Частное учреждение образования

«Колледж бизнеса и права»

тестирующая программа НА ОПРЕДЕЛЕНИЕ ТЕМПЕРАМЕНТА

Пояснительная записка

к курсовому проекту по дисциплине

«Конструирование программ и языки программирования»

КП Т.794005.401

Руководитель проекта (В.Ю. Михалевич )

Учащийся (Г.С. Выставкин )

2020

Содержание

Изм.

Лист

№ докум.

Подпись

Датаа

Лист

3

КП Т.794005.401 ПЗ

Разраб.

Выставкин Г.С.

Провер.

Михалевич В.Ю.

Реценз.

Н. Контр.

Утверд.

Лит.

Листов

КБП

62

у

[Введение 4](#_Toc43174936)

[1 Объектно-ориентированный анализ и проектирование системы 5](#_Toc43174937)

[1.1 Сущность задачи 5](#_Toc43174938)

[1.2 Проектирование модели 7](#_Toc43174939)

[2 Вычислительная система 13](#_Toc43174940)

[2.1 Требования к аппаратным и операционным ресурсам 13](#_Toc43174941)

[2.2 Инструменты разработки 13](#_Toc43174942)

[3 Проектирование задачи 16](#_Toc43174943)

[3.1 Требования к приложению 16](#_Toc43174944)

[3.2 Концептуальный прототип 16](#_Toc43174945)

[3.3 Организация данных 19](#_Toc43174946)

[3.4 Функции логическая и физическая организация 20](#_Toc43174947)

[3.5 Проектирование справочной системы 22](#_Toc43174948)

[4 Описание программного средства 23](#_Toc43174949)

[4.1 Общие сведения 23](#_Toc43174950)

[4.2 Функциональное назначение 23](#_Toc43174951)

[4.3 Входные и выходные данные 23](#_Toc43174952)

[5 Методика испытаний 25](#_Toc43174953)

[5.1 Технические требования 25](#_Toc43174954)

[5.2 Порядок проведения испытаний 25](#_Toc43174955)

[5.2.1 Функциональное тестирование 25](#_Toc43174956)

[5.2.2 Полное тестирование 28](#_Toc43174957)

[6 Применение 35](#_Toc43174958)

[6.1 Назначение программы 35](#_Toc43174959)

[6.2 Условие применения 35](#_Toc43174960)

[6.3 Справочная система 35](#_Toc43174961)

[Заключение 37](#_Toc43174962)

[Список информационных источников 38](#_Toc43174963)

[Приложение А. Текст программных модулей ...39](#_Toc43174964)

[Приложение Б. Формы выходных документов ………62](#_Toc43174965)

Тестирующая программа на определение темпермента

Тестирующая программа на определение темперамента

# Введение

Во время развития компьютерных технологий, хранение информации на бумаге становится все более и более невостребованным, так как информация на бумаге занимает много места, анализ этой информации требует большого количества усилий и времени, и поиск нужного документа также становится трудоемким.

С целью облегчения и повышения эффективности работы сотрудника библиотеки возникла необходимость в использовании современных компьютерных программ.

Целью курсового проекта является разработка программного средства тестирования на тип темперамента которая будет осуществлять авторизацию через базы данных, быстрый и простой доступ к тестам, выбор нужного теста, сохранение данных теста в файл и подробная информация о тестируемом.

Пояснительная записка к курсовому проекту состоит из шести разделов, содержащих необходимую информацию по организации эксплуатации программного средства. [1]

В первом разделе «Объектно-ориентированный анализ и проектирование системы» раскрывается организационная сущность задачи, описывается предметная область и круг задач, которые должны быть автоматизированы. Описываются новые возможности программного средства, а также ее отличия от предыдущих версий, перечисляются основные функции программного средств. Строится информационная модель, отражающая сущности задачи, их свойства и взаимосвязи.

Во втором разделе «Вычислительная система» перечисляются требования к аппаратному обеспечению и конфигурации компьютера, проводится характеристика операционной системы, обоснование выбранной среды для разработки приложения.

В третьем разделе «Проектирование задачи» перечисляются требования к программному средству, концептуальный прототип, логическую и физическую структуры данных в контексте среды разработки, структуру и описание функций пользователя в рамках среды разрабатываемого программного средства, функции и элементы управления, проектирование справочной системы программного средства.

В четвертом разделе «Описание программного средства» приводятся общие сведения о программном средстве, ее функциональное назначение, структура входных и выходных данных.

В пятом разделе «Методика испытаний» описываются требования к техническим средствам для проведения испытаний, требования к характеристикам программного средства применительно к условиям эксплуатации, требования к информационной и программной совместимости. Представляются результаты функционального тестирования.

Шестой раздел «Применение» предназначен для описания сведений о назначении программного средства и области его применения классе решаемых задач, ограничениях, накладываемых на область применения, требования к техническим и программным ресурсам. В этом разделе приводится структура справочной системы, а также методика ее использования.

В заключении проанализировано созданное программное приложение, определена степень соответствия поставленной задачи и выполненной работы.

Приложение А содержит текст программного средства.

Приложение Б содержит выходные документы

Графическая часть содержит:

– диаграмму вариантов использования;

– диаграмму классов;

– диаграмму последовательности;

– диаграмму деятельности;

– диаграмму компонентов.

# 1 Объектно-ориентированный анализ и проектирование системы

## 1.1 Сущность задачи

Предметной областью курсового проекта является тест темперамента.

Тест – способ изучения глубинных процессов деятельности системы, посредством помещения системы в разные ситуации и отслеживание доступных наблюдению изменений в ней.

Тесты бывают: на интеллект, проверка способностей, достижений, личностные тесты, управленческих качеств и т.д.

Тестирование как метод психологического исследования личности осуществляется посредством тестов. Применяя тесты, можно получить достаточно точную качественную или количественную характеристику изучаемого явления. От других методов сбора информации в психологии тест выгодно отличается тем, что в основе его лежит четкая процедура сбора и обработки первичных данных. К разнообразным тестам прибегают в том случае, если требуется изучить или сравнить психологию различных групп людей, дать дифференцированные и сопоставимые оценки. Психологические тесты нужны для того, чтобы узнать что-либо о личности, которая будет их проходить. Признаками таких тестов являются: кратковременность, стандартизованность, не требует сложных технических приспособлений, результаты – число, статистическая обработка.

Психологический тест – стандартизированная методика, направленная на измерение индивидуальных свойств и качеств респондента (психофизиологических и личностных характеристик, способностей, знаний и навыков, состояний).

Методика оценки может называться тестом при одновременном наличии у нее следующих признаков:

– стандартизированный набор вопросов или заданий иного типа;

– одна или несколько измерительных шкал, позволяющих выразить результаты количественно;

– связь каждого ответа на каждое задание с одной или несколькими измерительными шкалами (наличие «ключей к тесту»)

– стандартизированная процедура проведения, включающая однозначную (стандартную) инструкцию для тестируемого, правила использования вспомогательной информации, правила завершения или приостановки тестирования и т.п.;

– возможность автоматической (без участия человека) обработки результатов, то есть формализованная процедура подсчета баллов по шкалам с помощью весовых коэффициентов (ключей);

– тестовые нормы – фиксированные границы перевода тестовых баллов в оценочные категории;

– формализованная модель интерпретации результатов и/или рекомендации по принятию тех или иных решений, связанные с определенными интервалами значений на шкале (шкалах) и сочетаниями значений шкал (при наличии двух шкал и более);

– направленность на индивидуальную количественную оценку какой-либо характеристики одного человека (а не группы, коллектива и т.п.).

Темперамент – это врожденная характеристика, которая зависит от свойств нервной системы, таких как сила, подвижность и уравновешенность. Он неизменный и остается в течение жизни. Характер, в свою очередь, это приобретенные качества человека, зависящие от окружения, воспитания, социального и культурного окружения, национальности, религии и т.д. Отсюда можно понять, что характер − характеристика не постоянная и может меняться в зависимости от внешних факторов. Стоит отметить, что не бывает хороших или плохих типов темперамента. У каждого есть свои отрицательные и положительные стороны, зная которые можно определить, как их правильно использовать.

Всего существует 4 типа темперамента: сангвиник, флегматик, холерик, меланхолик. Но в природе крайне редко встречаются люди с одним каким‐то типом, обычно это смесь нескольких в разных соотношениях. Различия видов темперамента находятся в таблице 1.

Таблица 1 Различия видов темперамента

|  |  |
| --- | --- |
| Типы по Гиппократу-Галену | Черты темперамента |
| Холерик | Импульсивность, агрессия |
| Флегматик | Социально отстраненный, погруженный в себя |
| Меланхолик | Грустный, боязливый, подавленный, слабый |
| Сангвиник | Подвижный, социальный, уверенный в себе |

Психологический портрет личности всегда был и будет главнейшим интересом психологов, ведь с каждым разом в нем открываются новые грани, которые требуют особого внимания. Каждый человек хоть раз в жизни пытался узнать тип своего темперамента посредством тестирования. В настоящее время опубликовано более 4 тысяч различных тестов, но одними из самых распространенных являются методики определения темперамента Г. Айзенка и А. Белова.

Г. Айзенк большое значение придавал индивидуальным различиям. Значит можно сделать вывод, что никакой из комбинаций данных типов личности не может быть отдано предпочтение. Более того, категории, которые выделял Г. Айзенк, полностью соответствуют 4 основным типам темперамента, а именно: холерик, сангвиник, флегматик, меланхолик. Рассматривая практическое применение тестирования по определению темперамента по Г. Айзенку, нужно отметить, что оно достаточно просто в восприятии. В данном тесте содержится 57 вопросов, которые без труда помогут определить тип темперамента. Рассматривая практическое применение тестирования по определению темперамента по Г. Айзенку, нужно отметить, что оно достаточно просто в восприятии. В данном тесте содержится 57 вопросов, которые без труда помогут определить тип темперамента. По нашим данным максимальная продолжительность ответов на все вопросы 25 минут. Подсчет результатов вызвал некоторые затруднения, что и привело к такому временному результату.

Методика тестирования темперамента A. Белова, которую автор называл «Формула темперамента А. Белова», является более современной, данная методика служит также, как и вышеупомянутая, для определения преобладающего типа темперамента, но и так же для выявления представленности в нем свойств других типов. Но, в данном тестировании основным «выходным» элементом выступает процентное соотношение типов темпераментов в виде формулы. Так же, как и методика тестирования Г. Айзенка, она является легко воспринимаемой. Но, в тестировании по А. Белову, есть один большой плюс - все вопросы данного теста разбиты на 4 блока. Данный тест более емкий, содержит в себе большее количество вопросов 80, соответственно по 20 в каждой группе. Удобство разделения вопросов заключается в том, что даже без расчетов, по отмеченным результатам сразу видео, к какому типу темперамента относится тестируемый. Такого эффекта в процедуре тестирования по Г. Айзенку нет. Более того, такой способ организации теста позволяет сократить время его прохождения. В нашем эксперимаксимальный результат затраченного времени на тест по А. Белову составил 18 минут.  
Ниже представлены результаты в виде рисунка 1.

Рисунок 1 – Результаты диагностики по А.Белову и Г.Айзенку

Важность знания своего темперамента – это умение находить общий язык с разными людьми. Также это является основой любого общения, ведь не всегда нам удается заключать сделки, поддерживать разговор и получать желаемое от разных людей. Именно знания о темпераментах помогают подобрать тот самый ключик при общении.

В программном средстве предусмотрено разбиение на пользователя и администратора. У пользователя в основных функциях будет выбор теста, прохождение теста, просмотр статистики теста. У администратора основные функции: добавление, редактирование, удаление тестов, просмотр статистики между всеми пользователями.

На основе изученной предметной области приложение должно выполнять следующие функции:

– отображение тестов

– выбор теста

– прохождение теста

– отображение результатов теста

– отображение статистики теста

Аналогами приложения являются такие: тест на темперамент от сайта testometrika.com, wikigrowth.ru и temperamenttest.org и др.

В своей сути данные программные средства имеют одинаковый функционал, но в каждой также имеется своя «изюминка». Удобный интерфейс,

Создаваемое программное средство имеет следующие отличия: раздел тестов, авторизация, полная отчетность и анализ темперамента после прохождения теста.

## 1.2 Проектирование модели

ER-модель – модель данных, позволяющая описывать концептуальные схемы предметной области. ER-модель используется при высокоуровневом (концептуальном) проектировании баз данных. С её помощью можно выделить ключевые сущности и обозначить связи, которые могут устанавливаться между этими сущностями.

Во время проектирования баз данных происходит преобразование ER-модели в конкретную схему базы данных на основе выбранной модели данных (реляционной, объектной, сетевой или др.).[2]

ER-модель представляет собой формальную конструкцию, которая сама по себе не предписывает никаких графических средств её визуализации. В качестве стандартной графической нотации, с помощью которой можно визуализировать ER-модель, была предложена диаграмма «сущность-связь».

ERD-диаграмма по Баркеру располагается на рисунке 2.

