Частное учреждение образования

«Колледж бизнеса и права»

ПРОГРАММНОЕ СРЕДСТВО ДЛЯ АВТОМАТИЗАЦИИ РАБОЧЕГО МЕСТА ПСИХОЛОГА КОЛЛЕДЖА

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту по дисциплине

«Конструирование программ и языки программирования»

КП Т.991023.401 ПЗ

Руководитель проекта (В.Ю.Михалевич)

Учащийся (Д.Д.Якубовский)

2022

Содержание

Изм.

Изм.

Изм.

Изм.

Лист

Лист

Лист

Лист

№ докум.

№ докум.

№ докум.

№ докум.

Подпись

Подпись

Подпись

Подпись

Дата

Дата

Дата

Дата

Лист

Лист

Лист

Лист

3

3

3

3

КП Т.991023.401 ПЗ

КП Т.716007.401 ПЗ

КП Т.716009.401 ПЗ

КП Т.716007.401 ПЗ

Разраб.

Разраб.

Разраб.

Разраб.

*Якубовский Д.Д*

*Голубев И.В.*

*Гринь А.М.*

*Голубев И.В.*

Провер.

Провер.

Провер.

Провер.

Михалевич В.Ю.

Банцевич. С.В

Банцевич. С.В

Банцевич. С.В

Т. контр.

Т. контр.

Т. контр.

Т. контр.

Н. контр.

Н. контр.

Н. контр.

Н. контр.

Утверд.

Утверд.

Утверд.

Утверд.

*АРМ психолога колледжа*

Лит.

У

Лит.

У

Лит.

У

Лит.

У

Листов

Листов

Листов

Листов

47

50

55

50

КБП

КБП

КБП

КБП

[Введение 4](#_Toc41922226)

[1 Объектно-ориентированный анализ и проектирование систем 5](#_Toc41922227)

[1.1 Сущность задачи 5](#_Toc41922228)

[1.2 Проектирование модели 6](#_Toc41922229)

[2 Вычислительная система 9](#_Toc41922230)

[2.1 Требования к аппаратным и операционным ресурсам 9](#_Toc41922231)

[2.2 Инструменты разработки 9](#_Toc41922232)

[3 Проектирование задачи 11](#_Toc41922233)

[3.1 Требования к приложению 11](#_Toc41922234)

[3.2 Концептуальный прототип 11](#_Toc41922235)

[3.3 Организация данных 16](#_Toc41922236)

[3.4 Функции: логическая и физическая организация 18](#_Toc41922237)

[3.5 Проектирование справочной системы приложения 21](#_Toc41922238)

[4 Описание программного средства 22](#_Toc41922239)

[4.1 Общие сведения 22](#_Toc41922240)

[4.2 Функциональное назначение 22](#_Toc41922241)

[4.3 Входные и выходные данные 23](#_Toc41922242)

[5 Методика испытаний 24](#_Toc41922243)

[5.1 Технические требования 24](#_Toc41922244)

[5.2 Функциональное тестирование 24](#_Toc41922245)

[6 Применение 27](#_Toc41922246)

[6.1 Назначение программы 27](#_Toc41922247)

[6.2 Условия применение 27](#_Toc41922248)

[6.3 Справочная система 27](#_Toc41922249)

[Заключение 29](#_Toc41922266)

[Список информационных источников 30](#_Toc41922267)

[Приложение А Текст программных модулей 31](#_Toc41922268)

[Приложение Б Формы входных и выходных документов 45](#_Toc41922269)

[Приложение В Результаты работы программы 46](#_Toc41922270)

Введение

Психолог колледжа – сотрудник колледжа, следящий за морально психологическим состоянием учащихся. Его основной работой является учет студентов, имеющих какие-либо проблемы. Так же в обязанности психолога входит проведение так называемого «Анкетирование», которое по сути своей является просто тестом с определенным количеством вопросов.

Для учета информации по студентам на учете и простоты работы с этой информацией, психологу колледжа необходимо содержать всю информацию в одном месте. Ведение базы данных студентов с определёнными функциями и шаблонами документов – идеальное решение данной ситуации.

Целью курсового проекта на тему «Программное средство для автоматизации рабочего места психолога Колледжа Бизнеса и Права», предназначенной для автоматизации учета студентов у психолога, а также учета групп и курсов колледжа. Данная программа упростит ведение информации по студенту колледжа, с помощью которой можно будет вести учет студентов, курсов и их групп, а также создавать тесты выводя их в документ. После создания электронного документа есть возможность вывода документа на печать.

Решение поставленных задач отражено в пояснительной записке, которая состоит из шести разделов и содержит необходимую и достаточную информацию по использованию данного программного средства.

В первом разделе «Объектно-ориентированный анализ и проектирование системы» рассматривается сущность и актуальность поставленной задачи, описание существующих аналогов, проектирование модели, отображающей функциональную структуру объекта.

Второй раздел «Вычислительная система» содержит описание вычислительной системы: технические характеристики персонального компьютера, требования, которые будут предъявляться к персональному компьютеру, описание операционной системы, языка реализации и языка моделирования.

Третий раздел «Проектирование задачи» включает требования к программному средству, концептуальный прототип, логическую и физическую организацию данных в контексте среды разработки, описание функций пользователя в рамках среды разрабатываемого программного средства, проектирование справочной системы программного средства.

Четвертый раздел «Описание программного средства» отражает общие сведения о программе, функциональное назначение, структуру входных и выходных данных.

В пятом разделе «Методика испытаний» рассматриваются требования к техническим средствам для проведения испытания, требования к характеристикам программы применительно к условиям эксплуатации, требования к информационной и программной совместимости. Также описывается порядок проведения функционального тестирования.

Шестой раздел «Применение» содержит информацию, необходимую в процессе эксплуатации программного средства: его назначение, условия применения, а также приводится описание справочной системы.

В заключении описывается выполнение поставленной задачи, степень соответствия проектных решений задания, причины несоответствия, если таковые имеются.

В приложении А представлен текст программных модулей и в приложении Б – формы выходных документов.

Графическая часть представлена диаграммами вариантов использования, классов, последовательности, деятельности и компонентов. Диаграмма последовательности сделана для функции добавления нового студента на учет. Диаграмма деятельности сделана для всех функций программы.

1. Объектно-ориентированный анализ и проектирование систем
   1. Сущность задачи

Предметной областью решаемой задачи является учет работы психолога «Колледжа бизнеса и права».

Психолог колледжа – это сотрудник колледжа, занимающийся методической работой с учащимися и ведет учет студентов колледжа, которые обращаются к нему за помощью. Так же психолог колледжа для сбора статистической информации регулярно проводит тестирование за счет тестов, составленных им или полученных из других ресурсов. Помимо вышеперечисленного от психолога требуется регулярно подавать отчеты по тем или иным критериям.

Из выше описанной информации можно сделать вывод что психолог выполняет следующие функции:

1. создание тестов;
2. сбор статистической информации по средству тестирования учащихся;
3. создание и подача отчетов;
4. учет обратившихся к нему учащихся;
5. хранение архива учащихся, которые были сняты с учета.

Психологу колледжа необходимо знать следующую информацию о приходящих к нему студентах, а именно:

1. Ф.И.О;
2. группа;
3. курс;
4. количество полных лет.

Психолог колледжа разрабатывает тесты, которые обязательно будут содержать в себе название теста, а после него уже будут идти вопросы и ответы на них в количестве которое выбрал психолог перед составлением теста.

Для налаженной работы психологического кабинета. Для четкой координации работы, ориентирования среди студентов колледжа, групп, курсов, отслеживания студентов, состоящих на учете, нужен специалист, владеющий всей общей информацией, компетентный и в случае проблемы имеющий возможность поставить приблизительный диагноз студенту или отправить его к более компетентному врачу.

На основании исследованной предметной области программное средство должно выполнять следующие функции:

1. добавление студента на учет;
2. редактирование информации о студенте, состоящем на учете;
3. полностью удалить студента с учета;
4. перенос студента в архив;
5. возвращение студента на учет;
6. создание тестов;
7. создание отчетов по выбранным критериям;
8. поиск студента, состоящего на учете по определенным критериям;
9. сортировка списка студентов, состоящих на учете по различным критериям.

