**Введение**

Данное приложение реализует функциональные возможности проверки знаний студентов по языку программирования С#. Подобного рода приложения должны реализовывать возможность объективной проверки знаний студентов с автоматическим выставлением оценки и хранением всех результатов в базе данных. Приложение должно быть расширяемым и изменяемым: должна быть реализована возможность редактирования существующих данных, а так же возможность добавление новых вопросов и заданий, так как технологии, как и язык C#, постоянно меняются и улучшаются. Среда проверки знаний должна быть максимально простой и одновременно реализовывать как можно больше функционала, который облегчит работу и студента, и преподавателя. Оценка выставляется автоматически, с просчетом всех данных внутри системы. Необходимым так же является наличие разделения ролей на студентов и преподавателей, причем возможности у данных ролей должны быть разные: преподаватель способен добавлять вопросы, проверять правильность ответов, а так же получать информацию о выполненных тестах группы или целого потока; студент же должен иметь минимальный набор возможностей: прохождение тестов, а так же просмотр выполненных заданий и результатов своих работ.

**Обзор темы проекта и анализ требований**

Так как основной частью разработки данного приложения будет тестовая среда, то стоит произвести четкий анализ правил составления тестов, а так же их проверки. Данная процедура необходима для того, чтобы знания студентов проверялись максимально объективно, а так же, чтобы оценка знаний была максимально точной.

Тесты можно классифицировать по различным признакам:

* по целям — информационные, диагностические, обучающие, мотивационные, аттестационные;
* по процедуре создания — стандартизованные, не стандартизованные;
* по способу формирования заданий — детерминированные, стохастические, динамические;
* по форме заданий — закрытого типа, открытого типа, установление соответствия, упорядочивание последовательности;
* по наличию обратной связи — традиционные и адаптивные

Важно составлять тесты таким образом, чтобы свести к минимуму вероятность их угадывания студентом наугад.

В среднем, многие исследователи полагают, что угадывающий наугад тесты студент пользуется следующим алгоритмом:

* предпочитать более развернутые, более обоснованные ответы, которые учитывают частные случаи, и избегать кратких, менее развернутых. Разработчик теста знаний может этим воспользоваться и сделать верный ответ кратким;
* если ответы образуют последовательность (например, на вопрос о дате Куликовской битвы: (1) 1200 год, (2) 1240 год, (3) 1300 год, (4) 1380 год), респондент будет стремиться дать средний ответ, предполагая, что правда в золотой середине. И ошибется, потому что Куликовская битва была в 1380 году, а автор задания позаботился о том, чтобы поставить верный ответ с краю;
* выбирать более наукообразные ответы, содержащие малоизвестные или иностранные слова;
* выбирать ответы, напоминающие по стилю написания что-то очень знакомое, ассоциирующееся с известной частью предметной области.

Для противодействия угадывания в своих тестах автор использует 6 вариантов правильного ответа, при этом, только один является верным. Таким образом, вероятность «угадать» правильный ответ составляет 1/6. Угадывая данный тест математически только каждый 1 студент из 28 430 288 029 929 700 000 студентов решит тест на 100%.

Именно данную вероятность и призван увеличить описанный ранее алгоритм, используемый студентами при угадывании тестов.

Существует ряд эффективных рекомендаций к составлению компьютерных тестов:

* Все ответы к данному вопросу должны выглядеть правдоподобно, заставляя учащегося анализировать каждый вариант ответа и выявлять в нем неточность или ошибку.
* Там, где это возможно, стоит привести несколько истинных ответов, каждый из которых, являясь верным, в той или иной степени дополняет остальные правильные ответы.
* Правильное утверждение не должно быть полностью созвучно определению, данному в учебнике, чтобы в нем сразу не угадывался правильный ответ. Для этого же допускается приводить заведомо неверные ответы, созвучные приведенным в учебниках определениям.
* Вопросы по каждой теме стоит подбирать таким образом, чтобы они наиболее полно охватывали все разделы курса и позволяли контролировать как усвоение учащимися теоретических знаний, так и навыки применения этих знаний на практике.
* Выбор способа оформления заданий обусловлен как возможностями применяемых тестирующих программ (в случае применения для тестирования компьютерной техники), так и психологическими особенностями учащихся.
* Процесс создания вариантов тестов всегда должен включать опытную стадию, т.е. прежде чем предложить всему потоку, целесообразно дать тест для небольшой группы учащихся. Этот метод в сочетании с разбором ответов наиболее эффективно выявляет все ошибки, допущенные при составлении тестов.

Важно применить проанализированные требования к своей тестовой среды, чтобы приложение соответствовала объективной проверке знаний студентов.