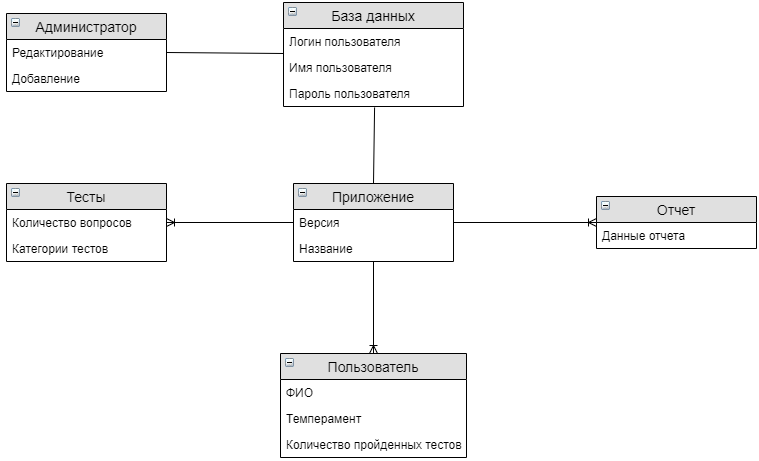


Рисунок – ERD-диаграмма по Баркеру

Сущность «Пользователь» состоит из атрибутов:

– фио;

– темперамент;

– количество пройденных тестов.

Сущность «Приложение» состоит из атрибутов:

– название;

– версия.

Сущность «Отчёт» состоит из атрибута: данные отчёта.

Сущность «Тесты» состоит из атрибутов:

– категории тестов;

– количество тестов.

Сущность «База данных» состоит из атрибутов:

– логин пользователя;

– имя пользователя;

– пароль пользователя.

Сущность «Администратор» состоит из атрибутов:

– добавление;

– редактирование.

Диаграмма вариантов использования – диаграмма, отражающая отношения между актерами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Прецедент – возможность моделируемой системы (часть её функциональности), благодаря которой пользователь может получить конкретный, измеримый и нужный ему результат. Прецедент соответствует отдельному сервису системы, определяет один из вариантов её использования и описывает типичный способ взаимодействия пользователя с системой. Варианты использования обычно применяются для спецификации внешних требований к системе.

Основное назначение диаграммы – описание функциональности и поведения, позволяющее заказчику, конечному пользователю и разработчику совместно обсуждать проектируемую или существующую систему.

Отношение расширения – определяет взаимосвязь экземпляров отдельного варианта использования с более общим вариантом, свойства которого определяются на основе способа совместного объединения данных экземпляров; отношение является направленным и указывает, что применительно к отдельным вариантам использования выполнены конкретные условия и что один из вариантов может присоединять к своему поведению некоторое дополнительное поведение, определенное для другого варианта; отношение расширения между вариантами использования обозначается пунктирной линией со стрелкой, направленной от варианта использования, который является расширением .

Отношение включения –указывает, что некоторое заданное поведение для одного варианта использования включается в качестве составного компонента в последовательность поведения другого варианта использования. Данное отношение является направленным бинарным отношением, при котором, когда экземпляр первого варианта использования в процессе своего выполнения достигает точки включения в последовательность поведения экземпляра второго варианта использования, он выполняет последовательность действий, определяющий поведение второго варианта, после чего продолжает выполнение действий своего поведения. Один вариант может быть включен в несколько других вариантов, а также включать в себя другие варианты.

На данной диаграмме имеется два «актёра» – пользователь и администратор. Пользователь имеет такие варианты использования как:

– регистрация;

– авторизация;

– просмотр тестов;

– выбор теста;

– прохождение теста;

– просмотр количества вопросов;

– переход на следующий вопрос.

Администратор имеет такие варианты использования как

– авторизация

– просмотр списка тестов

– просмотр списка пользователей

– просмотр результатов тестов

– добавление теста

– редактирование теста

– удаление теста

– удаление пользователя

– просмотр информации пользователя

– удаление пользователя.

В основном при создании диаграммы рассматривались такие связи как «расширения», т.е. при выборе и выполнении одного определенного варианта, у пользователя или администратора расширяется диапазон выбора различных функций (добавляются дополнительные поведения).

Диаграмма вариантов использования располагается на листе 1 в графической части.

Диаграмма классов – структурная диаграмма языка моделирования UML, демонстрирующая общую структуру иерархии классов системы, их коопераций, атрибутов (полей), методов, интерфейсов и взаимосвязей между ними. Широко применяется не только для документирования и визуализации, но также для конструирования посредством прямого или обратного проектирования.

Целью создания диаграммы классов является графическое представление статической структуры декларативных элементов системы (классов, типов и т. п.) Она содержит в себе также некоторые элементы поведения (операции), однако их динамика должна быть отражена на диаграммах других видов (диаграммах коммуникации, диаграммах состояний). Для удобства восприятия диаграмму классов можно также дополнить представлением пакетов, включая вложенные.

Отношение агрегации имеет место между несколькими классами в том случае, если один из классов представляет собой некоторую сущность, включающую в себя в качестве составных частей другие сущности. Отношение композиции служит для выделения специальной формы отношения, при которой составляющие части в некотором смысле находятся внутри целого. Подробное описание классов приложения находится в таблице 2.

Таблица 2. Описание классов приложения

|  |  |  |
| --- | --- | --- |
| Название класса | Поля класса | Функции класса |
| Пользователь | Name, Surname, GroupNumber | Register (Name, Surname, GroupNumber) |
| Приложение | – | Login (Name, Surname, GroupNumber) |
| Администратор | Name, Surname, Password | Editing(), Remove(), Adding() |
| Тест | NameTest, CategoryTest, CountQuest | Questions(), TestSelection(), Statictics() |
| Вывод информации | – | ShowTests(), ShowQuest(), ShowStatistics() |

Классы «Приложение», «Тест» и «Администратор» соединены между собой отношением композиции. «Приложение» и «Пользователь» – отношением агрегации. «Тест» и «Вывод информации» – отношением ассоциации. Обусловлено это тем, что отношение композиции связывает классы, которые обязательны и являются основными компонентами программного средства. В то время как те классы, которые связаны отношением агрегации, наоборот, являются вторыми по приоритету (необязательны). Классы связанные отношением ассоциации показывает, что объекты одной сущности (класса) связаны с объектами другой сущности таким образом, что можно перемещаться от объектов одного класса к другому. Диаграмма классов располагается на листе 2 в графической части.

Диаграмма последовательности – диаграмма, на которой для некоторого набора объектов на единой временной оси показан жизненный цикл какого-либо определённого объекта (создание-деятельность-уничтожение некой сущности) и взаимодействие актеров (действующих лиц) в рамках какого-либо определённого прецедента (отправка запросов и получение ответов). Используется в языке UML.

Основными элементами диаграммы последовательности являются обозначения объектов (прямоугольники с названиями объектов), вертикальные «линии жизни», отображающие течение времени, прямоугольники, отражающие деятельность объекта или исполнение им определенной функции (прямоугольники на пунктирной «линии жизни»), и стрелки, показывающие обмен сигналами или сообщениями между объектами.

На данной диаграмме показана последовательность действий от регистрации или авторизации, до выхода из программного средства. Также показан специальный блок действий, который показывает альтернативные варианты поведения в программном средстве. В альтернативном блоке показано, как взаимодействует между собой приложение, база данных и функции приложения. Таким образом можно просмотреть, как поведет себя программное средство, если пользователь введёт неверные данные при регистрации или получит ошибку при начале теста.

Диаграмма последовательности располагается на листе 3 в графической части.

Диаграмма деятельности – UML диаграмма, на которой показаны действия, состояния которых описано на диаграмме состояний. Под деятельностью понимается спецификация исполняемого поведения в виде координированного последовательного и параллельного выполнения подчинённых элементов – вложенных видов деятельности и отдельных действий, соединённых между собой потоками, которые идут от выходов одного узла ко входам другого. Диаграммы деятельности используются при моделировании бизнес-процессов, технологических процессов, последовательных и параллельных вычислений. В диаграмме присутствует начало и конец какого-либо действия. Отмечается начало действия – закрашенным черным кружком и на границе кружка – черная линия. Конец действия – чуть более уменьшенный черный кружок и с небольшим отступом на границах – красная линия.

На данной диаграмме показан детальный процесс прохождения теста пользователем в программном средстве. От начала – запуска приложения и до конца – сохранения результатов.

Диаграмма деятельности располагается на листе 4 в графической части.

Диаграмма компонентов –элемент языка моделирования UML, статическая структурная диаграмма, которая показывает разбиение программной системы на структурные компоненты и связи (зависимости) между компонентами. В качестве физических компонентов могут выступать файлы, библиотеки, модули, исполняемые файлы, пакеты.

С помощью диаграммы компонентов представляются инкапсулированные классы вместе с их интерфейсными оболочками, портами и внутренними структурами (которые тоже могут состоять из компонентов и коннекторов).

Компоненты связываются через зависимости, когда соединяется требуемый интерфейс одного компонента с имеющимся интерфейсом другого компонента. Таким образом иллюстрируются отношения клиент-источник между двумя компонентами.

Зависимость показывает, что один компонент предоставляет сервис, необходимый другому компоненту. Зависимость изображается стрелкой от интерфейса или порта клиента к импортируемому интерфейсу.

Когда диаграмма компонентов используется, чтобы показать внутреннюю структуру компонентов, предоставляемый и требуемый интерфейсы составного компонента могут делегироваться в соответствующие интерфейсы внутренних компонентов.

Делегация показывает связь внешнего контракта компонента с внутренней реализацией этого поведения внутренними компонентами.

Диаграмма компонентов –элемент языка моделирования UML, статическая структурная диаграмма, которая показывает разбиение программной системы на структурные компоненты и связи (зависимости) между компонентами. В качестве физических компонентов могут выступать файлы, библиотеки, модули, исполняемые файлы, пакеты.

С помощью диаграммы компонентов представляются инкапсулированные классы вместе с их интерфейсными оболочками, портами и внутренними структурами (которые тоже могут состоять из компонентов и коннекторов).

Компоненты связываются через зависимости, когда соединяется требуемый интерфейс одного компонента с имеющимся интерфейсом другого компонента. Таким образом иллюстрируются отношения клиент-источник между двумя компонентами.

Зависимость показывает, что один компонент предоставляет сервис, необходимый другому компоненту. Зависимость изображается стрелкой от интерфейса или порта клиента к импортируемому интерфейсу.

Когда диаграмма компонентов используется, чтобы показать внутреннюю структуру компонентов, предоставляемый и требуемый интерфейсы составного компонента, могут делегироваться в соответствующие интерфейсы внутренних компонентов.

Делегация показывает связь внешнего контракта компонента с внутренней реализацией этого поведения внутренними компонентами.

На данной диаграмме показан состав программных компонентов, в роли которых может выступать исходный, бинарный и исполняемый код, а также установить зависимости между ними. На диаграмме представлена папка с файлами программного средства, папка включает такие файлы:

– «AdminPanel.cs» – файл админ панели;

– «AizenkTest.cs» – файл теста по Айзенку;

– «BelovTest.cs» – файл теста по Белову;

– «Database.cs» – файл подключения к базе данных;

– «HelpSystem.cs» – файл справочной системы;

– «MainAfterAutoriz.cs» – файл главного окна программного средства;

– «MainWindow.cs» – файл авторизации пользователя;

– «RegisterWindow.cs» – файл регистрации пользователя;

– «Users.mdf» – файл базы данных хранящий информацию пользователя;

– «Autoriz.mdf» - файл базы данных хранящий данные пользователя;

– «ResultWindow.cs» – файл результата теста.

– «KursachWPF.sln» – структурированный файл программного средства;

– «KursachWPF.exe» – исполняемый файл программного средства.

Диаграмма компонентов располагается на листе 5 в графической части.

# 2 Вычислительная система

## 2.1 Требования к аппаратным и операционным ресурсам

Данное программное средство не должно быть требовательно к ресурсам персонального компьютера. Приведенных ниже минимальных системных требований должно быть достаточно для функционирования программного средства:

**–** операционная система Windows 7 64-bit и выше;

**–** процессор Intel Core i3-2365M CPU @ 1.40GHz, AMD Ryzen 3 2200G

**–** ОЗУ от 512 МБ;

**–** видеокарта IntelHDGraphics 3000, AMD Vega 8;

**–** места на HDD 100 МБ;

**–** мышь, клавиатура, touchpad.

Приведенных ниже оптимальных системных требований должно быть достаточно для хорошего функционирования программного средства:

– операционная система Windows 7/8/10;

– процессор Intel Core i3 – 2365M CPU @ 2.30GHz;

– ОЗУ от 3000 МБ;

– видеокарта IntelHDGraphics 3000;

– места на HDD 100 МБ;

**–** мышь, клавиатура, touchpad.

## 2.2 Инструменты разработки

Инструментами разработки являются:

**–** операционная система Windows 10;

**–** язык программирования C#;

**–** система для построения приложений WPF;

**–** средство построения диаграмм Draw.io;

**–** среда разработки Visual Studio Professional 2019;

**–** язык структурированных запросов SQL SMS;

**–** текстовый редактор Microsoft Word 2016

Операционная система Windows 10 является одной из самых надежных операционных систем Windows. Она имеет множество удобных приложений и функций, которых не было в предшественниках. Windows 10 занимает первое место в мире среди операционных систем, используемых для доступа к сети Интернет, опередив в апреле 2017 года предыдущего лидера **–** Windows 7 [7].