В качестве аналогов для данного программного средства можно выделить программы, использующиеся психологами других учебных заведений. Был рассмотрен 1 аналог данного программного средства. В ходе ознакомления с ним было выделено что аналог имеет менее лаконичный и более усложненный интерфейс, а также рассмотренный аналог имеет менее широкий функционал по сравнению с разработанным программным средством.

* 1. Проектирование модели

Цель моделирования данных состоит в обеспечении разработчика информационной системы концептуальной схемой базы данных в форме одной или нескольких локальных моделей, которые относительно легко могут быть отображены в любую систему баз данных [3].

Наиболее распространенным средством моделирования данных является диаграмма «Сущность-связи» (ERD). С их помощью определяется важные для предметной области объекты (сущности), их свойства (атрибуты) и отношения друг с другом (связи). ERD непосредственно используются для проектирования реляционных баз данных. Нотация ERD была впервые введена П. Ченном и получился дальнейшее развитие в работах Баркера.

Диаграмма «Сущность-связи» в нотации П.Чена представлена на рисунке 1.

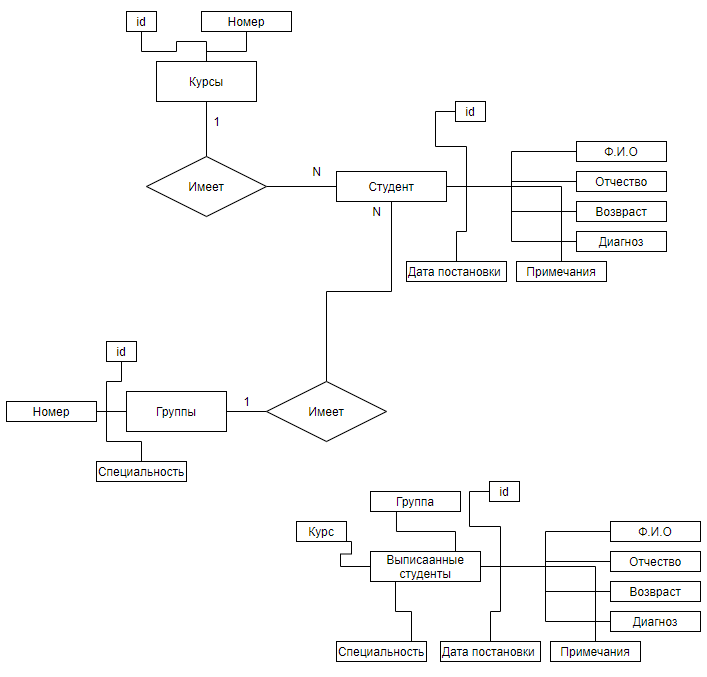


Рисунок 1

Исходя из предметной области можно выделить следующие сущности разработки: «Группа», «Курс», «Студент», «Студент в архиве».

Для сущности «Группа» атрибутами будут являться:

* id;
* номер группы;
* специальность.

Для сущности «Курс» атрибутами будут являться:

* id;
* номер курса.

Для сущности «Студент» атрибутами будут являться:

* id;
* имя;
* фамилия;
* отчество;
* возраст;
* диагноз;
* примечания;
* дата постановки на учет.

Для сущности «Студент в архиве» атрибутами будут являться:

* id;
* имя;
* фамилия;
* отчество;
* возраст;
* диагноз;
* примечания;
* дата постановки на учет;
* группа;
* курс.

Суть диаграммы вариантов использования заключается в том, что проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью, так называемых вариантов использования.

К «extend» относится входная информация:

* о справке;
* об БД;
* о редактировании информации;
* о добавлении студента.

К «include» относится:

* поиск студентов;
* возвращение студента на учет;
* очистка архива студентов;
* сортировка списка студентов;
* удаление студента с учета;
* сброс результатов поиска или сортировки;
* выбор количества вопросов;
* ввод названия тестов;
* выбор категории отчета.

Диаграмма вариантов использования представлена на листе 1 графической части.

Диаграмма классов служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. Диаграмма может отражать различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывать их внутреннюю структуру и типы отношений. Для организации колледж бизнеса и права в программном средстве реализовано 11 классов которые содержат в себе методы, благодаря которым работают основные функции программы и происходит работа с базой данных. Выше описанные классы представлены на листе 2 графической части.

При моделировании поведения проектируемой или анализируемой системы возникает необходимость детализировать особенности алгоритмической и логической реализации выполняемых системой операций. Для моделирования процесса выполнения операций в языке UML используются так называемые диаграммы деятельности.

Диаграммы деятельности – частный случай диаграмм состояний. Основная цель использования таких диаграмм – визуализация особенностей реализации операций классов, когда необходимо представить алгоритмы их выполнения. Диаграмма деятельности представлена на листе 3 графической части.

Временной аспект поведения имеет существенное значение при моделировании синхронных процессов, описывающих взаимодействия объектов. Именно для этой цели и используются диаграммы последовательности, в которых ключевым моментом является динамика взаимодействия объектов во времени. Например, сотрудник переходит с вкладки на вкладку и добавляет данные, что является временным аспектом поведения сотрудника. Он включает программу, вводит данные, сохраняет данные – последовательность действий. Диаграмма последовательности представлена на листе 4 графической части.

Диаграмма компонентов описывает объекты реального мира – компоненты программного обеспечения. Эта диаграмма позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами, в роли которых может выступать исходный, бинарный и исполняемый код.

Определим компоненты для данного программного средства и построим диаграмму компонентов. Главным исполняемым файлом является файл «Kursovik.exe». Этот файл является результатом компиляции проекта «Kursovik.sln». Компонент «Kursovik.sln» реализован в среде разработки Visual Studio 2019. В состав компонента «Kursovik.sln» входят следующие файлы:

* файлы проекта с расширениями \*.cs, \*.xaml, \*.cd, \*.config;
* файлы отчетов \*.docx, реализованные в текстовом процессоре MS Word;
* файл базы данных «accounting.db», созданные в SQLite.

Вид диаграммы компонентов для данной проектируемой системы представлен в графической части на листе 5.

1. Вычислительная система
   1. Требования к аппаратным и операционным ресурсам

Минимальными требованиями к аппаратному и программному обеспечению, необходимыми для корректной работы программы являются:

* операционная система Windows 7 и выше;
* процессор Intel i31600 МГц и выше;
* объем оперативной памяти 2 ГБ и более;
* мышь;
* монитор;
* принтер;
* клавиатура.

Наиболее удобной операционной системой для проведения испытаний является Windows 10, так как она ориентирована на максимальное использование всех возможностей персонального компьютера, сетевых ресурсов и обеспечение комфортных условий работы.

На компьютере должны быть установлены программы Microsoft Office Word 2016 и NET.Framework 4.0 и выше.

* 1. Инструменты разработки

Ниже описаны инструменты разработки, которые будут использоваться для написания программного средства.

Ноутбук со следующей аппаратной конфигурацией:

* процессор Intel Core i5 10300H 2500 МГц;
* оперативная память DDR4 8 ГБ;
* видеокарта GeForce GTX 1650;
* винчестер SSD 255 ГБ;
* клавиатура и мышь.

Программное обеспечение:

* операционная система Windows 10;
* среда программирования Visual Studio 2019 c использованием интерфейса программирования приложений WPF;
* язык программирования C#;
* система управления базами данных (СУБД) DB Browser(SQLite);
* программная платформа Microsoft .NET Framework 4.8;
* офисный пакет приложений Microsoft Office (MS Office) 2016;
* программа для построения диаграмм DRAW.IO;
* средство для создания установочных пакетов Smart Install Maker.

Операционная система – это набор управляющих программ, предназначенных для управления ресурсами вычислительной системы как единого комплекса, другими словами, операционная система – это набор программного обеспечения, который обеспечивает работу компьютера.

