Содержание

[Введение 2](#_Toc99599808)

[1. Назначение разработки 3](#_Toc99599809)

[2. Требования к программному модулю 4](#_Toc99599810)

[2.1 Требования к функциональным характеристикам 4](#_Toc99599811)

[2.2 Требования к составу и параметрам технических средств 4](#_Toc99599812)

[2.3 Требования к информационной и программной совместимости 5](#_Toc99599813)

[3. Руководство программиста 6](#_Toc99599814)

[3.1 Разработка базы данных 6](#_Toc99599815)

[3.2 Разработка программного модуля 10](#_Toc99599816)

[3.3 Текст программы 13](#_Toc99599817)

[4. Руководство пользователя 21](#_Toc99599818)

[4.1 Выполнение программного модуля 21](#_Toc99599819)

[4.2 Сообщения пользователю 27](#_Toc99599820)

[5. Тестирование и отладка программного модуля 29](#_Toc99599821)

[Заключение 29](#_Toc99599822)

[Список литературы 31](#_Toc99599823)

# 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.

# **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.

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

# Требования к программному модулю

## **Требования** **к функциональным характеристикам**

## 

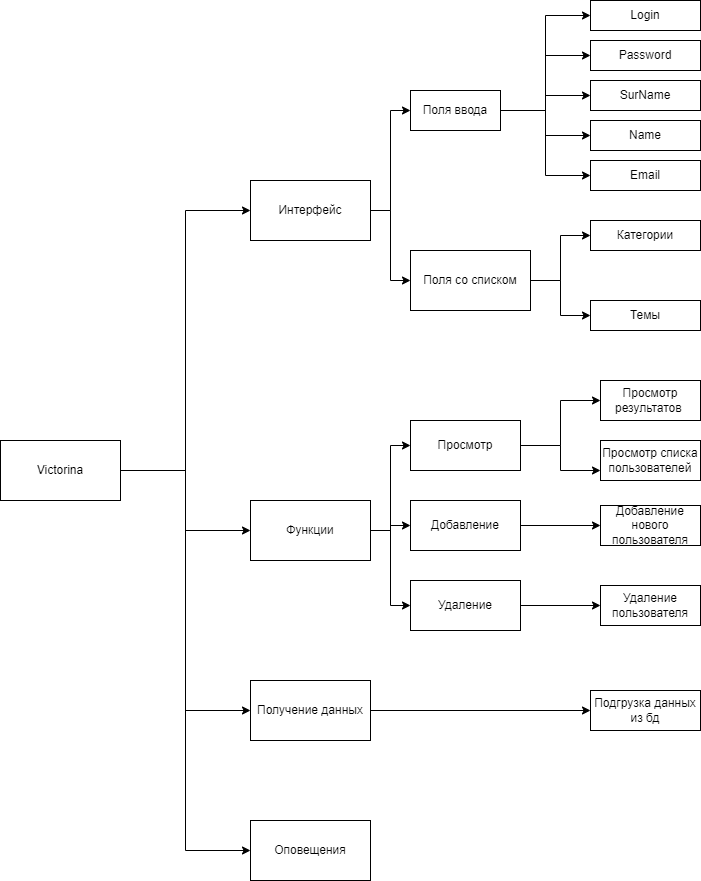


Рис №1 Функциональные характеристики

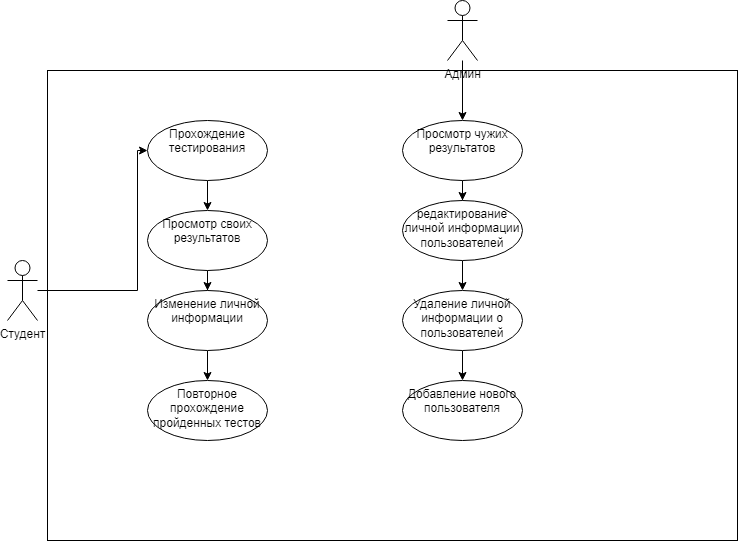


Рис №2. Сase-диаграмма

## **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 |
| Дополнительно | Постоянное подключение к сети, в которой находится база данных |
|  |  |

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 |