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML. Данный язык использует объектно-ориентированный подход к программированию во всем. Это означает, что нужно будет описывать абстрактные конструкции на основе предметной области, а потом реализовывать между ними взаимодействие. Также в языке присутствует обилие синтаксического «сахара», который делает тяжелую жизнь капельку слаще. Вместо того, чтобы писать кучу строк кода, можно просто использовать готовую конструкцию, а компилятор сделает за тебя всю «грязную» работу. Но некоторые такие конструкции являются не самыми оптимальными с точки зрения производительности. Но все это перекрывается за счет удобочитаемости кода и высокой скоростью разработки. [3]

Windows Presentation Foundation (WPF) **–** это платформа пользовательского интерфейса для создания клиентских приложений для настольных систем. Платформа разработки WPF поддерживает широкий набор компонентов для разработки приложений, включая модель приложения, ресурсы, элементы управления, графику, макет, привязки данных, документы и безопасность. Также выбор был сделан исходя из следующих приемуществ:

**–** независимость от разрешающей способности, аппаратное ускорение, поддержку мультисенсорного ввода.

**–** стили, триггеры, шаблоны, анимацию, аудио и видео, команды, мощную систему привязки, маршрутизируемые события.

**–** продвинутые возможности для рисования.

Кроме того, идеи, принципы и философия, которые положены в основу данной технологии, мне очень удобны, импонируют и симпатичны.

Draw.io **–** инструмент, который позволяет создавать блок-схемы, сетевые диаграммы, интеллект-карты, отношения сущностей, программные блоки, UML, макеты и т. д. Богатая функциональность Draw.io позволяет пользователям отслеживать и восстанавливать изменения, импортировать и экспортировать в PDF, PNG, XML, VSDX, HTML, а также автоматически публиковать и делиться работами.

С помощью сервиса можно создавать:

**–** графики;

**–** блок-схемы;

**–** диаграммы;

**–** ментальные карты;

**–** макеты.

Особенности:

**–** большой набор шаблонов;

**–** быстрое создание блоков

**–** интуитивно понятный интерфейс;

**–** экспорт в форматы JPG, PNG, SVG, VDSX;

**–** совместная работа;

**–** русскоязычный интерфейс;

**–** возможность скачать в виде программного средства для компьютера.

Microsoft Visual Studio **–** линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии WPF, а также веб-сайты, веб-приложения, веб-службы. Visual Studio включает в себя редактор исходного кода с возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. [8]

SQL **–** декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных. Является, прежде всего, информационно-логическим языком, предназначенным для описания, изменения и извлечения данных, хранимых в реляционных базах данных. SQL считается языком программирования, в общем случае не является тьюринг-полным, но вместе с тем стандарт языка спецификацией SQL/PSM предусматривает возможность его процедурных расширений. При всех своих изменениях SQL остаётся самым распространённым лингвистическим средством для взаимодействия прикладного программного обеспечения с базами данных. В то же время современные СУБД, а также информационные системы, использующие СУБД, предоставляют пользователю развитые средства визуального построения запросов. [6]

Microsoft Word **–** текстовый процессор, предназначенный для создания, просмотра и редактирования текстовых документов, с локальным применением простейших форм таблично-матричных алгоритмов. Выпускается корпорацией Microsoft в составе пакета Microsoft Office. Microsoft Word является наиболее популярным из используемых в данный момент текстовых процессоров, что сделало его бинарный формат документа стандартом де-факто, и многие конкурирующие программы имеют поддержку совместимости с данным форматом. Формат документа разных версий Word меняется, различия бывают довольно тонкими. Форматирование, нормально выглядящее в последней версии, может не отображаться в старых версиях программы, однако есть ограниченная возможность сохранения документа с потерей части форматирования для открытия в старых версиях продукта. Последняя версия MS Word 2007 «использует по умолчанию» формат, основанный на XML. Ранее большая часть информации, нужной для работы с данным форматом, добывалась посредством обратного инжиниринга, поскольку основная её часть отсутствовала в открытом доступе или была доступна лишь ограниченному числу партнёров и контролирующих организаций.

# 3 Проектирование задачи

## 3.1 Требования к приложению

Интерфейс должен быть спроектирован с учетом разных экранных разрешений. Ни для одного сценария работы с приложением кнопки доступа к основным функциям не должны оказаться вне видимой области окна. Для размещенных вне видимой области элементов управления должна быть предусмотрена возможность прокрутки; для всех допускающих прокрутку областей сам факт возможности прокрутки и метод прокрутки должен быть интуитивно понятен пользователю. Несложное управление и взаимодействие с пользователем.

Состав интерфейса:

– кнопка закрытия программного средства в верхнем правом углу программного средства;

– кнопка выбора теста находится в главном меню по центру интерфейса программного средства;

– кнопка выхода из теста находится в нижнем левом углу программного средства.

Накладываемые ограничения:

– кнопки выбора теста располагается в главном меню по центру интерфейса программного средства;

– кнопка выхода из располагается в нижнем левом углу программного средства;

– кнопка закрытия программного средства располагается в верхнем правом углу программного средства.

Средства защиты для данного программного средства:

Для входа в приложение требуется ввести уже существующие Логин и Пароль и так же для просмотра информации в файле требуются права администратора у данного аккаунта. Совокупность этих факторов определяет хорошую защиту данных от посторонних личностей.

## 3.2 Концептуальный прототип

При запуске программного средства пользователю сразу встречается простой интерфейс, с которым пользователь и продолжит дальнейшее пользование приложением. На макете представлено 2 поля для ввода информации: «Логин» в который нужно ввести логин авторизированного пользователя и «Пароль» в который вводится пароль. На рисунке 3 предоставляется макет окна авторизации в программном средстве.

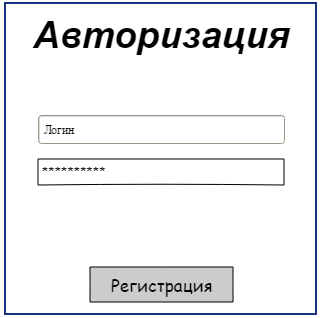


Рисунок 3 – Макет окна авторизации

При нажатии на кнопку Регистрация откроется окно с полями для ввода данных. На макете представлено 3 поля для ввода информации: «Логин» в который нужно ввести логин авторизированного пользователя, «Пароль» в который вводится пароль и «Имя» в которое пользователь вводит своё имя. На рисунке 4 предоставляется макет окна регистрации в программном средстве.

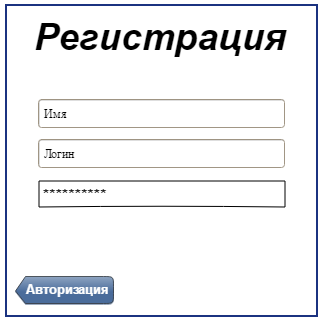


Рисунок 4 – Макет окна регистрации

После успешной авторизации в программном средстве мы переходим в главное меню, в котором предлагается выбрать один из тестов на темперамент, либо же посетить справочное меню приложения. На рисунке 5 предоставляется макет главного меню программного средства

.

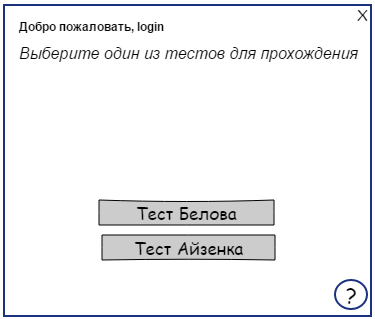


Рисунок 5 – Макет главного меню программного средства.

При выборе одного из тестов мы переходим в окно прохождения теста. На рисунке 6 предоставляется макет окна прохождения теста.

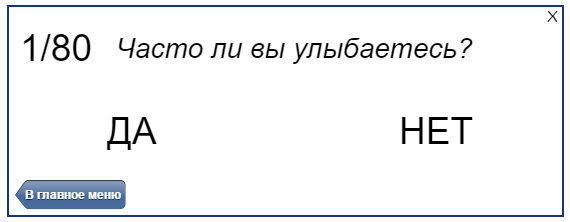


Рисунок 6 – Макет окна прохождения теста

После прохождения теста пользователю открывается окно с его результатом и подробным описанием. Также предлагается пройти тест ещё раз, сохранить результат или выйти в главное меню. Макет окна результата предоставляется на рисунке 7.

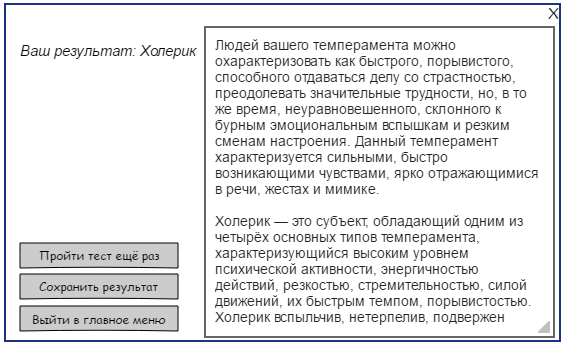


Рисунок 7 – Макет результата теста

При авторизации под данными администратора, ему предоставляются такие возможности как: просмотр всех вопросов, добавление и удаление вопросов, просмотр отчёта по пройденным тестам пользователей, вывод отчёта в файл. Макет данного окна представляется на рисунке 8.

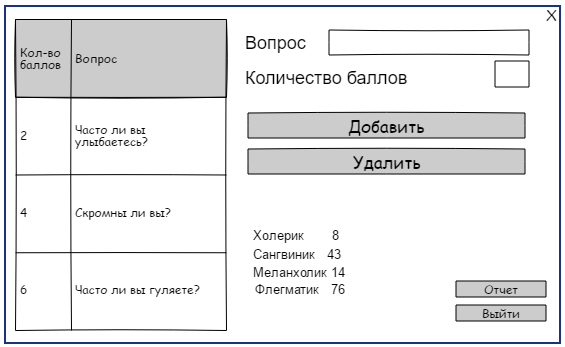


Рисунок 8 – Макет окна администратора

## 3.3 Организация данных

Организация данных подразумевает создание модели данных, главными элементами которой являются сущности и их связи.

Реляционная модель основана на понятии отношения, представлением которого является таблица. В реляционной модели отношения используются для хранения информации об объектах, представленных в базе данных. Отношение имеет вид двухмерной таблицы, в которой строки соответствуют записям, а столбцы – атрибутам. Каждая запись должна однозначно характеризоваться в таблице. Для этого используют первичные и вторичные ключи. Достоинством реляционной модели является простота и удобство физической реализации.

Реляционная модель базы данных подразумевает нормализацию всех таблиц данных. Нормализация – это формальный метод анализа отношений на основе их первичного ключа и функциональных зависимостей, существующих между их атрибутами [5].

Структура базы данных разрабатываемого программного средства включает две таблицы.

Таблица «Users» хранит информацию о пользователях, структура которой приведена в таблице 3.

Таблица 3 – Структура таблицы «Users»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| Login | varchar | 255 | Логин пользователя |
| Name | varchar | 255 | Имя пользователя |
| Password | varchar | 255 | Пароль пользователя |

На рисунке 9 представлена схема базы данных.

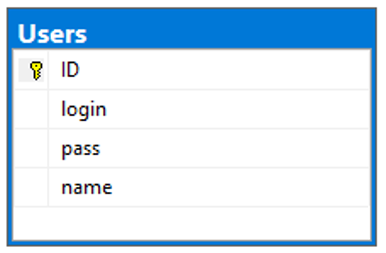


Рисунок 9 – Схема базы данных

## 3.4 Функции логическая и физическая организация

Для авторизации в приложении необходимо перейти в «Форму авторизации», на которой нужно нажать кнопку «Войти». После нажатия кнопки «Войти» вызывается обработчик событий ButtonLogin\_Click, который обрабатывает введенный логин и пароль. Данная реализация функции была реализована в средства Visual Studio Professional 2019, код которого представлен ниже. [8]

private void ButtonLogin\_Click(object sender, RoutedEventArgs e)

{

String loginUser = loginField.Text;

String passUser = passField.Password;

DataBase dataBase = new DataBase();

DataTable table = new DataTable();

SqlDataAdapter adapter = new SqlDataAdapter();

SqlCommand command = new SqlCommand("SELECT \* FROM Users WHERE login = @uL AND pass = @uP", dataBase.getConnection());

command.Parameters.Add("@uL", SqlDbType.VarChar).Value = loginUser;

command.Parameters.Add("@uP", SqlDbType.VarChar).Value = passUser;

adapter.SelectCommand = command;

adapter.Fill(table);

if (loginField.Text == "" || passField.Password == "")

{

MessageBox.Show("Не удалось! Попробуйте ещё раз.");

}

else {

if (table.Rows.Count > 0)

{

if (loginField.Text.ToLower() == "Admin".ToLower())

{

new AdminPanel().Show();

this.Close();

}

else

{

MainAfterAutoriz autoriz = new MainAfterAutoriz(loginField.Text);

autoriz.Title = buttonLogin.Content.ToString();

this.Close();

autoriz.Show();

}

}

else

MessageBox.Show("Не удалось! Попробуйте ещё раз.");

}

}

Для выполнения функции «Регистрации» необходимо выполнить щелчок мыши по кнопке «Регистрация». Данная реализация функции была реализована в средства Visual Studio Professional 2019. Функция позволяет создать новый аккаунт в программном средстве. Код представлен ниже. [8]

private void ButtonLogin\_Click(object sender, RoutedEventArgs e)

{

if (userNameField.Text == "")

{

MessageBox.Show("Введите имя");

return;

}

if (loginField.Text == "")

{

MessageBox.Show("Введите логин");

return;

}

if (passField.Password == "")

{

MessageBox.Show("Введите пароль");

return;

}

if (isUserExists())

return;

DataBase dataBase = new DataBase();

SqlCommand command = new SqlCommand("INSERT INTO Users (login, pass, name) VALUES (@login, @pass, @name)", dataBase.getConnection());

command.Parameters.Add("@login", SqlDbType.VarChar).Value = loginField.Text;

command.Parameters.Add("@pass", SqlDbType.VarChar).Value = passField.Password;

command.Parameters.Add("@name", SqlDbType.VarChar).Value = userNameField.Text;

dataBase.OpenConnection();

if (command.ExecuteNonQuery() == 1)

MessageBox.Show("Создано!");

else

MessageBox.Show("Не удачно! :(");

dataBase.CloseConnection();

MainWindow mainWindow = new MainWindow();

mainWindow.Title = mainWindow.Content.ToString();

this.Hide();

mainWindow.Show();

}

Полный код программного средства располагается в приложении А.