При разработке программного средства использовалась операционная система Windows 10, так как на данный момент эта операционная система является самой распространённой операционной системой. В Windows 10 были исправлены практически все недостатки предыдущих операционных систем. Аппаратные требования Windows 10 скромнее, она способна работать даже на маломощных компьютерах и ещё добавлено множество функций, существенно облегчающих работу за компьютером.

Visual Studio 2019 – это лучший инструмент для разработки любого приложения под любую платформу. Система управления версиями в этом выпуске делает разработку гибкой, а совместную работу – эффективной [13].

WPF(Windows Presentation Foundation– это набор различных управляемых библиотек, с помощью которых вы можете выполнить все необходимые для оконного приложения действия, начиная от обмена сообщениями с операционной системой для отслеживания любых событий клиентского окна, заканчивая диалоговыми системами, связью с другими компьютерами по сети и многими другими возможностями [13].

Язык программирования C# – чрезвычайно мощный язык, содержащий средства создания эффективных программ практически любого назначения, от низкоуровневых утилит и драйверов до сложных программных комплексов самого различного назначения [12].

СУБД DB Browser(SQLite) – основная СУБД для рабооты с базами данных SQLite. Версия 3.18 – это наиболее стабильная и многофункциональная СУБД: более быстрое выполнение запросов, удобный интерфейс, ценные сведения на любом устройстве, расширенная аналитика, новые технологии безопасности и новые варианты использования при базы данных. DB Browser предоставляет передовые функции для решения задач высокой важности за счет выполнения операций в памяти и встроенной операционной аналитики [9].

SQLite – это хорошо масштабируемая, полностью реляционная, быстродействующая и простая база данных, способная обрабатывать большие объемы данных для приложений [9].

.NET Framework – [программная платформа](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B0#%D0%BA%D1%80%D0%BE%D1%81%D1%81%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), выпущенная компанией [Microsoft](https://ru.wikipedia.org/wiki/Microsoft) в [2002 году](https://ru.wikipedia.org/wiki/2002_%D0%B3%D0%BE%D0%B4). Основой платформы является общеязыковая среда исполнения [Common Language Runtime (CLR)](https://ru.wikipedia.org/wiki/Common_Language_Runtime), которая подходит для разных языков программирования. Функциональные возможности CLR доступны в любых языках программирования, использующих эту среду [6].

Microsoft Office 2016 – офисный пакет приложений, созданных корпорацией Microsoft для операционных систем Microsoft Windows [8].

DRAW.IO – это инструмент UML (Unified Modeling Language) с открытым исходным кодом на основе JavaScript, предназначенный для обучения унифицированному языку моделирования и для быстрого создания диаграмм UML. Это инструмент моделирования, а не инструмент рисования, поскольку есть базовый словарь и каталог объектов многократного использования.

Smart Install Maker – это мощное и удобное средство для создания инсталляторов. Программа обладает удобным и интуитивно понятным интерфейсом, а также полным набором необходимых функций для создания профессиональных инсталляторов с минимальным размером, высокой степенью сжатия файлов и приятным интерфейсом [10].

1. Проектирование задачи
   1. Требования к приложению

Разрабатываемое программное средство должно иметь понятный и удобный в использовании интерфейс, чтобы взаимодействие между программой и пользователем было максимально упрощено.

Согласно общим требованиям, графический интерфейс разрабатываемого приложения должен:

* ориентироваться на пользователя, который общается с программой на внешнем уровне взаимодействия;
* сохранять стандартизированное назначение и местоположение на экране графических объектов, работающих в среде Windows.

Интерфейс программного приложения будет разрабатываться с учетом общих требований к пользовательскому интерфейсу.

Существуют общие требования, которые предъявляются к программным приложениям:

* соответствие стандартам организации интерфейса: использование многооконного подхода, реализация управления работой программного приложения с помощью элементов управления;
* требования к выбранной цветовой схеме рабочего экрана;
* выполнение одной функции с помощью разных элементов управления.

Пользователю необходимо установить пакет Microsoft Office 2016 для работы с документами и переноса отчетов в документ Microsoft Office Word 2016.

Для обучения пользователя необходимо разработать справочную систему, в которой должны быть раскрыты все аспекты работы с программой, возможные трудности, возникшие во время работы и пути их решения.

Для удобства работы пользователя с программным средством необходимо при разработке форм придерживаться единого стиля оформления. Формы не должны быть перегружены излишней информацией или содержать информацию, не относящуюся к данной форме. Также необходимо предусмотреть защиту данных от удаления и изменения, а также от ввода некорректных данных. В случае ввода некорректных данных или попытке совершить запрещенные действия, пользователь должен быть проинформирован о своих действиях с помощью диалоговых окон.

* 1. Концептуальный прототип

Концептуальный прототип состоит из описания внешнего пользовательского интерфейса, а именно, элементов управления.

При создании данного приложения важную роль играют окна, так как они являются основным средством работы с пользователем. Разрабатываемое программное средство будет содержать несколько окон.

Все окна будут содержать стандартные пользовательские элементы управления.

В рабочем режиме программного средства, пользователю, для удобной навигации, будет предоставлено меню.

Первым окном, которое появляется сразу после запуска программы является окно меню, содержащее в себе кнопки, отвечающие за переход по окнам, которое содержат в себе основные функции программы. Окно содержит четыре кнопки. Макет окна «Меню» представлен на рисунке 2.



Рисунок 2

При нажатии на кнопку «Список студентов» пользователю будет открыто окно для работы со списком студентов на учете. Оно будет содержать поиск по различным критериям, различные сортировки, и работы с данными.

Макет формы «Список студентов» представлен на рисунке 3.

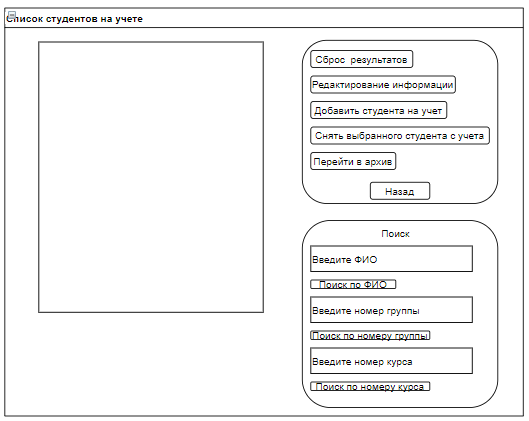


Рисунок 3

При нажатии на кнопку «Сброс результатов не будет осуществляется какого-либо перехода по страница будет вызвано обновление таблицы содержащей записи из Допри нажатии на кнопку «Редактирование информации» пользователю будет открыто окно «Редактирование», в котором по средству введения новых данных вместо старых которыми будут заполнены текстовые поля на окне редактирование.

Макет окна «Редактирование» представлен на рисунке 4.

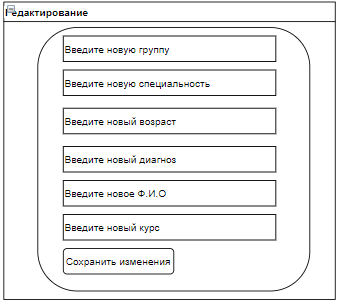


Рисунок 4

При нажатии на кнопку добавить студента на учет пользователю будет открыто окно содержащее текстовые поля для заполнения информацией о новом студенте и комбо боксы в которых пользователь сможет выбрать информацию из других баз данных которая так же нужна для добавления нового пользователя.

Макет окна «Добавление студента» представлен на рисунке 5.

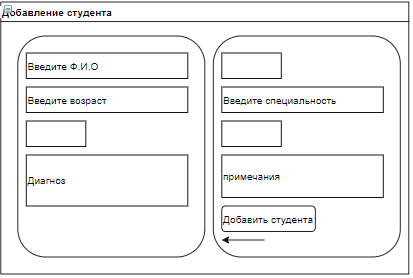


Рисунок 5

Так же в комбо боксах в окне «Добавление студента» будут содержатся кнопки для добавления курса и для добавления группы. При нажатии на эти кнопки пользователю будут открыты соответствующие окна, на которых он сможет добавить новый курс или новую группу, а также пользователь будет иметь возможность удалить уже существующую группу.

Макеты окон «Добавления курса» и добавления группы представлены на рисунках 6 и 7.