| Дополнительно | Постоянное подключение к сети, в которой находится база данных |
|  |  |

## **Требования** **к информационной и программной совместимости**

Для корректной работы программы необходимо: **ОС Windows 7/8/8.1/10 32х/64x**, установленный дистрибутив **.Net Framework 4.7.2.**

**Word 2016** – текстовый процессор, предназначенный для создания, просмотра, редактирования и форматирования текстов статей, деловых бумаг, а также иных документов, с локальным применением простейших форм таблично-матричных алгоритмов. Используется для ведения хода разработки и составления отчета для подведения итога работоспособности программы.

**Visual Studio 2019** (и выше) – продукт включающая интегрированную среду разработки программного обеспечения и ряд других инструментальных инструментов.

**Язык программирования C#** - компилируемый статически типизированный язык программирования общего назначения, разработанный в 1969—1973 годах сотрудником Bell Labs Деннисом Ритчи как развитие языка Би. Первоначально был разработан для реализации операционной системы UNIX, но впоследствии был перенесён на множество других платформ

**Firebase.Database** – Nu-get пакет нужен для установки соединения с удаленной базой данных firebase. Также предоставляет удобные инструменты для написания CRUD операций(Select,Update,Insert,Delete)

**FireBase** - это облачная база данных, которая позволяет пользователям хранить и получать сохраненную информацию, а также имеет удобные средства и методы взаимодействия с ней.