## 3.5 Проектирование справочной системы

Для корректной работы с приложением требуется обеспечить пользователя справочной системой, в которой будут приведены приемы работы с приложением, включающие данные о том, что произойдёт после нажатия на определенную кнопку или при выборе пункта меню.

Справочная система необходима для ознакомления с программой. В ней должна присутствовать информация, которая может напомнить принцип работы программного средства.

Структура справочной системы:

– введение – предварительные сообщения общего характера, предпосылаемые произведению, с целью ввести пользователя в курс программного средства;

– виды тестов – описаны все виды тестов, какие пользователь может найти в программном средстве;

– основной интерфейс программного средства – описаны главные функции программного средства;

– контекстное меню – описано как пользоваться меню программного средства;

– типы темпераментов – описано какие типы темперамента существуют и т.д.;

– прохождение теста – как пройти тест, выбрать подходящий и т.д.;

– о программе – описано что собой представляет программное средство.

Справочная система программного средства представляет собой отдельное окно с полным описанием основных функций программного средства. Чтобы загрузить справочную систему, нужно нажать кнопку со знаком вопроса в правой части ленты. В справочной системе даны ответы на типичные вопросы, возникающие при работе с приложением, что, несомненно, должно помочь при освоении программного средства.

Справочная система будет разработана в Visual Studio Professional 2019.

# 4 Описание программного средства

## 4.1 Общие сведения

Тестирующее программное средство на определение темперамента предназначено **для эффективного использования аппаратных и программных средств персонально компьютера и функционирования программного продукта**.

Загрузочным файлом программного средства является «KursachWPF.exe».

Основными задачами программного средства являются:

– предоставление пользователю простой интерфейс;

– удобство использования приложения.

Данное программное средство было разработано на персональном компьютере со следующей конфигурацией:

– операционная система Windows 10;

– процессор Intel Core i5-3210M CPU @ 2.50GHz;

– ОЗУ 8 ГБ;

– видеокарта Intel HD Graphics 4000, NVIDIA GeForce GT 650M;

– места на HDD 1000 ГБ;

– мышь;

– клавиатура.

Программное средство создано в средстве разработки Visual Studio Professional 2019 на языке программирования C# в операционной системе Windows 10. Программное средство может работать в средах операционных систем семейства Microsoft Windows, начиная с Windows 7.

## 4.2 Функциональное назначение

Программное средство предназначено для тестирования на тип темперамента, добавления данных пользователя в базу данных и взаимодействия с данными. **Данное программное средство предназначено для проведения теста на тип темперамента, путём созданных в** программном средстве **улучшенных алгоритмов, которые с наивысшей точностью определяют темперамент пользователя с использованием облегчённого интерфейса для удобного и быстрого пользования программой. [9]**

Программное средство использует стандартные элементы управления, такие как кнопки, списки, поля ввода, что обеспечивает удобство в использовании, понятный и удобный интерфейс программного средства. **Данный программный продукт не имеет широкого назначения, поэтому сетевой поддержки не имеет.**

## 4.3 Входные и выходные данные

Входными данными являются данные пользователя при авторизации. Входные данные программного средства представлены в таблице 4. [10]

Таблица 4 – Входные данные

|  |  |  |  |
| --- | --- | --- | --- |
| Входные данные | Тип поля | Размер поля, байт | Описание поля |
| Login | varchar | 255 | Логин пользователя |
| Password | varchar | 255 | Пароль пользователя |
| Name | varchar | 255 | Имя пользователя |

Продолжение таблицы 4

|  |  |  |  |
| --- | --- | --- | --- |
| Входные данные | Тип поля | Размер поля, байт | Описание поля |
| Questions | varchar | 255 | Вопросы для теста |

Выходными данными являются данные отчёта о типе темперамента пользователя. Выходные данные программного средства представлены в таблице 5.

Таблица 5 – Выходные данные

|  |  |  |  |
| --- | --- | --- | --- |
| Выходные данные | Тип поля | Размер поля, байт | Описание поля |
| Temperament | varchar | 255 | Темперамент пользователя |
| Pass Tests | int | 4 | Количество пройденных тестов |

Подробные данные о выходных данных находятся в приложении Б. Отчёт после прохождения теста расписана на рисунке Б.1 в приложении Б.

# 5 Методика испытаний

## 5.1 Технические требования

Данная программное средство было протестировано на персональном компьютере со следующей конфигурацией:

– операционная система Windows 10;

– процессор Intel Core i5-3210M CPU @ 2.50GHz;

– ОЗУ 8 ГБ;

– видеокарта Intel HD Graphics 4000, NVIDIA GeForce GT 650M;

– места на HDD 1000 ГБ;

– мышь;

– клавиатура.

## 5.2 Порядок проведения испытаний

### 5.2.1 Функциональное тестирование

В процессе написания программного средства необходимо производить тестирование на правильность работы приложения. Одной из основных задач тестирования является устранение ошибок, происходящих при вводе данных.

Функциональное тестирование – это тестирование функций приложения на соответствие требованиям. Оценка производится в соответствии с ожидаемыми и полученными результатами (на основании функциональной спецификации), при условии, что функции отрабатывали на различных значениях.

Тестирование всего программного средства происходит по средствам проверки всех функций, представленных в программном средстве.

Тестирование программного средства начинается с главного окна программного средства – это окно авторизации. На рисунке 10 представлено главное окно программного средства.

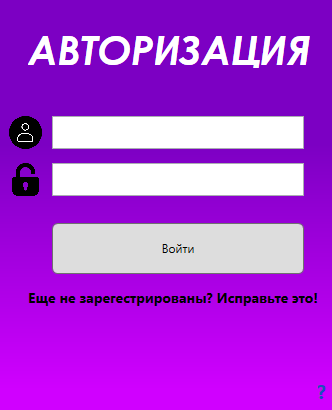


Рисунок 10 – Главное окно программного средства

Тест-кейс для авторизации приведена в таблице 6.

Таблица 6 – Тест-кейс для авторизации

|  |  |  |  |
| --- | --- | --- | --- |
| № | Модель / Функция | Шаги воспроизведения | Результат |
| 1 | Авторизация | 1. Войти в программу;  2. В поле «Login» ввести «Gleb»;  3. В поле «Password» ввести «qwerty123». | Ожидаемый: вход в программу под ником пользователя и отображение главного окна программного средства. |
| Окно авторизации с заполненными полями представлено на рисунке 11  Фактический: фактический результат представлен на рисунке 5.3, соответствует ожидаемому результату, все условия соблюдены. |

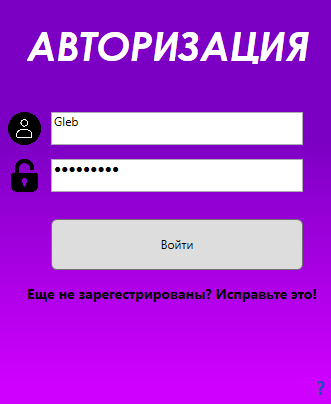


Рисунок 11 – Окно авторизации

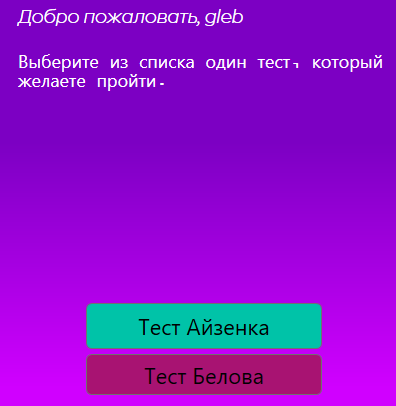


Рисунок 12 – Главное окно программного средства

Тест-кейс для прохождения теста приведен в таблице 7.

Таблица 7 – Тест-кейс для прохождения теста

|  |  |  |  |
| --- | --- | --- | --- |
| № | Модель / Функция | Шаги воспроизведения | Результат |
| 2 | Прохождение теста | 1. Выбрать «Тест Белова»;  2. Ответить на все предложенные вопросы. | Ожидаемый:Появляется окно с результатом теста |
| Фактический: фактический результат представлен на рисунке 13, соответствует ожидаемому результату, все условия соблюдены. |

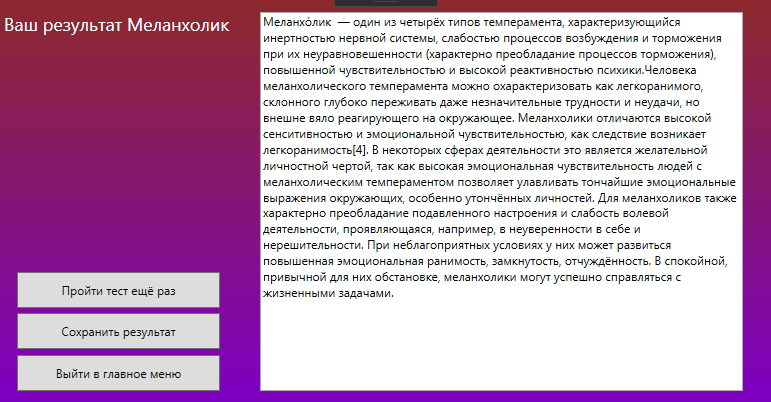


Рисунок 13 – Окно результата теста

### 5.2.2 Полное тестирование

Выполнение полного тестирование программного средства начинается с главного окна программного средства. На рисунке 14 представлено главное окно программного средства.



Рисунок 14 – Главное окно программного средства

Тест-кейс для проведения регистрации приведена в таблице 8.

Таблица 8 – Тест-кейс для проведения регистрации

|  |  |  |  |
| --- | --- | --- | --- |
| № | Модель / Функция | Шаги воспроизведения | Результат |
| 1 | Регистрация | 1. Войти в приложение;  2. Заполнить поля требуемыми данными   * 1. «Имя» - Пётр   2. «Логин» - Petr228   3. «Пароль» - qwerty   3. Нажать кнопку «Зарегистрироваться» . | Ожидаемый:вывод сообщения о успешной регистрации, откроется окно авторизации. |
| Окно регистрации с заполненными полями приведено на рисунке 15  Фактический: фактический результат представлен на рисунке 16, соответствует ожидаемому результату, все условия соблюдены. |

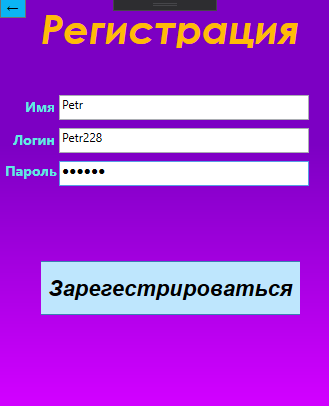


Рисунок 15 – Окно регистрации

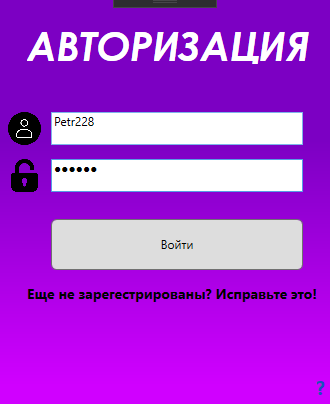


Рисунок 16 – Окно авторизации

Тест-кейс для проведения авторизации приведен в таблице 9.

Таблица 9 – Тест-кейс для проведения авторизации.

|  |  |  |  |
| --- | --- | --- | --- |
| № | Модель / Функция | Шаги воспроизведения | Результат |
| 2 | Авторизация | 1. Войти в приложение; 2. Заполнить поля требуемыми данными    1. «Логин» - Petr228    2. «Пароль» - qwerty 3. Нажать кнопку «Войти». | Ожидаемый:вход в программу и отображение главного окна программного средства. |
| Окно авторизации с заполненными полями представлено на рисунке 16  Фактический: фактический результат представлен на рисунке 5.9, соответствует ожидаемому результату, все условия соблюдены. |

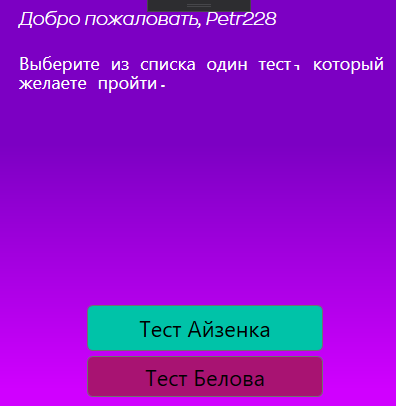


Рисунок 17 – Фактический результат

Тест-кейс для начала теста приведен в таблице 10.