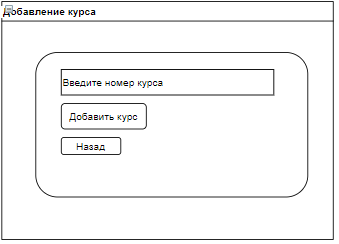


Рисунок 6

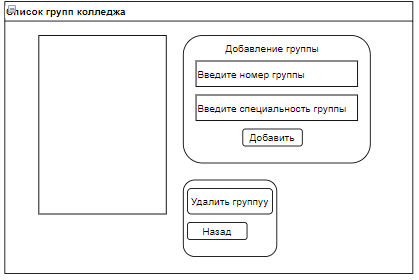


Рисунок 7

При нажатии на кнопку «Перейти в архив выбранных студентов» на окне «Список студентов на учете». Пользователю будет открыто окно содержащее архив студентов, снятых с учета, но не до конца удаленных. В это окно будет содержать основной функционал для работы с архивом и для того что бы вернуть студента обратно на учет.

Макет окна «Архив студентов» представлен на рисунке 8.

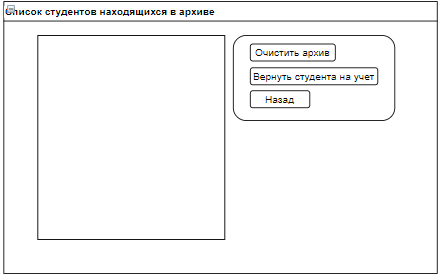


Рисунок 8

Нажав на кнопку «Редактор тестов» на окне меню пользователю будет открыто окно подготовки создания теста, на котором он должен ввести тему теста и выбрать количество вопросов, которые будет содержать тест из предложенных ему вариантов.

Макет окна «Редактор тестов» представлен на рисунке 9.

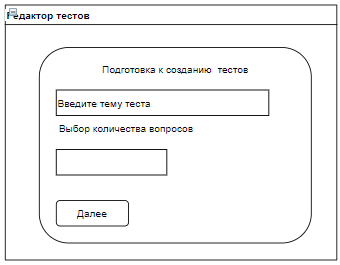


Рисунок 9

Нажав на кнопку «Далее» на окне «Редактор тестов» пользователю будет открыто окно, содержащее текстовые поля, которые он должен будет заполнить согласно подсказкам, которые содержатся в них, пользователю нужно будет заполнить названия вопросов и варианты ответов на вопрос и после заполнения всех вышеперечисленных полей нажать кнопку «Сохранить» для создания документа с тестом.

Макет окна «Создание тестов» представлен на рисунке 10.

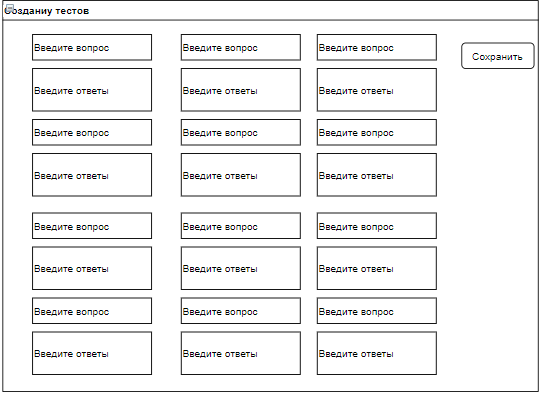


Рисунок 10

Нажав на кнопку «Создание отчетов» в окне «Меню» пользователю будет открыто окно, на котором ему нужно будет выбрать критерий отчета т.е. таблицу по которой будет создан файл отчета для дальнейшей печати. и нажать на кнопку «Создать».

Макет окна «Составление отчетов представлен на рисунке 11.

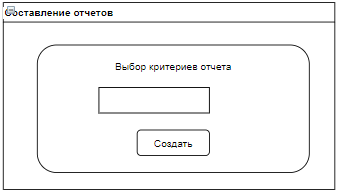


Рисунок 11

* 1. Организация данных

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

Реляционная модель основана на математическом 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 внешние ключи. Достоинством реляционной модели является простота и удобство физической реализации.

Структура базы данных разрабатываемого программного средства включает четыре таблицы которые содержат описание полей и так же описание смысловой нагрузки этих полей. Описание всех таблиц в базе данных приводится в таблицах 1-4.

Таблица «Курсы» хранит информацию о курсах колледжа, структура которой представлена в таблице 1.

Таблица 1 – Структура таблицы «Курсы»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип  поля | Размер поля,  байт | Описание поля |
| id | int | 4 | Идентификатор курса |
| Номер курса | int | 4 | Номер курса колледжа |

Таблица «Группы» хранит информацию об группах и специальностях, которые существуют в колледже, структура которой представлена в таблице 2.

Таблица 2 – Структура таблицы «Группы»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип  поля | Размер поля,  байт | Описание поля |
| Id | int | 4 | Идентификатор группы |
| Номер группы | text | 50 | Номер группы |
| Специальность | text | 20 | Наименование специальности к которой относится группа |

Таблица «Студенты» хранит информацию о студентах, состоящих на учете у психолога колледжа, структура которой представлена в таблице 3.

Таблица 3 – Структура таблицы «Студенты»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип  поля | Размер поля,  байт | Описание поля |
| id | int | 4 | Идентификатор студента |
| ФИО | text | 50 | ФИО студента |
| Группа | text | 20 | Группа студента |
| Пол | text | 10 | Пол студента |
| Возраст | int | 4 | Возраст студента |
| Специальность | text | 15 | Специальность на которую обучается студент |
| Курс | int | 2 | Курс на котором находится студент |
| Диагноз | text | 50 | Диагноз поставленный студенту |
| Примечания | text | 50 | Примечания касаемо данного студента |
| Дата | text | 15 | Дата постановки студента на учет |

Таблица «Архив студентов» хранит информацию о студентах, которые были сняты с учета и находятся в архиве. Структура таблицы представлена в таблице 4.

Таблица 4 – Структура таблицы «Архив студентов»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип  поля | Размер поля,  байт | Описание поля |
| id | int | 4 | Идентификатор студента |
| ФИО | text | 50 | ФИО студента |
| Группа | text | 20 | Группа студента |
| Пол | text | 10 | Пол студента |
| Возраст | int | 4 | Возраст студента |
| Специальность | text | 15 | Специальность на которую обучается студент |

Продолжение таблицы 4

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип  поля | Размер поля,  байт | Описание поля |
| Курс | int | 2 | Курс на котором находится студент |
| Диагноз | text | 50 | Диагноз поставленный студенту |
| Примечания | text | 50 | Примечания касаемо данного студента |
| выписки | text | 15 | Дата снятия студента с учета психолога |

Физическая схема базы данных представлена на рисунке 12.

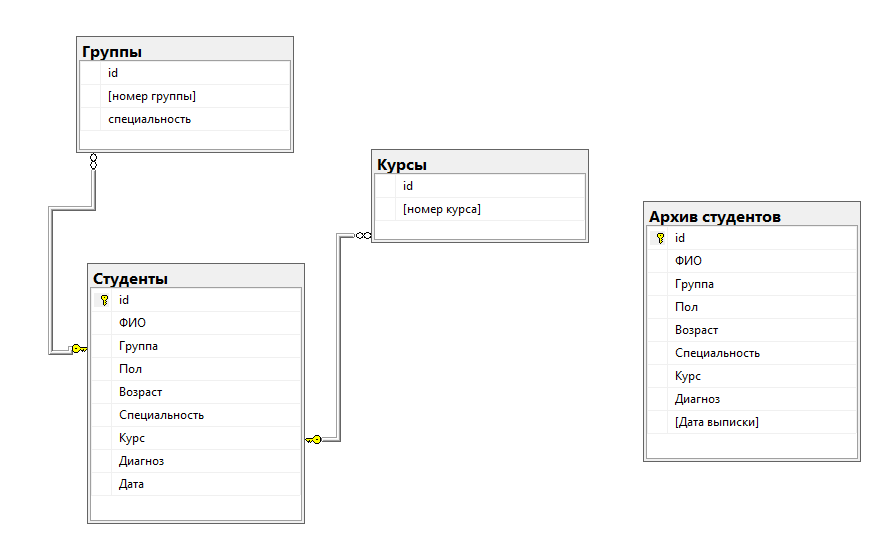


Рисунок 12

* 1. Функции: логическая и физическая организация

На основании диаграммы вариантов использования в программном средстве «Kursovik.exe» реализованы функции переноса студента в архив, создания отчета по определенным группам студентов и создания документа с тестом.