**Анализ и проектирование архитектуры приложения**

Для полной картины стоит описать и изобразить основные концепции работы приложения.

Данное приложение будет содержать базу данных, которая будет хранить следующий данные:

* Вопросы, а так же всю информацию о них (уровень сложности, количество приносимых баллов, категория)
* Учетные записи
* Все необходимые данные о пользователях
* Результаты тестов
* Оценки

Учитывая все необходимые данные, схема базы данных может иметь следующий вид (Рис. 1):

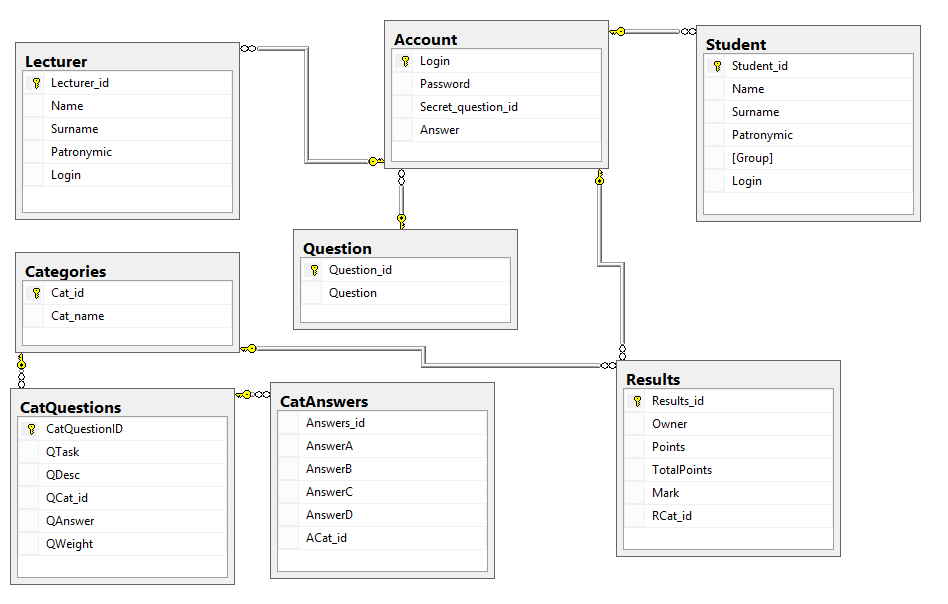


Рис. 1 – Диаграмма базы данных

Вся основная часть приложения будет реализована в виде WPF приложения, которая будет связана с базой данных через Entity Framework. Так же некоторые данные(например настройки пользователя) будут содержаться в локальных xml файлах, так как отводить специальные места в базе данных и производить к ней дополнительные обращения не имеет смысла. Схематически общие положения данного приложения можно изобразить так (Рис. 2):