Таблица 10 – Тест-кейс для начала теста.

|  |  |  |  |
| --- | --- | --- | --- |
| № | Модель / Функция | Шаги воспроизведения | Результат |
| 3 | Начало теста | 1. Войти в приложение; 2. Пройти процесс авторизации; 3. Нажать на кнопку «Тест Белова». | Ожидаемый:открытие теста с первым вопросом и вариантами ответа |
| Окно теста представлено на рисунке 18  Фактический: фактический результат представлен на рисунке 19, соответствует ожидаемому результату, все условия соблюдены. |

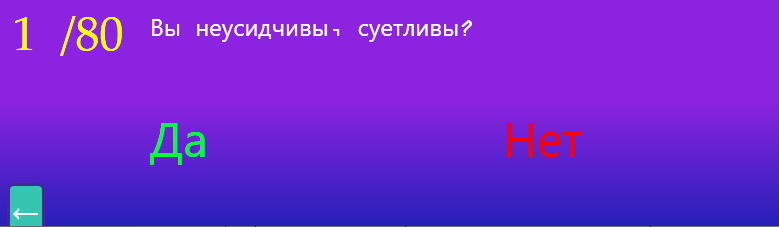


Рисунок 18 – Окно теста

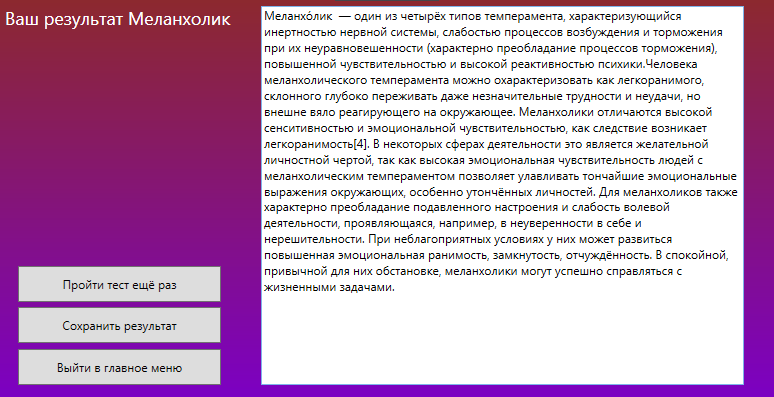


Рисунок 19 – Фактический результат

Тест-кейс для вывода отчёта приведен в таблице 11.

Таблица 11 – Тест-кейс для вывода отчёта

|  |  |  |  |
| --- | --- | --- | --- |
| № | Модель / Функция | Шаги воспроизведения | Результат |
| 4 | Вывод отчёта | 1. Войти в приложение; 2. Пройти процесс авторизации; 3. Нажать на кнопку «Тест Белова»; 4. Пройти тест до конца; 5. В окне результата выбрать кнопку «Сохранить результат». | Ожидаемый: вывод окна с выбором места сохранения документа |
| Окно результата представлено на картинке 19  Фактический: фактический результат представлен на рисунке 20, соответствует ожидаемому результату, все условия соблюдены. |

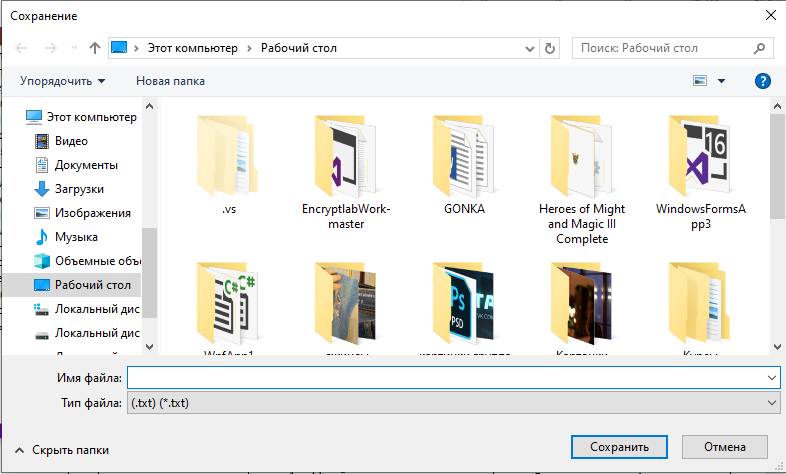


Рисунок 20 – Фактический результат

Тест-кейс для добавления вопроса приведен в таблице 12.

Таблица 12 – Тест-кейс для добавления вопроса

|  |  |  |  |
| --- | --- | --- | --- |
| № | Модель / Функция | Шаги воспроизведения | Результат |
| 5 | Добавление вопроса | 1. Войти в приложение; 2. Пройти процесс авторизации для администратора 3. Заполнить требуемые поля необходимыми данными:   а. «Вопрос» - «Вы часто улыбаетесь?  б. «Баллы за Да» - 5  в. «Баллы за Нет» - 2. | Ожидаемый: добавление вопроса в тест |
| Окно таблицы вопросов представлено на рисунке 21  Фактический: фактический результат представлен на рисунке 22, соответствует ожидаемому результату, все условия соблюдены. |

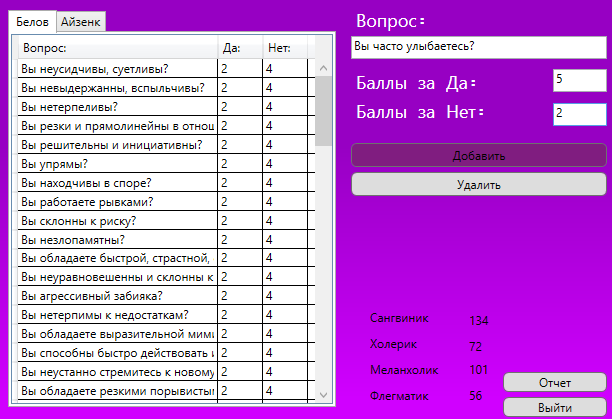


Рисунок 21 – Админ панель программного средства

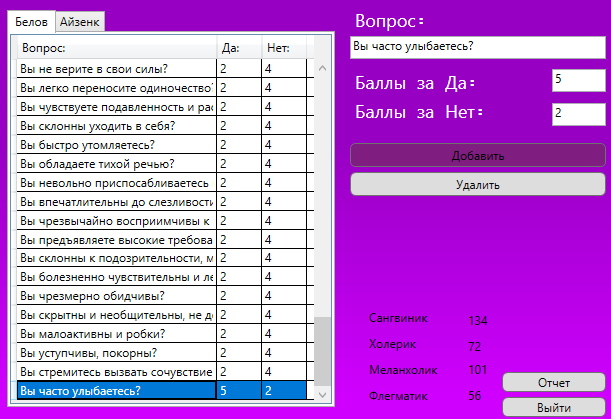


Рисунок 22 – Фактический результат

Тест-кейс для удаления вопроса в таблице 13.

Таблица 13 – Тест-кейс для удаления вопроса

|  |  |  |  |
| --- | --- | --- | --- |
| № | Модель / Функция | Шаги воспроизведения | Результат |
| 6 | Удаление вопроса | 1. Войти в приложение; 2. Пройти процесс авторизации для администратора 3. Выбрать нужный вопрос и кликнуть на него мышью; 4. Нажать кнопку «Удалить». | Ожидаемый: удаление вопроса. |
| Окно таблицы вопросов представлено на рисунке 23  Фактический: фактический результат представлен на рисунке 24, соответствует ожидаемому результату, все условия соблюдены. |

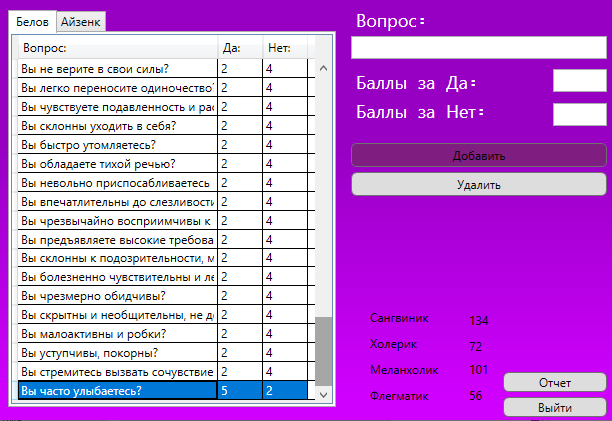


Рисунок 23 - Админ панель программного средства

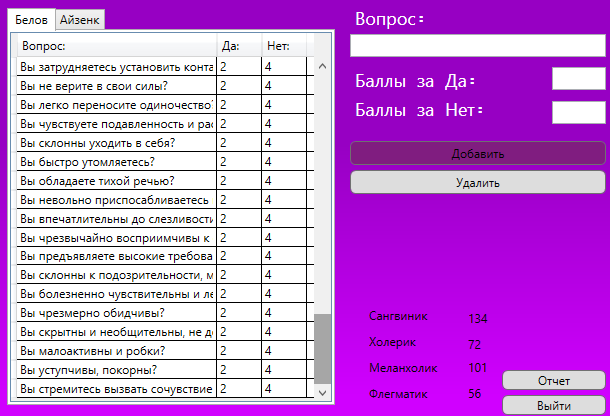


Рисунок 24 – Фактический результат

# 6 Применение

## 6.1 Назначение программы

Данное программное средство предназначено для тестирования пользователей на тип темперамента. Программное средство помогает быстро и удобно пройти тест и определить свой темперамент.

Предполагается, что программное средство будет использовано пользователем, который интересуется своим темпераментом и желает его узнать. Интерфейс программного средства соблюдает требования возложенные на него, он удобен, практичен, не перегружен, красив и приятен.

## 6.2 Условие применения

Для работы с данным программным средством автоматизация аренды недвижимости требуется:

– операционная система начиная с Windows 7;

– процессор от Intel Pentium 4 – 2365M CPU @ 1.30GHz;

– ОЗУ от 500 МБ;

– видеокарта от Palit GeForce 610;

– места на HDD 100 МБ;

– мышь / touchpad;

– клавиатура;

– программная платформа Microsoft .NET Framework 4.7

## 6.3 Справочная система

Для корректной работы с приложением была создана для пользователя справочная система, в которой приведены приемы работы с приложением, включающие данные о том, что произойдёт после нажатия на определенную кнопку или при выборе пункта меню.

Справочная система была создана для ознакомления с программой. В ней присутствует информация, которая может напомнить принцип работы программного средства.

Структура справочной системы:

– о программе – предварительные сообщения общего характера, предпосылаемые произведению, с целью ввести пользователя в курс программного средства;

– назначение программы – описано предназначение программного средства, также ввод в основные тезисы;

– применение – описан основной сценарий использования, введенные ограничения и описание как пользоваться программой.

Справочная система программного средства представляет собой отдельное окно с полным описанием основных функций программного средства. Чтобы загрузить справочную систему, нужно нажать кнопку со знаком вопроса в правом нижнем углу программного средства. В справочной системе даны ответы на типичные вопросы, возникающие при работе с программным средством, что, несомненно, должно помочь при освоении программного средства.

На рисунке 21 представлен вводный раздел «О программе».

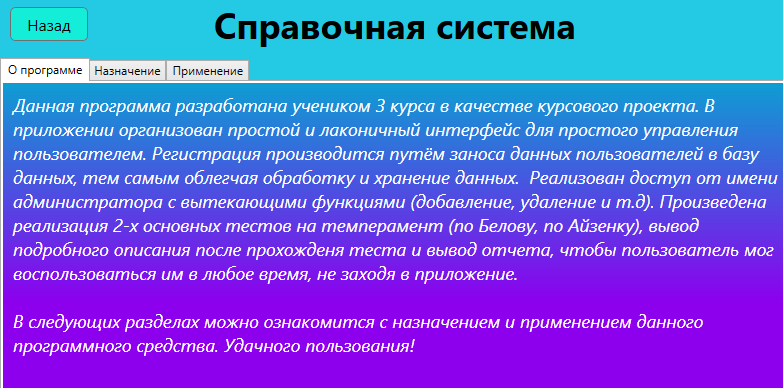


Рисунок 21 – Справочный раздел «О программе»

На рисунке 22 представлен справочный раздел «Назначение».



Рисунок 22 – Справочный раздел «Назначение»

На рисунке 23 представлен справочный раздел «Применение».

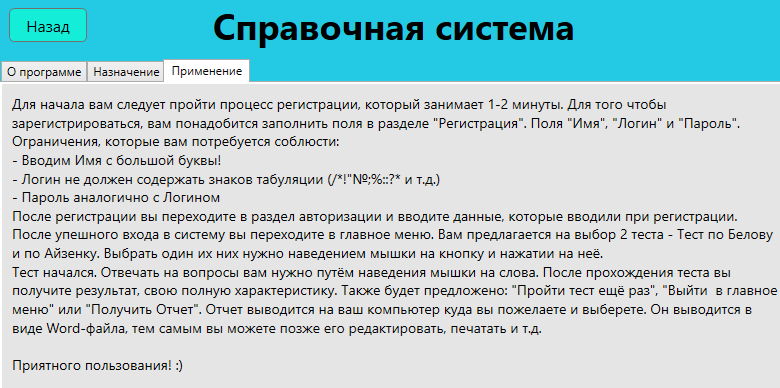


Рисунок 23 – Справочный раздел «Применение»

# Заключение

В рамках курсового проектирования на тему «Тестирующая программа на определение темперамента» было разработано программное средство «KursachWPF.exe», позволяющее проходить тесты, получать точный результат теста, выводить отчёты, производить регистрацию и авторизацию аккаунтов.