Рассмотрим данные функции программы.

Для того, удалить студента с учета с дальнейшим его переносом в архив пользователю сначала нужно будет в таблице студентов выбрать нужного студента и нажать на кнопку снять с учета и в всплывающем диалоговом окне необходимо дать согласие. После того как пользователь дает согласие программа сныча берет все данные из выбранной строки в таблице и распределяет их по соответствующим переменным. После этого программа создает запрос для вставки данных в таблицу «Архив студентов» при помощи команды INSERT вставляя выше полученные данные. После того как произошло добавление данных в таблицу данные надо будет удалить из основной таблицы, для этого вызывается функция Delete() в которой сначала происходит удаление выбранной строки из таблицы в программе. После удаления используется запрос DELETE который удаляет выбранную строку из таблицы в БД по ФИО. После того как произошли все удаления что бы вывести пользователю обновленные данные вызывается функция WindowLoaded() в которой происходит выборка всех данных их таблицы студенты и вывод их в таблицу в программе. Код вышеперечисленных функций представлен ниже.

public void Delete()

{

DataRowView dataRow = (DataRowView)StudentssGrid.SelectedItem;

string a = Convert.ToString(dataRow.Row.ItemArray[0]);

DataRowView row = (DataRowView)StudentssGrid.SelectedItem;

row.Row.Delete();

SQLiteConnection conn = new SQLiteConnection("Data Source=accounting.db");

conn.Open();

SQLiteCommand cmd = conn.CreateCommand();

string sql\_delete = $"DELETE FROM Students WHERE ФИО = '{a}'";

cmd.CommandText = sql\_delete;

try

{

cmd.ExecuteNonQuery();

}

catch (SQLiteException ex)

{

MessageBox.Show(ex.Message);

}

conn.Close();

Window\_Loaded(null, null);

}

public void Transport(object sender, RoutedEventArgs e)

{

MessageBoxResult result = MessageBox.Show("Переместить запись в архив?", " ", MessageBoxButton.YesNoCancel, MessageBoxImage.Question);

if (result == MessageBoxResult.Yes)

{

DataRowView dataRow = (DataRowView)StudentssGrid.SelectedItem;

string sur = Convert.ToString(dataRow.Row.ItemArray[0]);

string Gender = Convert.ToString(dataRow.Row.ItemArray[1]);

string Group = Convert.ToString(dataRow.Row.ItemArray[2]);

string spec = Convert.ToString(dataRow.Row.ItemArray[4]);

string age = Convert.ToString(dataRow.Row.ItemArray[3]);

string course = Convert.ToString(dataRow.Row.ItemArray[5]);

string diagnosis = Convert.ToString(dataRow.Row.ItemArray[6]);

string notes = Convert.ToString(dataRow.Row.ItemArray[7]);

MessageBox.Show($"{sur}, {Gender}, {Group},{spec},{age},{course},{diagnosis},{notes}");

SQLiteConnection sqlite\_conn = new SQLiteConnection("Data Source=accounting.db;Version=3;New=False;Compress=True;");

sqlite\_conn.Open();

SQLiteCommand sqlite\_cmd = sqlite\_conn.CreateCommand();

DateTime obj = DateTime.Now;

DateTime dateOnly = obj.Date;

string date = dateOnly.ToShortDateString();

sqlite\_cmd.CommandText =

$@"INSERT INTO archieve

(

ФИО,Группа,Пол,Возвраст,Специальность,Курс,Диагноз,Примечания,Выписан

)

VALUES

('{sur}', '{Gender}', '{Group}',{age},'{spec}',{course},'{diagnosis}','{notes}','{date}')";

sqlite\_cmd.ExecuteNonQuery();

sqlite\_conn.Close();

Delete();

}

else if (result == MessageBoxResult.No)

{

Delete();

}

public void Window\_Loaded(object sender, RoutedEventArgs e)

{

string baseName = "accounting.db";

DataSet dataSet = new DataSet();

string sql = @"SELECT \* FROM Students";

SQLiteFactory factory = (SQLiteFactory)DbProviderFactories.GetFactory("System.Data.SQLite");

using (SQLiteConnection connection = (SQLiteConnection)factory.CreateConnection())

{

connection.ConnectionString = "Data Source = " + baseName;

connection.Open();

using (SQLiteDataAdapter dataAdapter = new SQLiteDataAdapter(sql, connection))

{

dataAdapter.Fill(dataSet);

StudentssGrid.ItemsSource = dataSet.Tables[0].DefaultView;

StudentssGrid.Columns[9].Visibility = Visibility.Hidden;

}

connection.Close();

}

Для того что бы создать отчет в программе пользователю необходимо выбрать критерий отчета из списка и нажать кнопку «Создать отчет» после этого программа выполнит запрос на выборку данных из таблицы, которую выбрал пользователь эти данные будут помещены в Dataset по средству использоваться библиотеки они будут занесены в Microsoft Word 2016. Код вышеперечисленных функций программы представлен ниже.

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

{

MessageBox.Show($"Отчет был создан по слеждующему пути {Environment.CurrentDirectory}//export.doc");

SourseGrid.SelectAllCells();

SourseGrid.ClipboardCopyMode = DataGridClipboardCopyMode.IncludeHeader;

ApplicationCommands.Copy.Execute(null, SourseGrid);

SourseGrid.UnselectAllCells();

var result = (string)Clipboard.GetData(DataFormats.Text);

dynamic wordApp = null;

try

{

var sw = new StreamWriter("export.doc");

sw.WriteLine(result);

sw.Close();

//var proc = Process.Start("export.doc");

Type wordType = Type.GetTypeFromProgID("Word.Application");

wordApp = Activator.CreateInstance(wordType);

wordApp.Documents.Add(System.AppDomain.CurrentDomain.BaseDirectory + "export.doc");

wordApp.ActiveDocument.Range.ConvertToTable(1, SourseGrid.Items.Count, SourseGrid.Columns.Count);

}

catch (Exception ex){if (wordApp != null){wordApp.Quit();}}

Для того, чтобы пользователь мог составить документ, содержащий тест, который в последующем можно будет распечатать. Пользователю потребуется сначала перейти на вкладку «Создание отчетов». Пользователю откроется окно, в котором он должен будет обязательно выбрать тему теста и количество вопросов в тесте из предложенного списка и нажать кнопку «Далее» после этого ему откроется окно, в котором он должен будет заполнить содержимое теста т.е название вопросов и ответы на эти вопросы. После заполнения всех данных пользователь должен будет нажать кнопку сохранить. После нажатия данной кнопки программа запусти череду функций первая из которых получит из системы дату и время в последствии отбросив время и получив только дату. После этого программа откроет шаблон теста, который находится в корневой папке программы. После того как шаблон будет открыт программа считает все закладки из шаблона и по этим закладкам будут распределены данные которые ввел пользователь и документ будет сохранен на рабочем столе пользователя. Код вышеперечисленных функций программы представлен ниже.