# **Руководство** **программиста**

## **Разработка** **базы данных**

**Создание базы данных:**

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

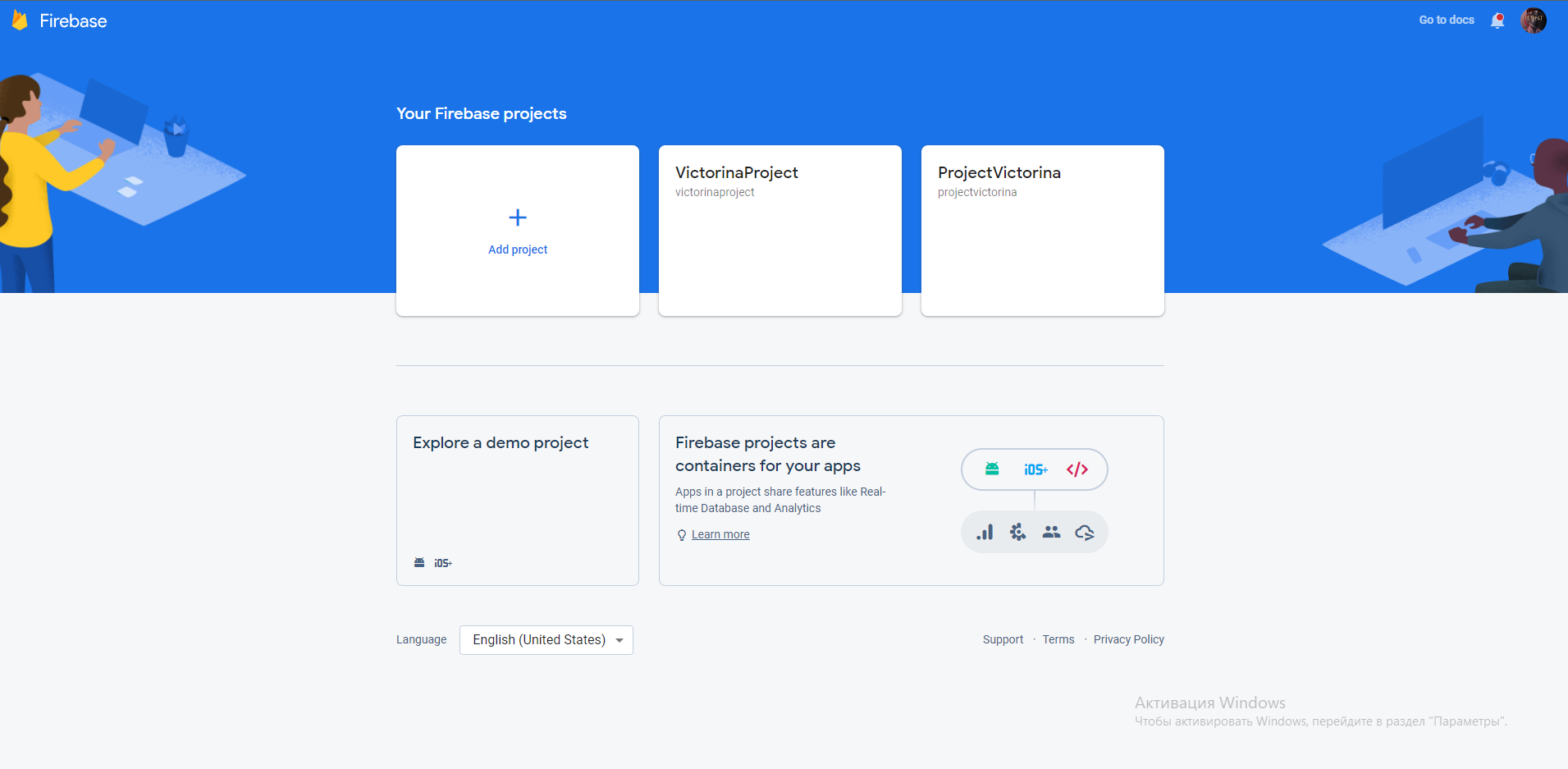


Рис №3. Создание БД

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

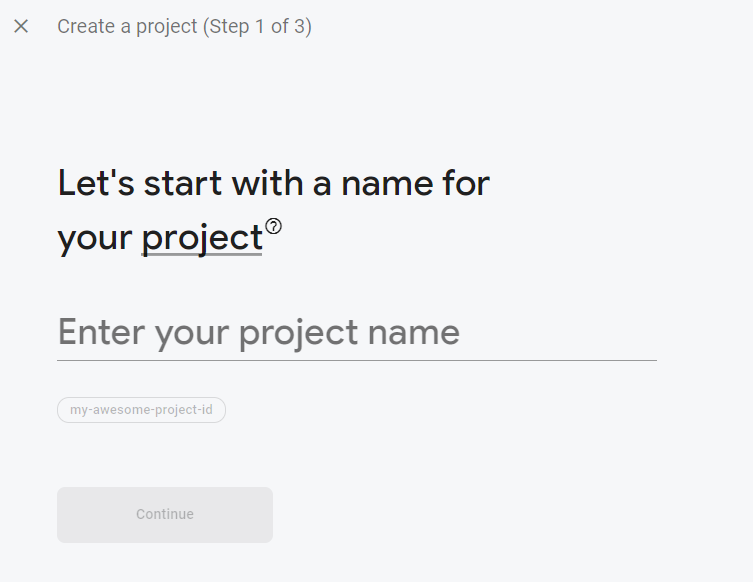


Рис №4. Создание БД

1. Нажимаем «Continue»