Для достижения цели курсового проектирования были решены следующие задачи:

– определена вычислительная система, необходимая для создания программного средства;

– разработана физическая и логическая модель данных;

– по модели выполнено проектирование задачи;

– разработано программное средство;

– описано созданное программное средство;

– выбрана методика испытаний;

– описан процесс тестирования;

– приведены примеры области применения.

Проект был разработан в среде Visual Studio Professional 2019 на языке C#. База данных была реализована в Microsoft SQL Server Management Studio 18.

Программное средство имеет свои недостатки, такие как:

– не реализовано сохранение процесса теста;

– малое количество тестов;

– не реализовано удаление аккаунта зарегестрированного пользователя.

А также свои преимущества:

– простой интерфейс;

– удобство использования;

– информативность.

Программное средство реализовано в полном объеме и в соответствии с заданными требованиями, полностью отлажена и протестирована. Поставленные задачи выполнены.

Программное средство готово к практическому использованию и может быть дополнено и модернизировано.

# Список информационных источников

1 Михалевич В.Ю. Методические указания к курсовому проектированию для учащихся специальности 2-40 01 01 «Программное обеспечение информационных технологий» / В.Ю. Михалевич. – Минск: КБП, 2020

2 Багласова Т.Г. Методические указания по оформлению курсовых и дипломных проектов / Т.Г.Багласова, К.О.Якимович. – Минск: КБП, 2013

3 Программа и методы испытаний. Требования к содержанию, оформлению и контролю качества : ГОСТ 19.301-2000. – Введ. 01.09.2001. – Минск : Межгос. Совет по стандартизации, метрологии и сертификации, 2000. – 16 с.

4 Руководство по программированию на C# [Электронный ресурс]. – Microsoft, 2020. – Режим доступа : http://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/. – Дата доступа : 15.03.2020.

5 SQL Задачи и решения [Электронный ресурс]. – Sql-Tutorial, 2020. – Режим доступа : http://www.sql-tutorial.ru. – Дата доступа : 22.02.2010.

6 Обзор обновлений и новых функций Windows 10 [Электронный ресурс]. – Microsoft, 2020. – Режим доступа : https://www.microsoft.com/ru-ru/windows/features/. – Дата доступа : 18.03.2020.

7 Visual Studio 2019 [Электронный ресурс]. – Microsoft, 2020. – Режим доступа : https://visualstudio.microsoft.com/ru/vs/. – Дата доступа : 23.04.2020.

8 Уроки C# .NET Windows Forms Авторизация пользователя через базу данных [Электронный ресурс]. – Youtube, 2015. Режим доступа : https://www.youtube.com/watch?v=gnvxVv60xnI&t. Дата доступа: 04.02.2020.

9 Winforms Boutique Dark UI Demo Template (UI + Dataviz) [Электронный ресурс]. – Youtube, 2015. Режим доступа : https://www.youtube.com/watch?v=pnlSDH0SCHo&t. Дата доступа: 12.02.2020.

10 How to Create Quiz Application in C#.NET using SQL Server [Электронный ресурс]. – Youtube, 2015. Режим доступа : https://www.youtube.com/watch?v=\_GX7A2jIovI&t. Дата доступа: 24.03.2020.

# Приложение А

(обязательное)

Текст программных модулей

//AdminPanel.cs

using Microsoft.Win32;

using System;

using System.Data;

using System.IO;

using System.Windows;

using System.Windows.Documents;

namespace Kursa4Wpf

{

/// <summary>

/// Логика взаимодействия для AdminPanel.xaml

/// </summary>

public partial class AdminPanel : Window

{

public class Quest

{

public string Question { get; set; }

public string Yes { get; set; }

public string No { get; set; }

public Quest(string question, string yes, string no)

{

Question = question;

Yes = yes;

No = no;

}

}

public void AddQuestsBelov() //метод добавления теста в таблицу панели

{

DataTable.Items.Clear();

using (var sr = new StreamReader("tests.txt")) //указываем откуда считываем

{

while (!sr.EndOfStream)

{

try //считываем вопросы из файла вместе с баллами

{

var flex = sr.ReadLine().Split('|');

var f = new Quest(flex[0], flex[1], flex[2]);

DataTable.Items.Add(f);

}

catch (Exception)

{

continue;

}

}

}

}

public void AddQuestsAizenk() //добавление в другую таблицу вопросов

{

DataTable1.Items.Clear();

using (var sr = new StreamReader("aizenkque.txt"))

{

while (!sr.EndOfStream)

{

try

{

var flex = sr.ReadLine().Split('|');

var f = new Quest(flex[0], flex[1], flex[2]);

DataTable1.Items.Add(f);

}

catch (Exception)

{

continue;

}

}

}

}

public AdminPanel()

{

InitializeComponent();

AddQuestsBelov();

CounterStats();

AddQuestsAizenk();

}

private void AddBtn\_Click(object sender, RoutedEventArgs e) //добавление вопроса в тест

{

if (TabTable.SelectedIndex == 0)

{

if (VoprosBox.Text != null && YesBals.Text != null && NopeBals != null) //проверка на заполненые поля

{

using (var sw = new StreamWriter("tests.txt", true))

sw.WriteLine($"{VoprosBox.Text}|{YesBals.Text}|{NopeBals.Text}");

}

}

else

{

if (VoprosBox.Text != null && YesBals.Text != null && NopeBals != null)

{

using (var sw = new StreamWriter("aizenkque.txt", true))

sw.WriteLine($"{VoprosBox.Text}|{YesBals.Text}|{NopeBals.Text}");

}

}

AddQuestsBelov();

AddQuestsAizenk();

}

private void RemoveBtn\_Click(object sender, RoutedEventArgs e) // удаление вопроса

{

string tempFile = System.IO.Path.GetTempFileName();

if (TabTable.SelectedIndex == 0) //проверяем, в какой вкладке мы удаляем вопрос

{

Quest row = (Quest)DataTable.SelectedItems[0];

string line\_to\_delete = $"{row.Question}|{row.Yes}|{row.No}";

using (StreamReader reader = new StreamReader(@"tests.txt"))

{

using (StreamWriter writer = new StreamWriter(tempFile)) //создаем временный темп-файл

{

string line;

while ((line = reader.ReadLine()) != null)

{

if (!(line == line\_to\_delete))

{

writer.WriteLine(line);

}

}

reader.Close();

File.Delete(@"tests.txt");

writer.Close();

File.Move(tempFile, @"tests.txt");

}

}

}

else

{

Quest row = (Quest)DataTable1.SelectedItems[0];

string line\_to\_delete = $"{row.Question}|{row.Yes}|{row.No}";

using (StreamReader reader = new StreamReader(@"aizenkque.txt"))

{

using (StreamWriter writer = new StreamWriter(tempFile))

{

string line;

while ((line = reader.ReadLine()) != null)

{

if (!(line == line\_to\_delete))

{

writer.WriteLine(line);

}

}

reader.Close();

File.Delete(@"aizenkque.txt");

writer.Close();

File.Move(tempFile, @"aizenkque.txt");

}

}

}

AddQuestsBelov();

AddQuestsAizenk();

}

void CounterStats() //считываем данные статистики из файла

{

using(StreamReader flex = new StreamReader("stats.txt"))

{

SangLabel.Content = flex.ReadLine();

HollLabel.Content = flex.ReadLine();

MelLabel.Content = flex.ReadLine();

FlexLabel.Content = flex.ReadLine();

}

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

SaveFileDialog save = new SaveFileDialog(); //сохранение в текст.файл

save.Filter = "(.txt)|\*.txt"; //указываем расширение

if (save.ShowDialog() == true)

{

using (StreamWriter flex = new StreamWriter(save.FileName, false)) //заполняем файл

{

flex.WriteLine("Сангвиников: " + SangLabel.Content);

flex.WriteLine("Холериков: " + HollLabel.Content);

flex.WriteLine("Меланхоликов: " + MelLabel.Content);

flex.WriteLine("Флегматиков: " + FlexLabel.Content);

}

}

}

private void ExitButton\_Click(object sender, RoutedEventArgs e)

{

MainWindow mainWindow = new MainWindow();

mainWindow.Show();

this.Close();

}

private void YesBals\_TextChanged(object sender, System.Windows.Controls.TextChangedEventArgs e)

{

}

}

}

//AizenkTest.cs

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace Kursa4Wpf

{

/// <summary>

/// Логика взаимодействия для AizenkTest.xaml

/// </summary>

public partial class AizenkTest : Window

{

public AizenkTest(string userName)

{

username = userName;

InitializeComponent();

Tests = new StreamReader("aizenkque.txt"); //создаем файл

string line;

while ((line = Tests.ReadLine()) != null) //читаем по одной линии формат "Вопрос|БаллыЗаДа|БаллыЗаНет"

{

max++;

}

Tests.BaseStream.Position = 0;

LabelCounter1.Content = "/" + max.ToString();

GetQuest();

}

public int Flex = 0; // Points

public int counter = 0; // Номер вопроса

public int max = 0; // Всего тестов

public string QuestText = null;

public string pointsYes = null;

public string pointsNo = null;

public string username = null;

public StreamReader Tests;

private void GetQuest()

{

if (counter != max)

{

++counter;

var result = Tests.ReadLine().Split('|');

QuestText = result[0];

pointsYes = result[1];

pointsNo = result[2];

}

MainTextBlock1.Text = QuestText;

LabelCounter.Content = counter; //присвоение к счётчику вопросов

}

private void GoNext\_Flex(bool Check) //переход к следующему вопросу

{

if (Check)

{

Flex += Convert.ToInt32(pointsYes);

}

else

{

Flex += Convert.ToInt32(pointsNo);

}

Debug.WriteLine(Flex);

if (counter == max)

{

EndQuests();

}

GetQuest();

}

private void EndQuests() //расчёт результата теста

{

string Result = null;

if (Flex < -40)

{

Result = "Сангвиник";

}

if (Flex > 40)

{

Result = "Холерик";

}

if (Flex < 40 && Flex > 0)

{

Result = "Меланхолик";

}

if (Flex > -40 && Flex <= 0)

{

Result = "Флегматик";

}

ResultWindow window = new ResultWindow(Result, username);

window.Show();

this.Close();

}

private void YesButtonClick(object sender, MouseButtonEventArgs e)

{

GoNext\_Flex(true);

}

private void ButtonBack\_Click(object sender, RoutedEventArgs e)

{

MainAfterAutoriz mainAfterAutoriz = new MainAfterAutoriz(username); //переход к главному окну

mainAfterAutoriz.Show();

this.Close();

}

private void NoBtn\_PreviewMouseDown(object sender, MouseButtonEventArgs e)

{

GoNext\_Flex(false);

}

}

}

//BelovTest.cs

using System;

using System.Diagnostics;

using System.IO;

using System.Windows;

using System.Windows.Input;

namespace Kursa4Wpf

{

public partial class BelovTest : Window

{

public int Flex = 0; // Points

public int counter = 0; // Номер вопроса

public int max = 0; // Всего тестов

public string QuestText = null;

public string pointsYes = null;

public string pointsNo = null;

public string username = null;

public StreamReader Tests;

public BelovTest(string userName)

{

username = userName;

InitializeComponent();

Tests = new StreamReader("tests.txt");

string line;

while ((line = Tests.ReadLine()) != null) //читаем по одной линии формат "Вопрос|БаллыЗаДа|БаллыЗаНет"

{

max++;

}

Tests.BaseStream.Position = 0;

LabelCounter1.Content = "/" + max.ToString();

GetQuest();

}

private void GetQuest() //получаем вопрос из файла

{

if (counter != max)

{

++counter;

var result = Tests.ReadLine().Split('|');

QuestText = result[0];

pointsYes = result[1];

pointsNo = result[2];

}

MainTextBlock1.Text = QuestText;

LabelCounter.Content = counter;

}

private void GoNext\_Flex(bool Check) //переход к следующему вопросу

{

if (Check)

{

Flex += Convert.ToInt32(pointsYes);

}

else

{

Flex += Convert.ToInt32(pointsNo);

}

Debug.WriteLine(Flex);

if (counter == max)

{

EndQuests();

}

GetQuest();

}

private void EndQuests() //расчёт результата теста

{

string Result = null;

if (Flex < -40)

{

Result = "Сангвиник";

}

if (Flex > 40)

{

Result = "Холерик";

}

if (Flex < 40 && Flex > 0)

{

Result = "Меланхолик";

}

if (Flex > -40 && Flex <= 0)

{

Result = "Флегматик";

}

ResultWindow window = new ResultWindow(Result,username); //после теста переходим в окно результата теста

window.Show();

this.Close();

}

private void YesButtonClick(object sender, MouseButtonEventArgs e)

{

GoNext\_Flex(true);

}

private void ButtonBack\_Click(object sender, RoutedEventArgs e) // переходим в главное окно

{

MainAfterAutoriz mainAfterAutoriz = new MainAfterAutoriz(username);

mainAfterAutoriz.Show();

this.Close();

}

private void NoBtn\_PreviewMouseDown(object sender, MouseButtonEventArgs e)

{

GoNext\_Flex(false);

}

}

}

//DataBase.cs

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Kursa4Wpf

{

class DataBase //класс БД

{

SqlConnection Connection = new SqlConnection("Data Source=.\\SQLEXPRESS;Initial Catalog=Authoriz;Integrated Security=True"); //подключение к sql

public void OpenConnection () // открываем соединение

{

if (Connection.State == System.Data.ConnectionState.Closed)