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

{

Tests obj = new Tests();

var dateAndTime = DateTime.Now;

var date1 = dateAndTime.ToShortDateString();

string date = date1.ToString();

\_Document oDoc = GetDoc($"{Environment.CurrentDirectory}\\TestPattern.doc");

oDoc.SaveAs(FileName: "C:\\Users\\" + Environment.UserName + "\\Desktop\\Test.docx");

oDoc.Close();

MessageBox.Show("Тест был создан и находится на рабочем столе");

}

private \_Document GetDoc(string path) //Вот этот ГетДок

{

\_Document oDoc = oWord.Documents.Add(path);

SetTemplate(oDoc);

return oDoc;

}

private void SetTemplate(Microsoft.Office.Interop.Word.\_Document oDoc)

{

Tests obj1 = new Tests();

oDoc.Bookmarks["A1"].Range.Text = Convert.ToString(A1.Text);

oDoc.Bookmarks["A10"].Range.Text = Convert.ToString(A10.Text);

oDoc.Bookmarks["A11"].Range.Text = Convert.ToString(A11.Text);

oDoc.Bookmarks["A12"].Range.Text = Convert.ToString(A12.Text);

oDoc.Bookmarks["A2"].Range.Text = Convert.ToString(A2.Text);

oDoc.Bookmarks["A3"].Range.Text = Convert.ToString(A3.Text);

oDoc.Bookmarks["A4"].Range.Text = Convert.ToString(A4.Text);

oDoc.Bookmarks["A5"].Range.Text = Convert.ToString(A5.Text);

oDoc.Bookmarks["A6"].Range.Text = Convert.ToString(A6.Text);

oDoc.Bookmarks["A7"].Range.Text = Convert.ToString(A7.Text);

oDoc.Bookmarks["A8"].Range.Text = Convert.ToString(A8.Text);

oDoc.Bookmarks["A9"].Range.Text = Convert.ToString(A9.Text);

oDoc.Bookmarks["Name"].Range.Text = Convert.ToString(s);

oDoc.Bookmarks["Q1"].Range.Text = Convert.ToString(Q1.Text);

oDoc.Bookmarks["Q10"].Range.Text = Convert.ToString(Q10.Text);

oDoc.Bookmarks["Q11"].Range.Text = Convert.ToString(Q11.Text);

oDoc.Bookmarks["Q12"].Range.Text = Convert.ToString(Q12.Text);

oDoc.Bookmarks["Q2"].Range.Text = Convert.ToString(Q2.Text);

oDoc.Bookmarks["Q3"].Range.Text = Convert.ToString(Q3.Text);

oDoc.Bookmarks["Q4"].Range.Text = Convert.ToString(Q4.Text);

oDoc.Bookmarks["Q5"].Range.Text = Convert.ToString(Q5.Text);

oDoc.Bookmarks["Q6"].Range.Text = Convert.ToString(Q6.Text);

oDoc.Bookmarks["Q7"].Range.Text = Convert.ToString(Q7.Text);

oDoc.Bookmarks["Q8"].Range.Text = Convert.ToString(Q8.Text);

oDoc.Bookmarks["Q9"].Range.Text = Convert.ToString(Q9.Text);

}

Полный текст программы представлен в приложении А.

* 1. Проектирование справочной системы приложения

Для эффективной работы пользователя с приложением необходимо обеспечить его качественной справочной системой, в которой должны быть приведены методы и приемы работы с приложением, включающие данные о том, что произойдет после нажатия на определенную кнопку или при выборе пункта меню, сведения о том, какую информацию и в каком виде следует вводить в соответствующие поля, каким образом можно работать.

Справочная система будет представлена в виде встроенного файла в приложении, содержащая информацию по эксплуатации программного средства. Файл будет разработан в программе MS Word 2016.

Система справки данного программного средства будет содержать два раздела. Первым из них будет раздел «О разработчике». В данном разделе будет содержатся общая информация о разработчике и о руководителе проекта.

Вторым разделом справки будет «О программе». В данном разделе программы пользователю будет предоставляется информация о программе в частности о целях ее создания, о том при помощи каких инструментов была создана данная программа и с какой целью. Так же в данном разделе справочной системы будет содержатся информация о том, как использовать те или иные функции программы, что следует нажимать и что следует вводить в определённых местах программы, как вызвать те или иные сортировки.

1. Описание программного средства
   1. Общие сведения

Программное средство «Kursovik» разрабатывалось с целью автоматизации рабочего места психолога Колледжа Бизнеса и Права и улучшения эффективности работы за счет автоматизации многих аспектов его работы, а также благодаря простому, лаконичному интерфейсу и функциональным возможностям приложения. Установка данного программного обеспечения позволяет проделывать сложные операции без ошибок. Данная программа была разработана на персональном компьютере со следующей конфигурацией:

* процессор Intel Core i5-10300H 2.5 Ггц;
* оперативная память DDR4 8 ГБ;
* встроенная видеокарта NVidia GTX 1660.

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

В программе присутствует база данных «accounting.db». Программа не требовательна к системным ресурсам, также проста в использовании и не требует специальных навыков при работе. Для работы данного программного средства необходима предварительная установка и настройка следующих программных продуктов:

* текстовый редактор MS Word 2016;
* платформа Microsoft Net Framework версии 4.8.

В состав программных входят:

* программа «Kursovik.exe»;
* база данных «accounting.db»;
* файл справочной системы «info.docx»;
* файл тестовой системы «TestPattern.docx»

Объем установленного приложения не превышает 300 МБ, файлов – 150 МБ.

Программное средство не требует логина и пароля.

* 1. Функциональное назначение

Программное средство разработано с целью автоматизировать рабочее место психолога колледжа за счет автоматизации таких аспектов его работы таких как анкетирование, различные учеты и поддержка составления отчетности. Данное программное средство имеет простой и лаконичный интерфейс что делает работу с ним достаточно легкой и не требующей специальных навыков.

Так же программа помогает решать некоторое задачи упрощая ведение и просмотр большого объёма данных. Такие функции программы как архив студентов и список учащихся помогают избежать огромного количества бумажной волокиты за счет переноса всего учета в электронный формат.

Программа использует стандартные элементы управления, такие как кнопки, меню, списка, поля ввода, что обеспечивает единство интерфейса системы и программного средства.

В работе программного средства предусмотрены некоторые ситуации, которые должны предупреждать пользователя, чтобы он выполнял все необходимые требования по эксплуатации программы. Для этого существуют сообщения системы, например, если пользователь повторно пытается добавить уже существующий курс или группу, или другие данные, которые не должны повторяться, то в этом случае на экран выводится сообщение с информацией о характерной ошибке.

Таким образом, программа может применяться в реальных условиях, представляя собой достаточно удобный помощник в работе.

* 1. Входные и выходные данные

Входными являются данные, введённые пользователем в таблицы базы данных: «Курсы», «Группы», «Студенты».

Входными данным при добавлении данных о новом курсе будут являться:

* номер курса.

Входными данными при добавлении данных о новой группе являются:

* специальность;
* номер группы.

Входными данными при добавлении данных о новом студенте являются:

* ФИО;
* группа;
* пол;
* возраст:
* специальность;
* курс;
* диагноз;
* примечания.

Вся вносимая в программное средство информация хранится в локальной базе данных.

Выходные данные – это сформированные списки, а также тесты и отчеты, которые необходимы для работы психолога.

Отчет обо всех студентах, находящихся в архиве представлен в приложении Б на рисунке Б.1.

Сформированный тест размерностью восемь вопросов представлен в приложении Б на рисунке Б.2.

1. Методика испытаний
   1. Технические требования

Основными минимальными требованиями, выдвигаемыми к аппаратному обеспечению персонального компьютера, являются:

* процессор тактовой частотой не ниже 1800 МГц;
* оперативная память DDR4 4 ГБ;
* встроенная видеокарта Intel HD Graphics 620;
* винчестер SSD 250 ГБ;
* клавиатура и мышь.

Для работы с программой необходимо наличие монитора.

Компьютер должен работать под управлением операционной системы, начиная с Windows 7 и выше. Необходимо наличие Microsoft Office 2016 и Microsoft .NET Framework 4.0 и выше.

* 1. Функциональное тестирование

Функциональное тестирование – это тестирование функций приложения на соответствие требованиям и проводится для выявления неполадок и недочетов программы на этапе ее сдачи в эксплуатацию.

Проведем тестирование проверки каждого пункта меню с целью проверки всех функций.

Тестирование программы будет производиться последовательно, переходя из одной части программы в другую. Во время теста будут проверяться все действия с программой, навигация пунктам меню, которые может произвести пользователь. После чего, все собранные и найденные ошибки будут исправлены [5].

