv  
from builtins import print  
  
FILENAME = "users2.csv"  
  
menu = {  
 '1': 'Добавить запись',  
 '2': 'Удалить запись',  
 '3': 'Вывести все записи о товаре на указанном складе: ',  
 '4': 'Вывести всю информацию о товаре с ценой меньше указанной: ',  
 '5': 'Вывести всю информацию о товаре с указанной ед. изм.: ',  
 '6': 'Вывести среднюю цену товара по складам в формате: «название склада: сред. Цена»: ',  
 '7': 'Вывести всю информацию об указанном товаре: ',  
 '8': 'Вывести все данные записи и сумму по каждой записи. Сумма = Цена\*кол-во: ',  
 '9': 'Ввывести все данные',  
 '10':'Открыть файл',  
 '11':'Сохранить в файл',  
 '0': ' <-Меню',  
 'exit': 'Выйти'  
}  
  
staff = []  
  
for key, val in menu.items():  
 print(key, '-', val)  
  
while True:  
 action = input('>\_ ')  
  
 if action == '1':  
 lid = int(input('Введите код: '))  
 NazProd = input('Название продовольствия:')  
 price = int(input('Цена:'))  
 Edizmer = input('Ед.изм:')  
 kolvo = int(input('Кол-во:'))  
 sklad = input('Cклад №:')  
 staff.append({'id': lid, 'Название продовольствия': NazProd, 'Цена': price, 'Ед.изм': Edizmer, 'Кол-во': kolvo, 'Склад №': sklad})  
 print('Данные добавлены!')  
  
 elif action == '2':  
 key\_del = input('Код продукции: ')  
 fl = False  
 for i in range(len(staff)):  
 if staff[i]['id'] == key\_del:  
 del staff[i]  
 print('Данные удалены!')  
 fl = True  
 break  
 if fl == False:  
 print('Такого кода нет в списке!')  
  
 elif action == '3':  
 sklad = input('Введите номер склада:')  
 fl = False  
 for i in range(len(staff)):  
 if sklad == staff[i]['Склад №']:  
 print("{}\t{}\t{}\t{}\t{}".format(staff[i]['id'], staff[i]['Название продовольствия'],  
 staff[i]['Цена'], staff[i]['Ед.изм'], staff[i]['Кол-во']))  
 fl = True  
 if fl == False:  
 print("Склад №", sklad, "не найден!")  
  
 elif action == '4':  
 price = int(input('Введите цену: '))  
 fl = False  
 for i in range(len(staff)):  
 if price > int(staff[i]['Цена']):  
 print("{}\t{}\t{}\t{}\t{}".format(staff[i]['id'], staff[i]['Название продовольствия'],  
 staff[i]['Цена'], staff[i]['Ед.изм'], staff[i]['Кол-во']))  
 fl = True  
 if fl == False:  
 print("Ниже такой цены ничего нету")  
  
 elif action == '5':  
 Edizmer = input('Введите ед.измерения: ')  
 fl = False  
 for i in range(len(staff)):  
 if Edizmer == staff[i]['Ед.изм']:  
 print("{}\t{}\t{}\t{}\t{}".format(staff[i]['id'], staff[i]['Название продовольствия'],  
 staff[i]['Цена'], staff[i]['Ед.изм'], staff[i]['Кол-во']))  
 fl = True  
 if fl == False:  
 print("Такого единицы", Edizmer, "Нету")  
  
 elif action == '6':  
  
 src\_sklad5 = 0  
 src\_sklad4 = 0  
 src\_sklad3 = 0  
 src\_sklad2 = 0  
 src\_sklad1 = 0  
  
 delitel1 = 0  
 delitel2 = 0  
 delitel3 = 0  
 delitel4 = 0  
 delitel5 = 0  
  
 for i in range(len(staff)):  
  
 if 4 == int(staff[i]['Склад №']):  
 src\_sklad4 += int(staff[i]['Цена'])  
 delitel4 += 1  
  
 elif 3 == int(staff[i]['Склад №']):  
 src\_sklad3 += int(staff[i]['Цена'])  
 delitel3 += 1  
  
 elif 2 == int(staff[i]['Склад №']):  
 src\_sklad2 += int(staff[i]['Цена'])  
 delitel2 += 1  
  
 elif 1 == int(staff[i]['Склад №']):  
 src\_sklad1 += int(staff[i]['Цена'])  
 delitel1 += 1  
  
 elif 5 == int(staff[i]['Склад №']):  
 src\_sklad5 += int(staff[i]['Цена'])  
 delitel5 += 1  
  
 src\_sklad5 /= delitel5  
 src\_sklad4 /= delitel4  
 src\_sklad3 /= delitel3  
 src\_sklad2 /= delitel2  
 src\_sklad1 /= delitel1  
  
 print("{}\t{}\t".format(1, round(src\_sklad1,3)))  
 print("{}\t{}\t".format(2, round(src\_sklad2,3)))  
 print("{}\t{}\t".format(3, round(src\_sklad3,3)))  
 print("{}\t{}\t".format(4, round(src\_sklad4,3)))  
 print("{}\t{}\t".format(5, round(src\_sklad5,3)))  
  
  
 elif action == '7':  
 NazProd = input('Введите продукцию:')  
 fl = False  
 for i in range(len(staff)):  
 if NazProd == staff[i]['Название продовольствия']:  
 print("{}\t{}\t{}\t{}\t{}".format(staff[i]['id'], staff[i]['Название продовольствия'],  
 staff[i]['Цена'], staff[i]['Ед.изм'], staff[i]['Кол-во']))  
 fl = True  
 if fl == False:  
 print("Продукция", NazProd, "не найден!")  
  
 elif action == '8':  
 sum = 0  
 for i in range(len(staff)):  
 sum = int(staff[i]['Цена']) \* int(staff[i]['Кол-во'])  
 print("{}\t{}\t{}\t{}\t{}\tСумма: {}".format(staff[i]['id'], staff[i]['Название продовольствия'],  
 staff[i]['Цена'], staff[i]['Ед.изм'], staff[i]['Кол-во'], sum))  
  
 elif action == '9':  
 for i in range(len(staff)):  
 print("{}\tНазвание продовольствия: {:13}\tЦена: {:4}\tЕд.изм: {:3}\tКол-во: {:4}\tСклад №{:3}".format(staff[i]['id'], staff[i]['Название продовольствия'],  
 staff[i]['Цена'], staff[i]['Ед.изм'], staff[i]['Кол-во'],staff[i]['Склад №']))  
  
 elif action == '10':  
 k = 0  
 with open(FILENAME, "r", newline="") as file:  
 reader = csv.DictReader(file)  
 for row in reader:  
 staff.append({"id": row["id"].strip(), "Название продовольствия": row["Название продовольствия"].strip(),  
 "Цена": row["Цена"].strip(), "Ед.изм": row["Ед.изм"].strip(),  
 "Кол-во": row["Кол-во"].strip(),"Склад №": row["Склад №"].strip()})  
 k += 1  
 print('Прочитано:', k, 'строк')  
  
 elif action == '11':  
 with open(FILENAME, "w", newline="") as file:  
 columns = ['id', 'Название продовольствия', 'Цена', 'Ед.изм', 'Кол-во', 'Склад №']  
 writer = csv.DictWriter(file, fieldnames=columns)  
 writer.writeheader()  
 # запись нескольких строк  
 writer.writerows(staff)  
 print("Данные добавлены!")  
  
  
 elif action == '0':  
 for key, val in menu.items():  
 print(key, '-', val)  
 elif action == 'exit':  
 break