Connection.Open();

}

public void CloseConnection() // закрываем соединение

{

if (Connection.State == System.Data.ConnectionState.Open)

Connection.Close();

}

public SqlConnection getConnection() //статус подключения

{

return Connection;

}

}

}

//HelpSystem.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace Kursa4Wpf

{

/// <summary>

/// Логика взаимодействия для HelpSystem.xaml

/// </summary>

public partial class HelpSystem : Window

{

public HelpSystem()

{

InitializeComponent();

}

private void Button\_Click(object sender, RoutedEventArgs e) //переходим в главное окно

{

MainWindow mainWindow = new MainWindow();

mainWindow.Show();

this.Close();

}

}

}

//MainAfterAutoriz.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace Kursa4Wpf

{

/// <summary>

/// Логика взаимодействия для MainAfterAutoriz.xaml

/// </summary>

public partial class MainAfterAutoriz : Window

{

string User = null;

public MainAfterAutoriz(string username) // переносим глобальную переменную

{

User = username;

InitializeComponent();

DearBlock.Text = $"Добро пожаловать, {username}"; //тут ее используем

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

AizenkTest aizenkTest = new AizenkTest(User);

aizenkTest.Show();

this.Close();

}

private void ButtonBelov\_Click(object sender, RoutedEventArgs e) //переход на тест Белова

{

BelovTest belov = new BelovTest(User);

belov.Show();

this.Close();

}

}

}

//MainWindow.cs

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace Kursa4Wpf

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private void ButtonLogin\_Click(object sender, RoutedEventArgs e) //подключение к БД

{

String loginUser = loginField.Text;

String passUser = passField.Password;

DataBase dataBase = new DataBase();

DataTable table = new DataTable();

SqlDataAdapter adapter = new SqlDataAdapter();

SqlCommand command = new SqlCommand("SELECT \* FROM Users WHERE login = @uL AND pass = @uP", dataBase.getConnection()); //запрос sql

command.Parameters.Add("@uL", SqlDbType.VarChar).Value = loginUser; //присваиваем заглушкам поля

command.Parameters.Add("@uP", SqlDbType.VarChar).Value = passUser;

adapter.SelectCommand = command;

adapter.Fill(table);

if (loginField.Text == "" || passField.Password == "") // проверка на пустые поля

{

MessageBox.Show("Не удалось! Попробуйте ещё раз.");

}

else {

if (table.Rows.Count > 0)

{

if (loginField.Text.ToLower() == "Admin".ToLower()) //проверка на админа

{

new AdminPanel().Show();

this.Close();

}

else

{

MainAfterAutoriz autoriz = new MainAfterAutoriz(loginField.Text);

autoriz.Title = buttonLogin.Content.ToString();

this.Close();

autoriz.Show();

}

}

else

MessageBox.Show("Не удалось! Попробуйте ещё раз.");

}

}

private void Label\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e) //свойство для перехода в окно регистрации

{

RegisterWindow registerWindow = new RegisterWindow();

registerWindow.Title = RegistrationBlock.Visibility.ToString();

this.Close();

registerWindow.Show();

}

private void HelpLabel\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

HelpSystem help = new HelpSystem();

help.Show(); //переход в справочную систему

this.Close();

}

}

}

//RegisterWindow.cs

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace Kursa4Wpf

{

/// <summary>

/// Логика взаимодействия для RegisterWindow.xaml

/// </summary>

public partial class RegisterWindow : Window

{

public RegisterWindow()

{

InitializeComponent();

}

private void ButtonLogin\_Click(object sender, RoutedEventArgs e)

{

if (userNameField.Text == "") //проверка на пустое поле

{

MessageBox.Show("Введите имя");

return;

}

if (loginField.Text == "") //проверка на пустое поле

{

MessageBox.Show("Введите логин");

return;

}

if (passField.Password == "") //проверка на пустое поле

{

MessageBox.Show("Введите пароль");

return;

}

if (isUserExists())

return;

DataBase dataBase = new DataBase();

SqlCommand command = new SqlCommand("INSERT INTO Users (login, pass, name) VALUES (@login, @pass, @name)", dataBase.getConnection());

command.Parameters.Add("@login", SqlDbType.VarChar).Value = loginField.Text;

command.Parameters.Add("@pass", SqlDbType.VarChar).Value = passField.Password;

command.Parameters.Add("@name", SqlDbType.VarChar).Value = userNameField.Text;

dataBase.OpenConnection();

if (command.ExecuteNonQuery() == 1) //авторизация

MessageBox.Show("Создано!");

else

MessageBox.Show("Не удачно! :(");

dataBase.CloseConnection();

MainWindow mainWindow = new MainWindow();

mainWindow.Title = mainWindow.Content.ToString();

this.Hide();

mainWindow.Show();

}

public Boolean isUserExists()

{

DataBase dataBase = new DataBase();

DataTable table = new DataTable();

SqlDataAdapter adapter = new SqlDataAdapter();

SqlCommand command = new SqlCommand("SELECT \* FROM Users WHERE login = @uL", dataBase.getConnection());

command.Parameters.Add("@uL", SqlDbType.VarChar).Value = loginField.Text;

adapter.SelectCommand = command;

adapter.Fill(table);

if (table.Rows.Count > 0) //проверка на существующий логин

{

MessageBox.Show("Такой логин уже существует, введите другой.");

return true;

}

else

return false;

}

private void Button\_Click(object sender, RoutedEventArgs e) //переход в главное окно авторизации

{

MainWindow mainWindow = new MainWindow();

mainWindow.Title = mainWindow.Content.ToString();

this.Hide();

mainWindow.Show();

}

}

}

//ResultWindow.cs

using Microsoft.Win32;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace Kursa4Wpf

{

/// <summary>

/// Логика взаимодействия для ResultWindow.xaml

/// </summary>

public partial class ResultWindow : Window

{

public string User = null;

public ResultWindow(string result,string username)

{

User = username;

InitializeComponent();

StatsReader();

CheckResult(result);

}

int SangCounter = 0;

int HollCounter = 0;

int MelLabel = 0;

int FlexLabel = 0;

void StatsReader()

{

using (StreamReader flex = new StreamReader("stats.txt")) // считываем результаты

{

SangCounter = Convert.ToInt32(flex.ReadLine());

HollCounter = Convert.ToInt32(flex.ReadLine());

MelLabel = Convert.ToInt32(flex.ReadLine());

FlexLabel = Convert.ToInt32(flex.ReadLine());

}

}

void CheckResult(string name) //заполняем текст бокс подробным описанием

{

using(StreamWriter flex = new StreamWriter("stats.txt",false))

{

switch (name)

{

case "Сангвиник":

ResultBlock.Text = "Ваш результат: " + name;

InfoBox.Text = "Человека сангвинического темперамента можно охарактеризовать как живого, подвижного, сравнительно легко переживающего неудачи и неприятности. Сангвиник — это личность, характеризующаяся высокой психической активностью, энергичностью, работоспособностью, быстротой и живостью движений, разнообразием и богатством мимики, быстрым темпом речи." +

"Вы стремитесь к частой смене впечатлений, легко и быстро отзываетесь на окружающие события, общителены. Эмоции у сангвиника преимущественно положительные, они быстро возникают и быстро сменяются. Вы быстро приспосабливаетесть к новым условиям и быстро сходитесть с людьми. Ваши чувства легко возникают и сменяются новыми, ему свойственна выразительность, но иногда и непостоянство." +

"Также И.Павлов считал, что у сангвиника возбудительный и тормозной процессы достаточно сильны, уравновешенны и подвижны. Он относил данный тип темперамента к сильному типу. При неблагоприятных условиях и отрицательных воспитательных влияниях эта подвижность может вылиться в отсутствие сосредоточенности, неоправданную поспешность поступков и поверхностность";

++SangCounter;

break;

case "Холерик":

ResultBlock.Text = "Ваш результат: " + name;

InfoBox.Text = "Человека вашего холерического темперамента можно охарактеризовать как быстрого, порывистого, способного отдаваться делу со страстностью, преодолевать значительные трудности, но, в то же время, неуравновешенного, склонного к бурным эмоциональным вспышкам и резким сменам настроения. Ваш темперамент характеризуется сильными, быстро возникающими чувствами, ярко отражающимися в речи, жестах и мимике." +

"Также ваш тип темперамента характеризуется высоким уровнем психической активности, энергичностью действий, резкостью, стремительностью, силой движений, их быстрым темпом, порывистостью. Холерик вспыльчив, нетерпелив, подвержен эмоциональным срывам, иногда бывает агрессивным." +

"И.Павлов относил холерика к сильному неуравновешенному типу. При отсутствии надлежащего воспитания недостаточная эмоциональная уравновешенность может привести к неспособности контролировать свои эмоции в трудных жизненных обстоятельствах.";

++HollCounter;

break;

case "Меланхолик":

ResultBlock.Text = "Ваш результат: " + name;

InfoBox.Text = "Меланхо́лик — один из четырёх типов темперамента, характеризующийся инертностью нервной системы, слабостью процессов возбуждения и торможения при их неуравновешенности (характерно преобладание процессов торможения), повышенной чувствительностью и высокой реактивностью психики." +

"Человека меланхолического темперамента можно охарактеризовать как легкоранимого, склонного глубоко переживать даже незначительные трудности и неудачи, но внешне вяло реагирующего на окружающее. Меланхолики отличаются высокой сенситивностью и эмоциональной чувствительностью, как следствие возникает легкоранимость. " +

"В некоторых сферах деятельности это является желательной личностной чертой, так как высокая эмоциональная чувствительность людей с меланхолическим темпераментом позволяет улавливать тончайшие эмоциональные выражения окружающих, особенно утончённых личностей. Для меланхоликов также характерно преобладание подавленного " +

"настроения и слабость волевой деятельности, проявляющаяся, например, в неуверенности в себе и нерешительности. При неблагоприятных условиях у них может развиться повышенная эмоциональная ранимость, замкнутость, отчуждённость. В спокойной, привычной для них обстановке, меланхолики могут успешно справляться с жизненными задачами.";

++MelLabel;

break;

case "Флегматик":

ResultBlock.Text = "Ваш результат: " + name;

InfoBox.Text = "Человека флегматического темперамента можно охарактеризовать как медлительного, спокойного, с устойчивыми стремлениями и более или менее постоянным настроением, со слабым внешним выражением душевных состояний. Флегматик отличается сильным, уравновешенным, инертным типом нервной системы. Такие люди имеют надёжную память, а их решение является продуманным, взвешенным и без риска." +

"У флегматика преобладает спокойное, ровное настроение; его чувства и эмоции обычно отличаются постоянством. Флегматик трудно переключается с одного вида деятельности на другой и приспосабливается к новой обстановке. Характерно то, что новые формы поведения у флегматика вырабатываются медленно, но долго сохраняются[2]. Он редко выходит из себя, не склонен к конфликтам; ему свойственны ровность, спокойствие, выдержка, иногда вялость, безразличие к окружающим, лень. Павлов относил данный тип темперамента к сильному типу." +

"При неблагоприятных условиях у флегматика может развиться вялость, бедность эмоций и склонность к выполнению однообразных привычных действий.";

++FlexLabel;

break;

}

flex.WriteLine(SangCounter);

flex.WriteLine(HollCounter);

flex.WriteLine(MelLabel);

flex.WriteLine(FlexLabel);

}

}

private void Button\_Click(object sender, RoutedEventArgs e) //метод сохранения отчёта

{

SaveFileDialog save = new SaveFileDialog();

save.Filter = "(.txt)|\*.txt";

if (save.ShowDialog() == true)

File.WriteAllText(save.FileName, InfoBox.Text);

}

private void Button\_Click\_1(object sender, RoutedEventArgs e) //переход в окно теста

{

BelovTest belovTest = new BelovTest(User);

belovTest.Show();

this.Close();

}

private void Button\_Click\_2(object sender, RoutedEventArgs e) //переход в главное окно

{

MainAfterAutoriz autoriz = new MainAfterAutoriz(User);

autoriz.Show();

this.Close();

}

}

}

//ResultWindow.xaml

<Window x:Class="Kursa4Wpf.ResultWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Kursa4Wpf"

mc:Ignorable="d"

Title="ResultWindow" Height="450" Width="800">

<Grid>

<Grid.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#FF8D2A2A" Offset="0"/>

<GradientStop Color="#FF7C00C3" Offset="1"/>

</LinearGradientBrush>

</Grid.Background>

<TextBlock x:Name="ResultBlock" HorizontalAlignment="Left" Margin="10,18,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Height="66" Width="238" Foreground="White" FontSize="22"/>

<TextBox x:Name="InfoBox" HorizontalAlignment="Left" Height="379" Margin="266,18,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="483" IsReadOnly="True" VerticalScrollBarVisibility="Auto"/>

<Button Content="Сохранить результат" HorizontalAlignment="Left" Margin="23,319,0,0" VerticalAlignment="Top" Width="203" Height="36" Click="Button\_Click"/>

<Button Content="Выйти в главное меню" HorizontalAlignment="Left" Margin="23,361,0,0" VerticalAlignment="Top" Width="203" Height="36" Click="Button\_Click\_2"/>

<Button Content="Пройти тест ещё раз" HorizontalAlignment="Left" Margin="23,278,0,0" VerticalAlignment="Top" Width="203" Height="36" Click="Button\_Click\_1"/>