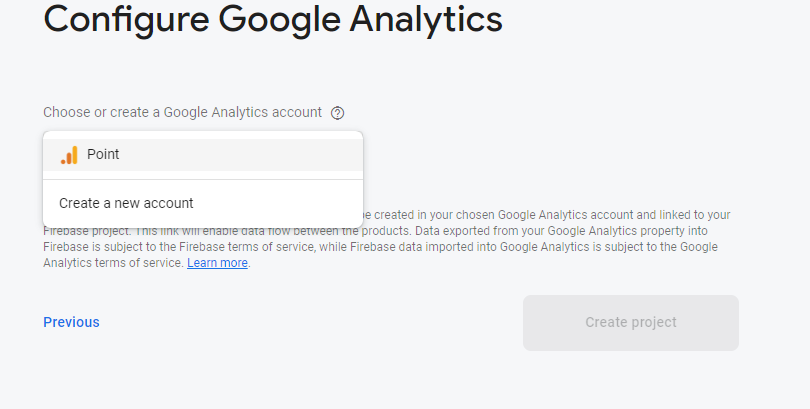


Рис №5. Создание БД

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

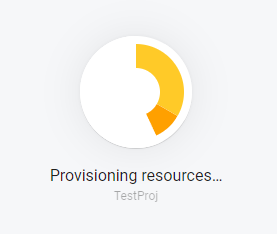


Рис №6. Создание БД

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

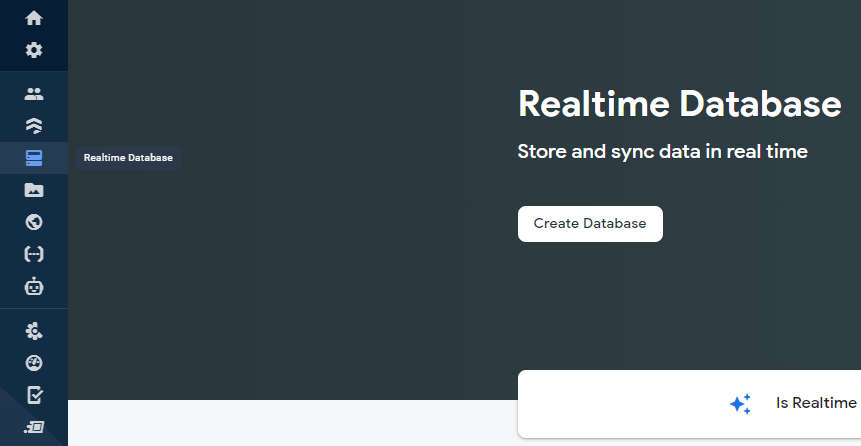


Рис №7. Создание БД

1. Выбираем страну и жмем Next.