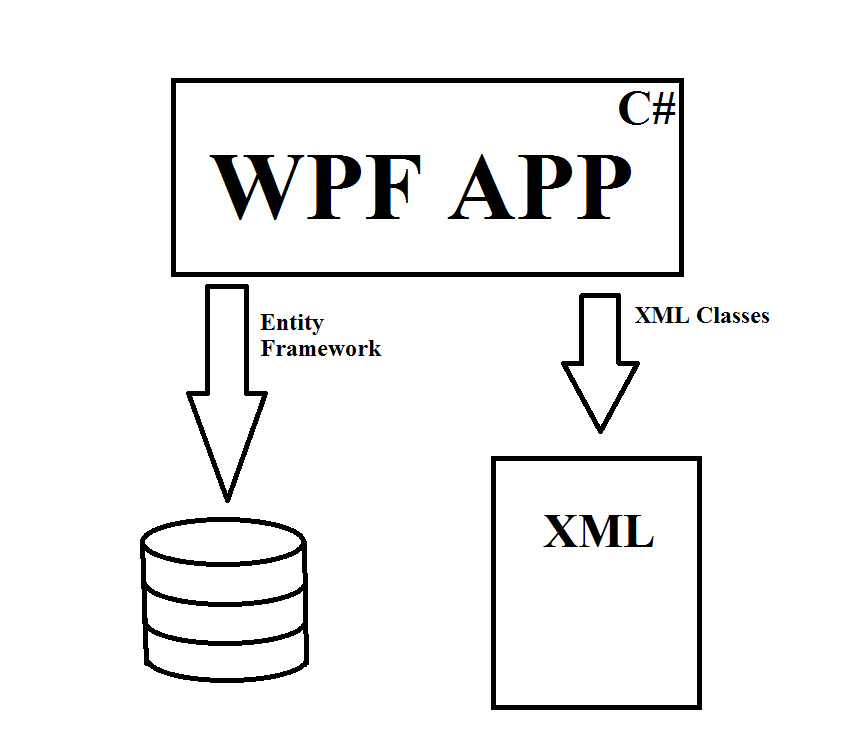


Рис. 2 – Схема работы приложения

Само WPF приложение будет реализовано с помощью паттерна MVVM(Model-View-ViewModel). Концепция данного паттерна подразумевает, что мы должны создать независимо View(Вид), который будет просто отвечать за вывод данных пользователю, а так же передачу информации в code behind. Независимая ViewModel, которая является связующим звеном между нашим View и моделью данных. ViewModel будет отвечать за события, а так же за посредничество между моделью и окнами. Так же существует Model, которая содержит всю информацию из базы данных, xml документов, а так же бизнес логику проекта. Стоит отметить, что между тремя компонентами приложения существует строгое построение связей: View не должно ничего знать о модели или ViewModel, но должно быть связано с ViewModel при помощи процедуры Binding, ViewModel ничего не знает о View, но влияет на ее с помощью изменения своих свойств, но прекрасно осведомлено о существовании Model и даже содержит в себе некоторые классы из модели, Model ничего не знает ни о View ни о ViewModel и лишь содержит в себе данные и методы обработки информации, а так же хранит информацию из базы данных. Схема данного паттерна может иметь вид (Рис. 3):

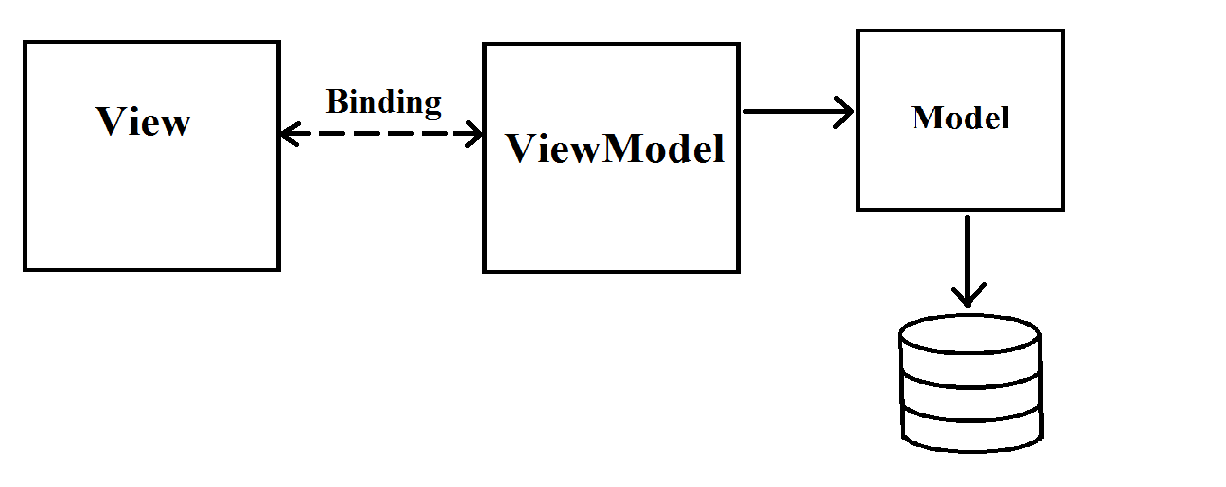


Рис. 3 – Схема паттерна MVVM

Стоит отметить, что данный паттерн желательно и даже нужно использовать по ряду причин:

* Позволяет легко расширять приложение
* Разбиение компонентов на отдельные части позволяет осуществлять разделение обязанностей по разработке приложения
* Так как Вид независим от code behind, то мы не зависим от дизайнера, который разрабатывает внешний вид приложения, а он не зависит от нас, что говорит о том, что приложение может разрабатываться максимально быстро и без помех
* Подобное разбиение компонентов позволяет вносить изменения в код и добавлять новую логику, практически не влияя на уже разработанную часть

Далее стоит описать каждый этап, который будет реализован в приложении, а так же все нюансы работы этапа.