</Grid>

</Window>

//RegisterWindow.xaml

<Window x:Class="Kursa4Wpf.RegisterWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Kursa4Wpf"

mc:Ignorable="d"

Title="RegisterWindow" Height="450" Width="350"

WindowStartupLocation="CenterScreen">

<Window.Resources>

<ControlTemplate x:Key="ButtonTemplate" TargetType="Button">

<Border BorderBrush="#FF412899" BorderThickness="3" CornerRadius="5"

TextBlock.Foreground="White"

TextBlock.TextAlignment="Center" VerticalAlignment="Center" HorizontalAlignment="Left">

<Border.Background>

<LinearGradientBrush>

<GradientStopCollection>

<GradientStop Color="#FF36C5B1"></GradientStop>

</GradientStopCollection>

</LinearGradientBrush>

</Border.Background>

<ContentPresenter RecognizesAccessKey="True"></ContentPresenter>

</Border>

</ControlTemplate>

</Window.Resources>

<Grid Cursor="Arrow" AutomationProperties.HelpText="Чтобы начать тест, нажмите" Margin="0,0,0,-7" VerticalAlignment="Center" Height="426">

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition Width="0\*"/>

</Grid.ColumnDefinitions>

<Grid.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#FFD000FF" Offset="0.948"/>

<GradientStop Color="#FF7C00C3" Offset="0.353"/>

</LinearGradientBrush>

</Grid.Background>

<TextBlock HorizontalAlignment="Left" Margin="43,10,0,372" TextWrapping="Wrap" VerticalAlignment="Center" Height="44" Width="268" FontFamily="Century Gothic" FontSize="40" Foreground="#FFFFBB07" Text="Регистрация" TextDecorations="{x:Null}" FontStyle="Italic" FontWeight="Bold"/>

<Button x:Name="buttonRegister" Content="Зарегестрироваться" Margin="43,267,32,105" VerticalAlignment="Center" Width="267" Height="54" Click="ButtonLogin\_Click" Background="#FF3BC1C7" HorizontalAlignment="Center" Cursor="Hand" FontSize="22" FontFamily="Arial" FontWeight="Bold" FontStyle="Italic"/>

<TextBox x:Name="userNameField" ToolTip="Введите имя" HorizontalAlignment="Left" Height="25" Margin="61,101,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="250" Cursor="IBeam"/>

<TextBox x:Name="loginField" HorizontalAlignment="Left" Height="25" Margin="61,134,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="250" Cursor="IBeam"/>

<PasswordBox x:Name="passField" HorizontalAlignment="Left" Margin="61,167,0,0" VerticalAlignment="Top" Width="250" Height="25"/>

<Label Content="Логин" HorizontalAlignment="Left" Margin="10,131,0,0" VerticalAlignment="Top" RenderTransformOrigin="0.58,0.489" FontSize="14" FontWeight="Bold" Foreground="#FF5EEAE3" Height="26" Width="51"/>

<Label Content="Пароль" HorizontalAlignment="Left" Margin="2,162,0,0" VerticalAlignment="Top" RenderTransformOrigin="0.58,0.489" FontSize="14" FontWeight="Bold" Background="{x:Null}" Foreground="#FF5EEAE3" Height="29" Width="62"/>

<Label Content="Имя" HorizontalAlignment="Left" Margin="22,98,0,0" VerticalAlignment="Top" RenderTransformOrigin="0.58,0.489" FontSize="14" Width="39" FontWeight="Bold" Background="{x:Null}" Foreground="#FF5EEAE3" Height="28"/>

<Button x:Name="ButtonExit" Content="←" HorizontalAlignment="Left" Margin="2,0,0,0" VerticalAlignment="Top" Width="26" Click="Button\_Click" Height="24" FontSize="16" Background="#FF0BB4F3"/>

</Grid>

</Window>

//MainWindow.xaml

<Window x:Class="Kursa4Wpf.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Kursa4Wpf"

mc:Ignorable="d"

Title="MainWindow" Height="450" Width="350" WindowStartupLocation="CenterScreen">

<Window.Resources>

<ControlTemplate x:Key="ButtonTemplate" TargetType="Button">

<Border BorderBrush="#FF412899" BorderThickness="3" CornerRadius="5"

TextBlock.Foreground="White"

TextBlock.TextAlignment="Center">

<Border.Background>

<LinearGradientBrush>

<GradientStopCollection>

<GradientStop Color="#FF36C5B1"></GradientStop>

</GradientStopCollection>

</LinearGradientBrush>

</Border.Background>

<ContentPresenter RecognizesAccessKey="True"></ContentPresenter>

</Border>

</ControlTemplate>

<Style x:Key="FocusVisual">

<Setter Property="Control.Template">

<Setter.Value>

<ControlTemplate>

<Rectangle Margin="2" SnapsToDevicePixels="true" Stroke="{DynamicResource {x:Static SystemColors.ControlTextBrushKey}}" StrokeThickness="1" StrokeDashArray="1 2"/>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

<SolidColorBrush x:Key="Button.Static.Background" Color="#FFDDDDDD"/>

<SolidColorBrush x:Key="Button.Static.Border" Color="#FF707070"/>

<SolidColorBrush x:Key="Button.MouseOver.Background" Color="#FFBEE6FD"/>

<SolidColorBrush x:Key="Button.MouseOver.Border" Color="#FF3C7FB1"/>

<SolidColorBrush x:Key="Button.Pressed.Background" Color="#FFC4E5F6"/>

<SolidColorBrush x:Key="Button.Pressed.Border" Color="#FF2C628B"/>

<SolidColorBrush x:Key="Button.Disabled.Background" Color="#FFF4F4F4"/>

<SolidColorBrush x:Key="Button.Disabled.Border" Color="#FFADB2B5"/>

<SolidColorBrush x:Key="Button.Disabled.Foreground" Color="#FF838383"/>

<Style x:Key="ButtonStyle1" TargetType="{x:Type Button}">

<Setter Property="FocusVisualStyle" Value="{StaticResource FocusVisual}"/>

<Setter Property="Background" Value="{StaticResource Button.Static.Background}"/>

<Setter Property="BorderBrush" Value="{StaticResource Button.Static.Border}"/>

<Setter Property="Foreground" Value="{DynamicResource {x:Static SystemColors.ControlTextBrushKey}}"/>

<Setter Property="BorderThickness" Value="1"/>

<Setter Property="HorizontalContentAlignment" Value="Center"/>

<Setter Property="VerticalContentAlignment" Value="Center"/>

<Setter Property="Padding" Value="1"/>

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type Button}">

<Border x:Name="border" CornerRadius="5" BorderBrush="{TemplateBinding BorderBrush}" BorderThickness="{TemplateBinding BorderThickness}" Background="{TemplateBinding Background}" SnapsToDevicePixels="true">

<ContentPresenter x:Name="contentPresenter" Focusable="False" HorizontalAlignment="{TemplateBinding HorizontalContentAlignment}" Margin="{TemplateBinding Padding}" RecognizesAccessKey="True" SnapsToDevicePixels="{TemplateBinding SnapsToDevicePixels}" VerticalAlignment="{TemplateBinding VerticalContentAlignment}"/>

</Border>

<ControlTemplate.Triggers>

<Trigger Property="IsDefaulted" Value="true">

<Setter Property="BorderBrush" TargetName="border" Value="{DynamicResource {x:Static SystemColors.HighlightBrushKey}}"/>

</Trigger>

<Trigger Property="IsMouseOver" Value="true">

<Setter Property="Background" TargetName="border" Value="{StaticResource Button.MouseOver.Background}"/>

<Setter Property="BorderBrush" TargetName="border" Value="{StaticResource Button.MouseOver.Border}"/>

</Trigger>

<Trigger Property="IsPressed" Value="true">

<Setter Property="Background" TargetName="border" Value="{StaticResource Button.Pressed.Background}"/>

<Setter Property="BorderBrush" TargetName="border" Value="{StaticResource Button.Pressed.Border}"/>

</Trigger>

<Trigger Property="IsEnabled" Value="false">

<Setter Property="Background" TargetName="border" Value="{StaticResource Button.Disabled.Background}"/>

<Setter Property="BorderBrush" TargetName="border" Value="{StaticResource Button.Disabled.Border}"/>

<Setter Property="TextElement.Foreground" TargetName="contentPresenter" Value="{StaticResource Button.Disabled.Foreground}"/>

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

</Window.Resources>

<Grid Cursor="Arrow" AutomationProperties.HelpText="Чтобы начать тест, нажмите" Margin="0,0,0,-1">

<Grid.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#FFD000FF" Offset="0.948"/>

<GradientStop Color="#FF7C00C3" Offset="0.353"/>

</LinearGradientBrush>

</Grid.Background>

<TextBlock HorizontalAlignment="Left" Margin="29,26,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Height="43" Width="289" FontFamily="Century Gothic" FontSize="40" Foreground="White" FontWeight="Bold" FontStyle="Italic"><Run Text="АВТОРИЗАЦИЯ"/><LineBreak/><Run/></TextBlock>

<Button x:Name="buttonLogin" Content="Войти" HorizontalAlignment="Left" Margin="53,224,0,0" VerticalAlignment="Top" Width="252" Height="51" Click="ButtonLogin\_Click" Cursor="Hand" Style="{DynamicResource ButtonStyle1}"/>

<TextBox x:Name="loginField" HorizontalAlignment="Left" Height="33" Margin="53,117,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="252"/>

<PasswordBox x:Name="passField" HorizontalAlignment="Left" Margin="53,164,0,0" VerticalAlignment="Top" Width="252" Height="33"/>

<Image HorizontalAlignment="Left" Height="33" Margin="10,117,0,0" VerticalAlignment="Top" Width="38" Source="Images/Login.png"/>

<Image HorizontalAlignment="Left" Height="33" Margin="10,164,0,0" VerticalAlignment="Top" Width="38" Source="Images/Password.png" RenderTransformOrigin="0.512,1.472"/>

<TextBlock x:Name="RegistrationBlock" HorizontalAlignment="Left" Style="{StaticResource HoverUnderlineStyle}" Margin="29,289,0,0" TextWrapping="Wrap" Text="Еще не зарегестрированы? Исправьте это!" VerticalAlignment="Top" Height="23" Width="298" FontWeight="Bold" FontSize="14" MouseLeftButtonDown="TextBlock\_MouseLeftButtonDown" Cursor="Hand"/>

<Label x:Name="HelpLabel" Content="?" HorizontalAlignment="Left" Margin="313,373,0,0" VerticalAlignment="Top" FontSize="20" FontWeight="Bold" MouseLeftButtonDown="HelpLabel\_MouseLeftButtonDown" Foreground="#FF1C4FBD" Cursor="Hand"/>

</Grid>

</Window>

# Приложение Б

(справочная)

Формы выходных документов

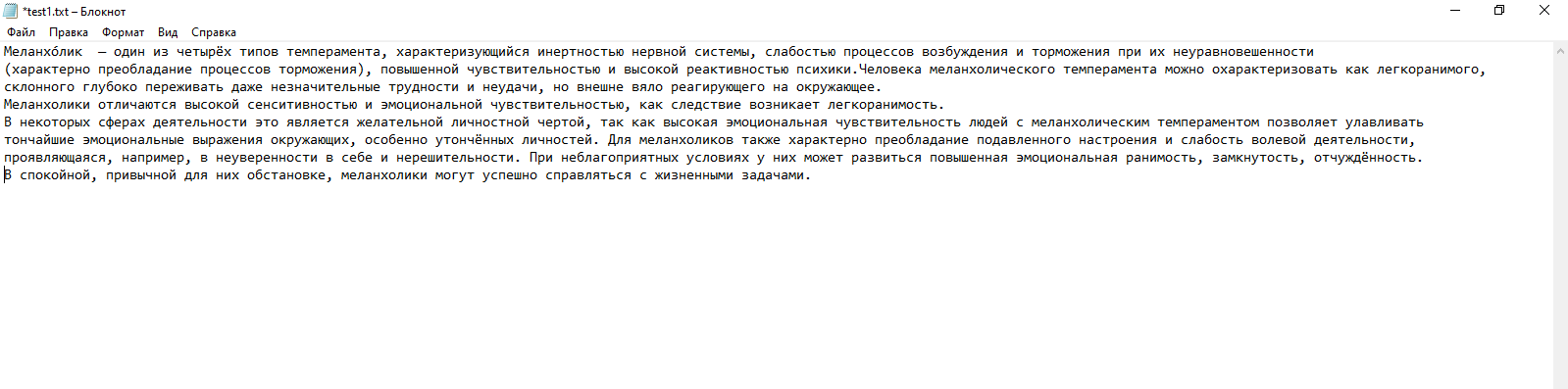


Рисунок Б.1.

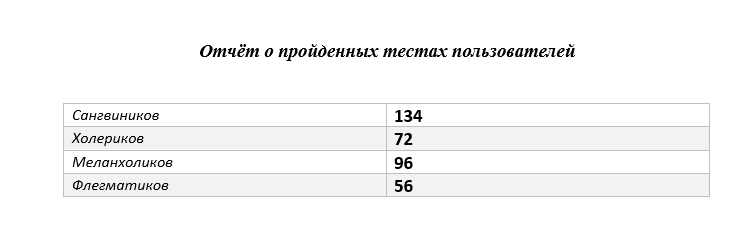


Рисунок Б.2