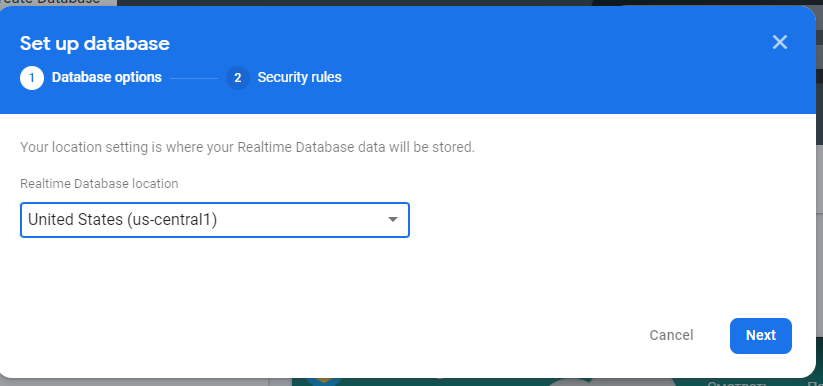


Рис №8. Создание БД

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

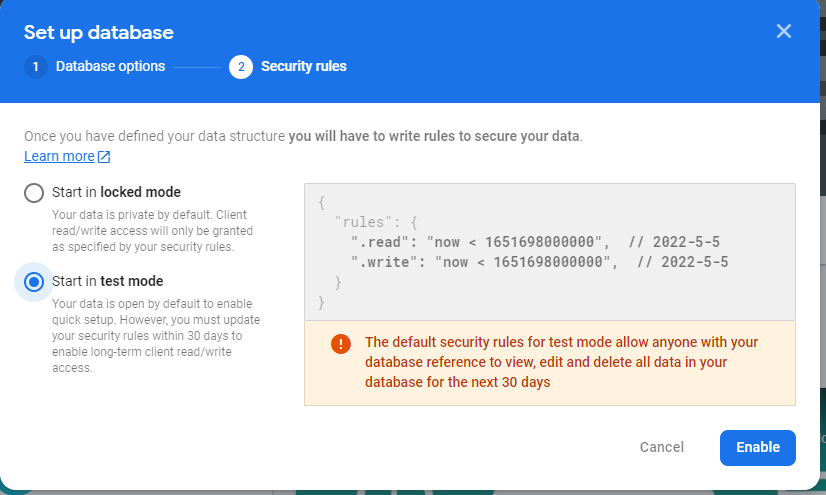


Рис №9. Создание БД

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

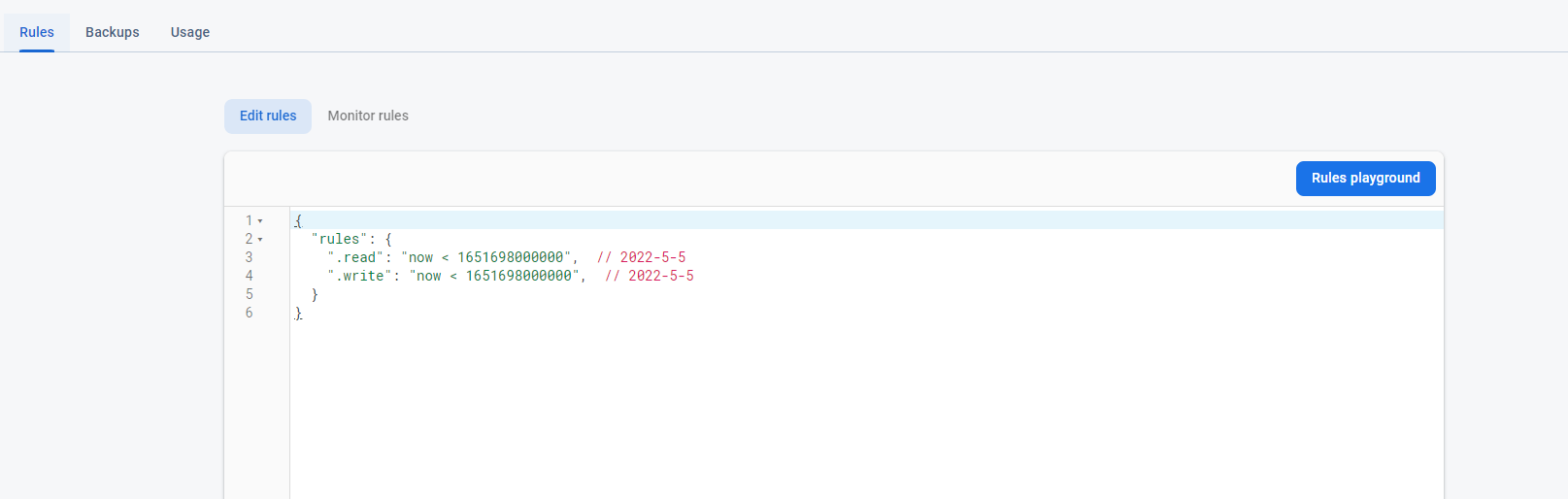


Рис №10. Создание БД

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

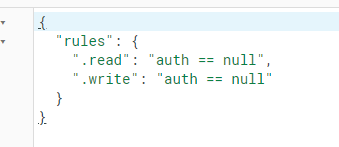


Рис №11. Создание БД

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:



Рис №12. БД

## **Разработка** **программного модуля**

Программа была разработана на Visual Studio 2019

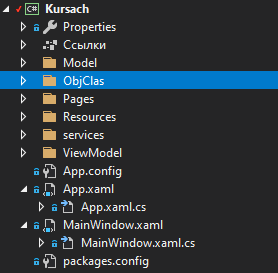
Шаги создания программы:

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.