Необходимым и важным в подобном приложении является наличие учетных записей для каждого отдельного пользователя, а, следовательно, надо сделать окно авторизации. Важными компонентами окна авторизации является такие элементы управления как: поля для ввода информации пользователя(логин, пароль), кнопка, подтверждающая введенные данные, кнопка регистрации учетной записи, кнопка типа «Забыли пароль?». Таким образом, окно регистрации может иметь следующий вид со следующими элементами управления:

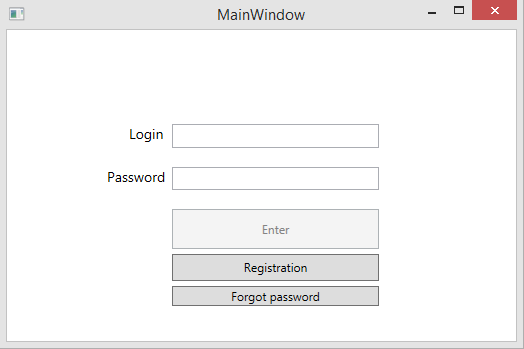


Рис. 4 – Окно авторизации

Наличие учетных записей так же подразумевает наличие возможности их создания, а значит, необходимым компонентом является окно регистрации учетной записи. Данное окно должно содержать все необходимые данные, которые после будут занесены в базу данных. Окно регистрации может иметь вид (Рис. 5):

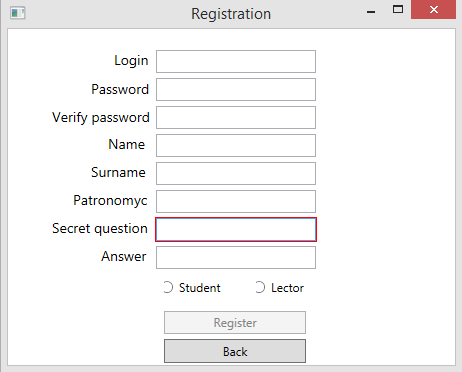


Рис. 5 – Окно регистрации

При авторизации под учетной записью студента, появляется окно, которое должно содержать формы для выбора теста, а так же просмотра результатов (Рис. 6):

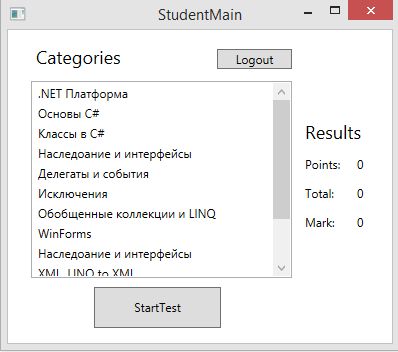


Рис. 6 – Окно управления студента

Далее следует создать окно для прохождения теста. Оно должно содержать: кнопку «Назад» (Вернуться к предыдущему вопросу), «Пропустить» (Пропуск вопроса), «Далее» (Засчитать выбранный ответ). Так же должна быть кнопка «Закончить», если пользователь желает покинуть данное окно и закончить тест. С учетом всех требований, окно прохождения теста может иметь вид:

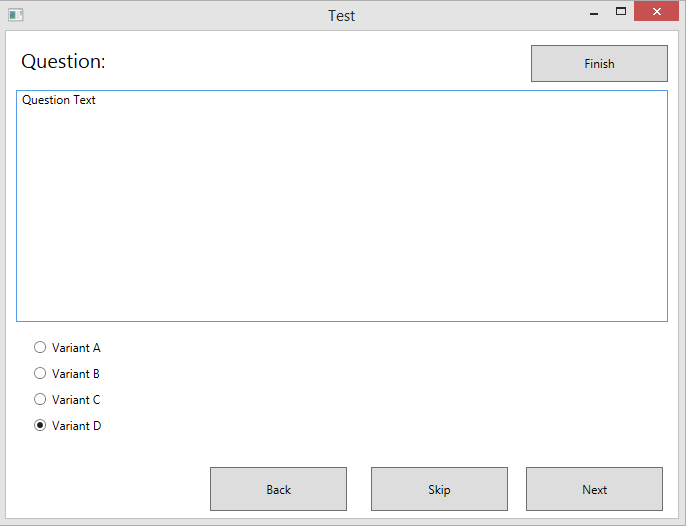


Рис. 7 – Окно прохождения теста

После прохождения теста будет выводиться MessageBox с текстом результата прохождения теста.

Далее, к данным окнам необходимо применить необходимое графическое оформление: изменить стили, подобрать цвета и изменить шрифты на свое усмотрение, учитывая все требования правильного дизайнерского оформления.

Вид окон и кнопок.