В таблице 5 представлены тест-кейсы, подготовленные для проведения функционального тестирования. Работа каждого ниже перечисленного тест кейса будет осуществляться с формы содержащей список студентов на учете, представленной на рисунке 13.

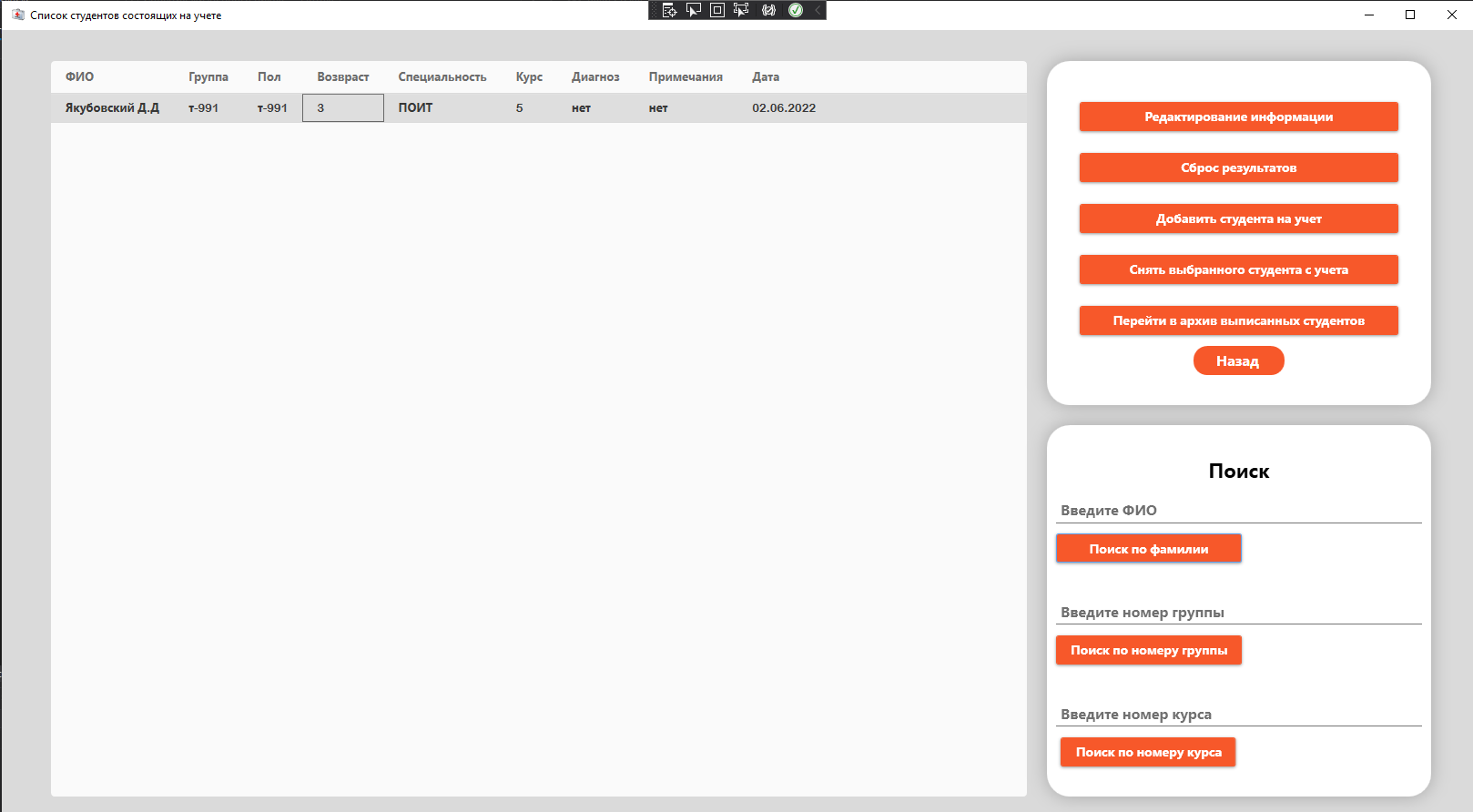


Рисунок 13

Таблица 5 – Тест-кейсы для формы работы с студентами на учете

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Модуль / Функция | Шаги выполнения | Ожидаемый результат | Фактический результат |
| 1 | Удаление студента с учета без переноса в архив | 1. Выбрать нужного студента из списка.  2.Нажать на кнопку «Снять выбранного студента с учета».  3. Во всплывшем диалоговом окне нажать на кнопку «Нет». | Удаление записи о студенте из таблицы в программе и удаление записи о нем из БД. | Запись о выбранном студенте удалена из программы и из БД,после удаления происходит обновление информации в таблице в программе |
| 2 | Перенос студента в архив | 1. Выбрать нужного студента из списка.  2.Нажать на кнопку «Снять выбранного студента с учета».  3. Во всплывшем диалоговом окне нажать на кнопку «Да».  4.Нажать на кнопку «Перейти в архив выбранных студентов». | Запись с информацией о студенте будет перемещена в архив со студентами. | Запись о студенте будет находится в архиве. Результат представлен в приложении В рисунке В.1. |
| 3 | Редактирование информации о выбранном студенте | 1. Выбрать нужного студента из списка.  2. Нажать на кнопку «Редактирование информации».  3. Изменить любое поле.  4.Нажать на кнопку «Сохранить изменения». | Информация о студенте будет отредактирована и изменения будут сохранены. | Информация о студенте будет изменена. Пример первоначальной записи о студенте представлен в приложении В на рисунке В.2 а запись о студенте после редактирования представлена на рисунке В.3. |
| 4 | Сортировка студентов по курсу по возрастанию | 1. На форме, содержащей список студентов на учете нажать на заголовок в таблице под названием «курс». | Вывод отсортированного списка студентов по полю курс по возрастанию. | Пользователю будет выведен отсортированный список, Пример списка после сортировки представлен в приложении В на рисунке В.4. |
| 5 | Поиск студента по Ф.И.О | 1.На форме, содержащей список студентов на учете заполнить поле «Ф.И.О».  2.Нажать на кнопку «Поиск по Ф.И.О». | Пользователю будут выведены все студенты имеющие введенное Ф.И.О. | Пользователю будут выведены результаты подходящие по введенные критерии |

Продолжение таблицы 5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Модуль / Функция | Шаги воспроизведения | Ожидаемый результат | Фактический результат |
| 6 | Поиск несуществующего студента по Ф.И.О | 1.На форме, содержащей список студентов на учете заполнить поле «Ф.И.О» данными студента, не состоящего на учете.  2. Нажать на кнопку «Поиск по Ф.И.О». | Пользователю будет выведено сообщение о том что по введенному Ф.И.О ничего не найдено. | Пользователю будет выведено сообщение о том что ничего не найдено, пример сообщения представлен в приложении В на рисунке В.5. |
| 7 | Добавление студента на учет | 1.На форме, содержащей список студентов на учете нажать на кнопку «Добавить студента на учет».  2.В открывшемся окне заполнить все поля.  3.Нажать на кнопку добавить. | Студент будет добавлен на учет. | Студент будет добавлен на учет и пользователю будет выведено сообщение о том, что студент добавлен. Пример сообщения будет представлен в приложении В на рисунке В.6. |
| 8 | Сброс результатов различных действий на таблицей в программе | 1. На форме, содержащей список студентов на учете нажать на кнопку «Сброс результатов». | Пользователю будет выведена исходная таблица в БД. | Вывод пользователю изначальной таблицы из БД со студентами на учете. |
| 9 | Возвращение студента на основной учет из архива | 1.На форме, содержащей список студентов на учете нажать на кнопку «Перейти в архив выбранных студентов».  2.В открывшемся окне выбрать нужного студента из списка.  3.Нажать на кнопку «Вернуть студента на учет».  4.Нажать на кнопку «Назад». | Запись будет перенесена из архива на основной учет без изменений. | Запись с информацией о студенте будет перенесена на основной учет. Пример на котором запись находится в архиве представлена в приложении В на рисунке В.7,а пример того как выглядит таблица со студентами после переноса записи будет представлен на рисунке В.8. |
| 10 | Поиск студента по существующему Ф.И.О | 1.На форме, содержащей список студентов на учете заполнить поле «Ф.И.О» данными студента, состоящего на учете.  2. Нажать на кнопку «Поиск по Ф.И.О». | Пользователю будет выведен список студентов подходящих по критерии поиска | Пользователю будут выведены результаты поиска. Пример списка студентов до поиска представлен в приложении В на рисунке В.9 а пример результатов поиска представлен на рисунке В.10 |

1. Применение
   1. Назначение программы

Приложение «Kursovik.exe» предназначено для автоматизации рабочего места психолога Колледжа Бизнеса и Права. Программное средство предоставляет пользователю возможности для поиска, отображения и редактирования имеющихся в базе студентов, добавления курсов и добавления и удаления групп, уменьшает затраты времени используемого при составлении отчетности. Осуществляет формирование отчетов о студентах на учете и в архиве. Программное средство формирует тесты с дальнейшей возможностью их печати.