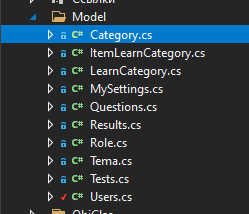
Структура проекта выглядит следующим образом:



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; }

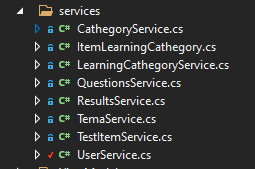
public string ImageUrl { get; set; }

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.

* 1. **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.

## **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 ()

# 4. **Руководство пользователя**

## 4.1 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.

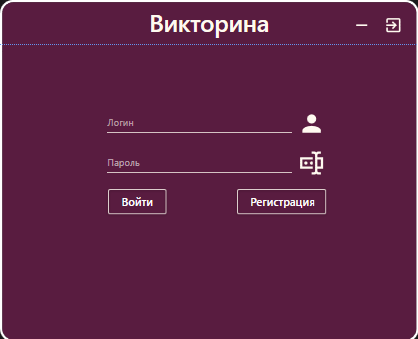


Рис №13. Страница авторизации



Рис №14. Страница пользовательского интерфейса



Рис №15. Страница админского интерфейса

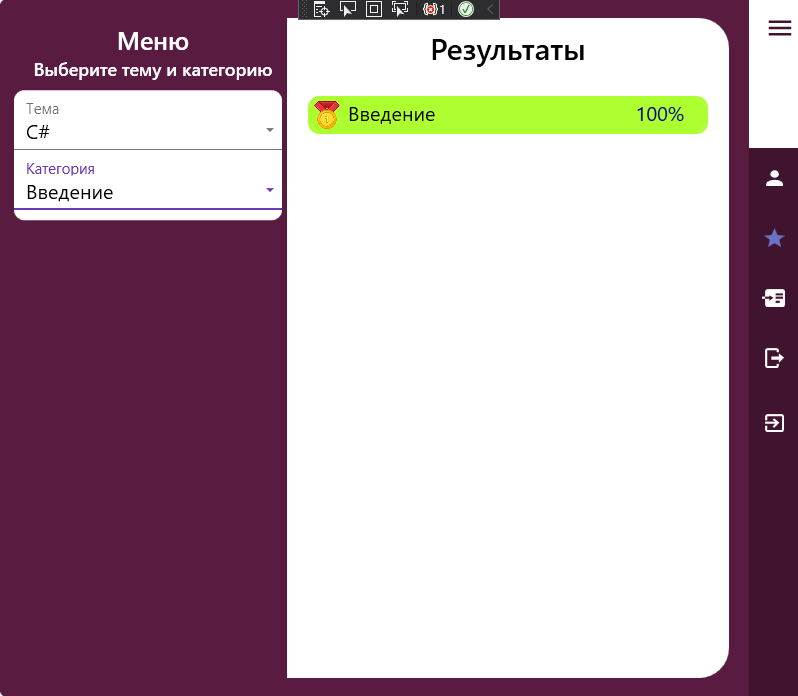


Рис №16. Страница просмотра результатов

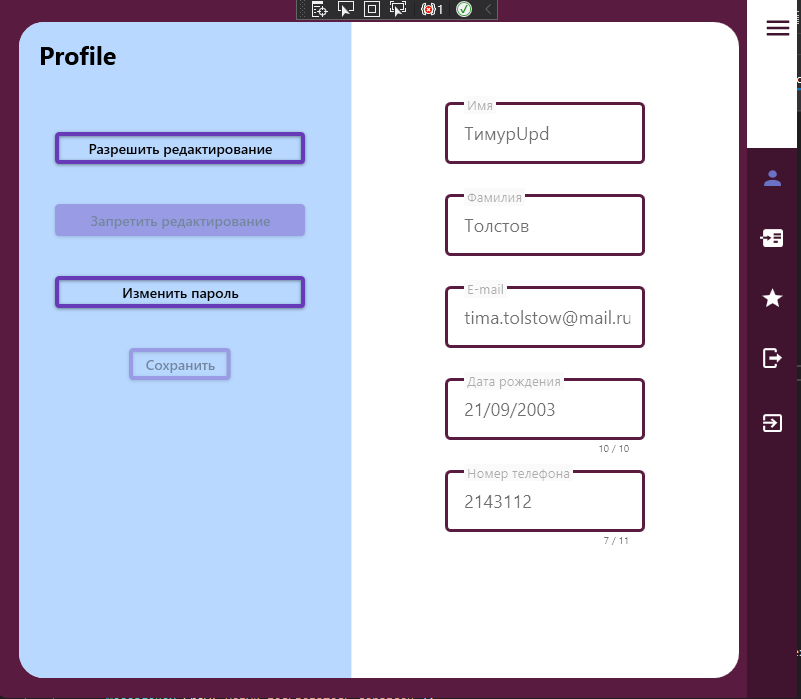


Рис №17. Страница профиля

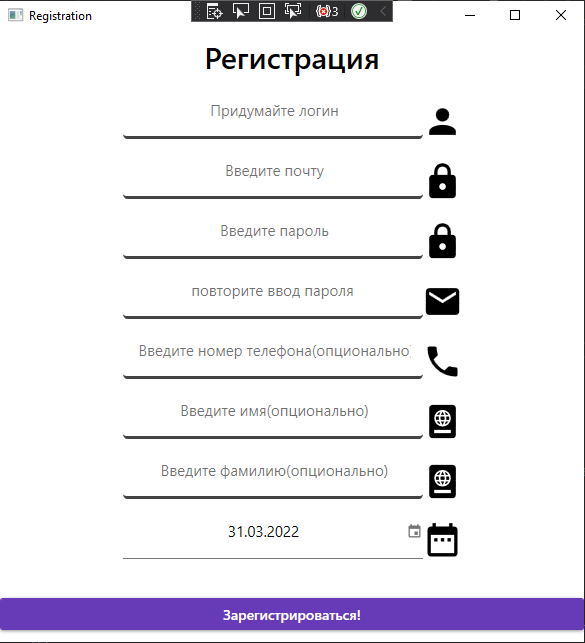


Рис №18. Окно регистрации

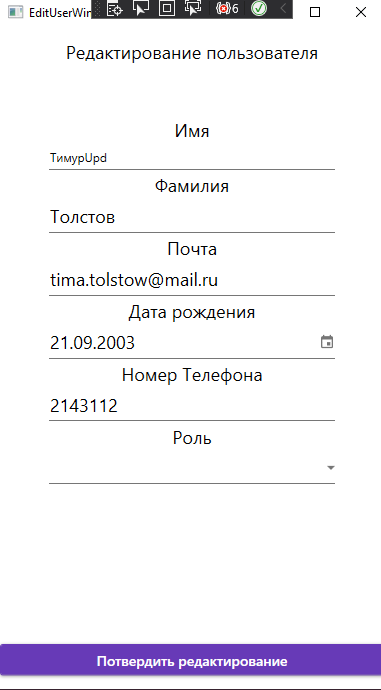


Рис №19. Окно редактирования пользователя