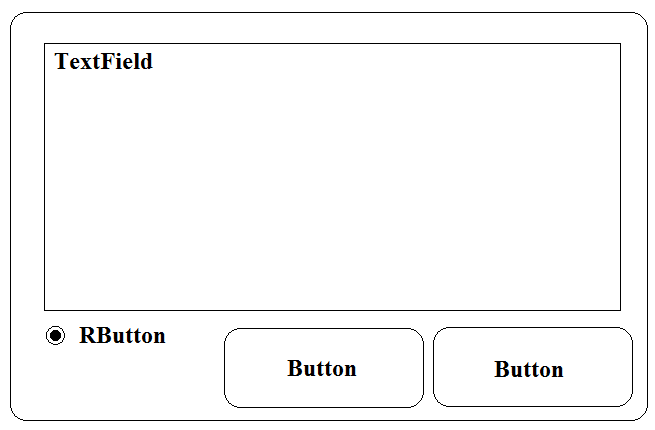


Рис. 8 – Вид окон

Для данных окон необходим не яркий и не вызывающий цвет, который не будет отвлекать пользователя от основных элементов интерфейса. Для данного приложения можно подобрать 3 основных цвета:

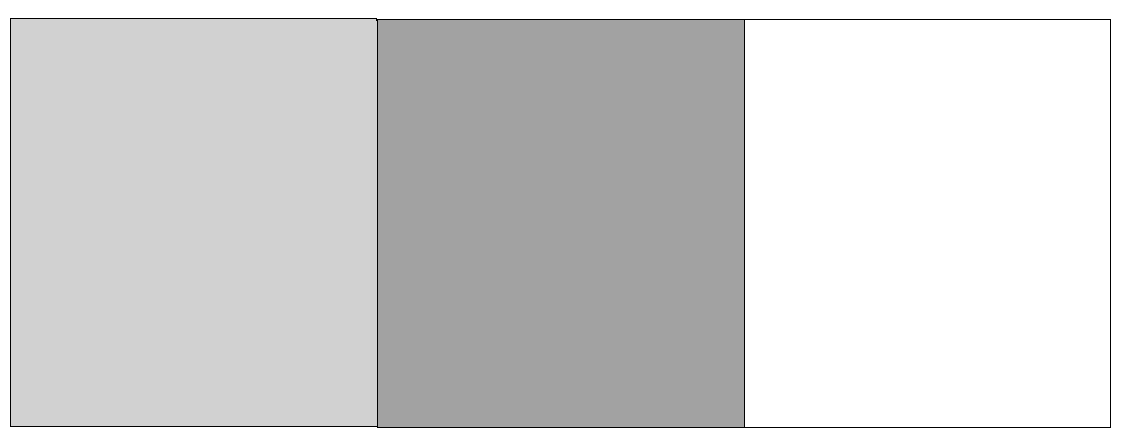


Рис. 9 – Основные цвета

Следуя цветовой раскраске, можем получить следующий вид окон:

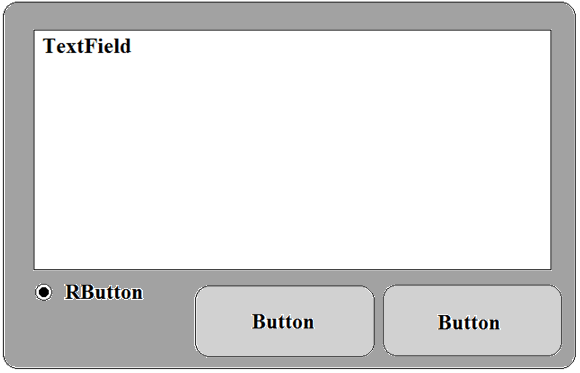


Рис. 10 – Закрашенное окно

Для данного приложения важен читабельный и различимый текст, следовательно, нужно подобрать шрифт, удовлетворяющий данное требование:

Georgia

Аа Ёё Лл Сс Чч Ээ Бб Жж Мм Тт Шш Юю Вв Зз Нн Уу Щщ Яя Гг Ии Оо Фф Ъъ Дд Йй Пп Хх Ыы Ее Кк Рр Цц Ьь

**Аа Ёё Лл Сс Чч Ээ Бб Жж Мм Тт Шш Юю Вв Зз Нн Уу Щщ Яя Гг Ии Оо Фф Ъъ Дд Йй Пп Хх Ыы Ее Кк Рр Цц Ьь**

*Аа Ёё Лл Сс Чч Ээ Бб Жж Мм Тт Шш Юю Вв Зз Нн Уу Щщ Яя Гг Ии Оо Фф Ъъ Дд Йй Пп Хх Ыы Ее Кк Рр Цц Ьь*

Аа Ёё Лл Сс Чч Ээ Бб Жж Мм Тт Шш Юю Вв Зз Нн Уу Щщ Яя Гг Ии Оо Фф Ъъ Дд Йй Пп Хх Ыы Ее Кк Рр Цц Ьь

<http://web.snauka.ru/issues/2014/02/31055> - Тесты как эффективный инструмент проверки знаний