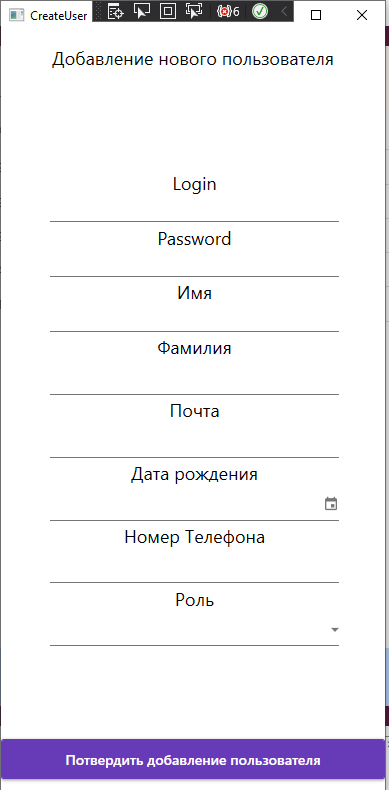


Рис №20. Окно добавления пользователя

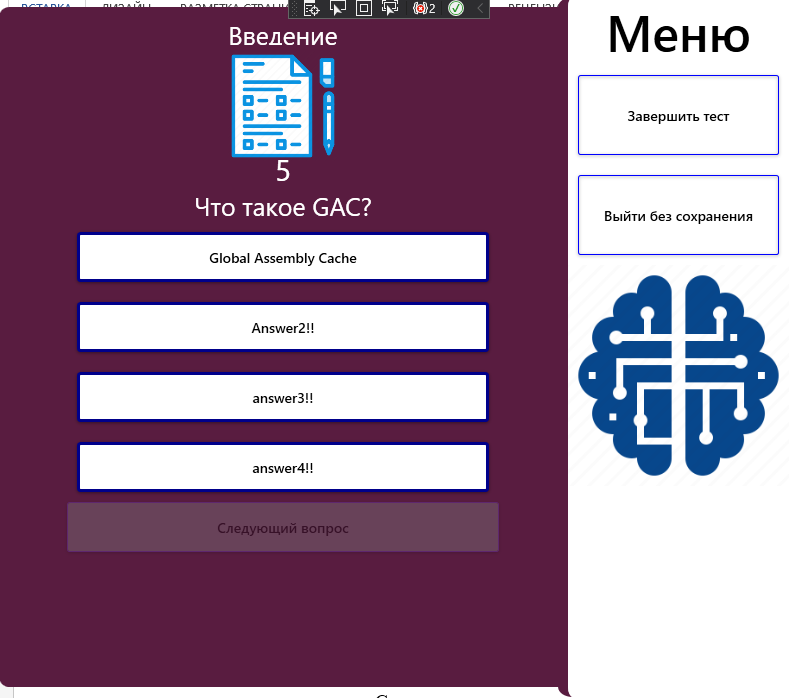


Рис №21. Страница прохождения теста

## Сообщения пользователю

* Сообщения при регистрации.

1. Успешная регистрация.

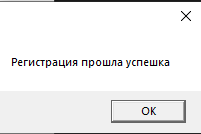


Рис №22. сообщение о регистрации

1. Некорректный ввод данных.

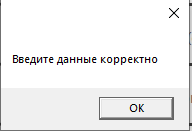


Рис №23. сообщение о не корректно введенных данных

1. Пользователь уже существует.

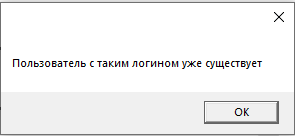


Рис №24. Сообщение о существование другого пользователя с таким логином.

* Сообщения после прохождения тестов.

1. Новый результат.

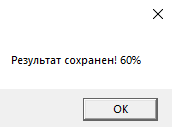


Рис №25. сообщение о результате тестирования

1. Обновление существующего.

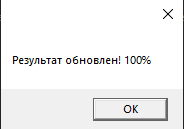


Рис №26. сообщение о обновлении результата.

1. Результат не сохранен.

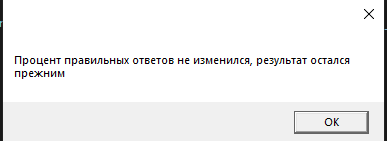


Рис №27. сообщение о не сохранённом результате тестирования

* Сообщения администратору.

1. При добавлении пользователя

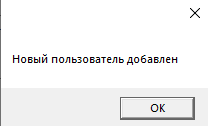


Рис №28. Сообщение при добавлении пользователя

1. При редактировании пользователя.

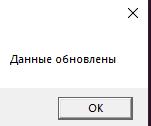


Рис №29. сообщение о обновлении данных пользователя

* Сообщения пользователю при изменении личных данных.

1. Изменение личных данных прошло успешно.

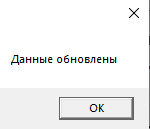


Рис №30. сообщение о обновлении данных пользователя

# 5. **Тестирование** **и отладка программного модуля**

Тестирование программного обеспечения — процесс исследования, испытания программного продукта, имеющий своей целью проверку соответствия между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов, выбранных определённым образом (ISO/IEC TR 19759:2005).

## 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. |

# 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.