* 1. Условия применение

Для работы с программой необходимо наличие программного обеспечения:

* операционная система, начиная с Windows 7;
* библиотека Microsoft .NET Framework 4.8;
* офисный пакет приложений Microsoft Office (MS Office) 2016;
* наличие интернета со скоростью не ниже 8-10 Мбит/с.

На случай редактирования проекта программы необходимо наличие программного обеспечения:

* СУБД DB browser(SQLite);
* среда разработки приложений Microsoft Visual Studio 2019.
  1. Справочная система

Справочная система представлена в виде встроенного файла в приложении, содержащая информацию по эксплуатации программного средства. Для полноценной и корректной работы со справочной системой необходимо просто прочитать ее.

Раздел справки в программе представлен 2 разделами в которых рассказывается как о разработке программы, так и о самой программе, и о принципах ее использования так же во втором разделе справки описываются элементы управления и как с ними взаимодействия так же расписывается как осуществить ту или иную работу в программе по типу добавления студента или создания теста. Так же данный раздел содержит информацию о том, как была разработана программа и какие цели преследовались при ее разработке.

Второй раздел справки представляет собой информацию о том, кто является автором программы и под чьим руководством была разработана программа.

Структура файла со справкой представлена на рисунке 14.

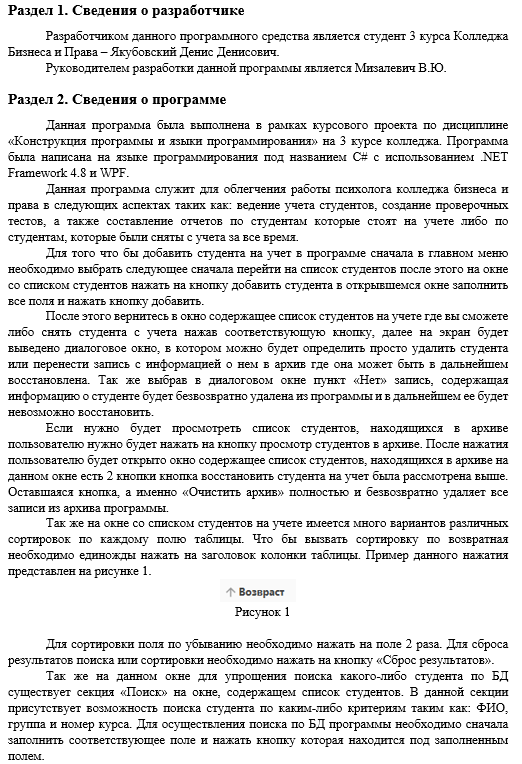


Рисунок 14

Заключение

В рамках курсового проекта на тему «АРМ психолога колледжа» было разработано программное средство «kursovik.exe», автоматизирующее часть работы сотрудника.

Для достижения целей 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.

Разработка имеет интуитивно понятный графический интерфейс, позволяющий даже с минимальным знанием компьютера использовать данное программного средство.

Программа реализована в полном объеме и в соответствии с заданными требованиями, полностью отлажена и протестирована. Поставленные задачи выполнены.

К достоинствам данного программного средства можно отнести: интуитивно-понятный интерфейс, простоту в использовании, хорошую оптимизацию, большой внутренний функционал, удобное создание отчетов и тестов, большие возможности в учете студентов.

В процессе разработки данного программного средства были применены и закреплены знания по уже изученному материалу, были отработаны навыки владения методами надежного программирования и эффективности разработки программного обеспечения в Microsoft Visual Studio 2019 с использованием языка программирования С# и технологии WPF, разработана база данных средствами системы управления базами данных DB Browser(SQLite).

Программное средство готово к практическому использованию психологом Колледжа Бизнеса и Права.

Данная программа может быть дополнена и модернизирована.

Список информационных источников

1. Багласова, Т.Г. Методические указания к курсовому проектированию для учащихся специальности 2-40 01 01 «Программное обеспечение информационных технологий»/В.Ю.Михалевич. – Минск : КБП, 2021. – 23 c.
2. Общие требования к тестовым документам : ГОСТ 2.105-95. – Введ. 01.01.1996 – Минск : Межгос. совет по стандартизации, метрологии и сертификации, 1995. – 84 с.
3. .NET Framework [Электронный ресурс]. – Microsoft, 2022. – Режим доступа : https://docs.microsoft.com/en-us/dotnet/framework/. – Дата доступа : 23.04.2022.
4. MS Word 2016 [Электронный ресурс]. – Microsoft, 2022. – Режим доступа : https://support.microsoft.com/ru-ru/word. – Дата доступа : 11.05.2020.
5. MS Office [Электронный ресурс]. – Microsoft, 2022. – Режим доступа : https://officeapplications.net/microsoft-word/. – Дата доступа : 24.04.2022.
6. DB Browser (SQLite) [Электронный ресурс]. – DB Browser, 2022. – Режим доступа : https://sqlitebrowser.org/. – Дата доступа : 17.04.2022.
7. Smart Install Maker [Электронный ресурс]. – InstallBuilders, 2022. – Режим доступа : https://www.ixbt.com/news/soft/index.shtml?10/52/79/. – Дата доступа : 16.05.2022.
8. DRAW.IO [Электронный ресурс]. – Draw.IO, 2022. – Режим доступа : https://app.diagrams.net/. – Дата доступа : 20.04.2022.
9. C# [Электронный ресурс]. – Microsoft, 2022. – Режим доступа : https://docs.microsoft.com/en-us/dotnet/csharp/. – Дата доступа : 18.04.2022.
10. Visual Studio [Электронный ресурс]. – Microsoft, 2022. – Режим доступа : https://visualstudio.microsoft.com/vs/. – Дата доступа : 18.04.2022.

Приложение А

(обязательное)

Текст программных модулей

public partial class AddCourse : Window

{

public SQLiteConnection sqlite\_conn;

public SQLiteCommand sqlite\_cmd;

public AddCourse()

{

InitializeComponent();

sqlite\_conn = new SQLiteConnection($"Data Source = {Environment.CurrentDirectory}\\accounting.db");

sqlite\_cmd = sqlite\_conn.CreateCommand();

}

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

{

AddStudent obj = new AddStudent();

obj.Show();

Close();

}

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

{

sqlite\_conn.Open();

sqlite\_cmd.CommandText = $"INSERT INTO Course (номер) VALUES ({Convert.ToInt32(num.Text)})";

sqlite\_cmd.ExecuteNonQuery();

sqlite\_conn.Close();

MessageBox.Show("Курс был успешно добавлен");

}

}

public partial class AddGroup : Window

{

public SQLiteConnection sqlite\_conn;

public SQLiteCommand sqlite\_cmd;

public AddGroup()

{

sqlite\_conn = new SQLiteConnection($"Data Source = {Environment.CurrentDirectory}\\accounting.db");

sqlite\_cmd = sqlite\_conn.CreateCommand();

InitializeComponent();

}

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

{

sqlite\_conn.Open();

sqlite\_cmd.CommandText = $"INSERT INTO 'Group' (номер,специальность) VALUES ('{TextBoxNum.Text}','{TextBoxSpec.Text}')";

sqlite\_cmd.ExecuteNonQuery();

sqlite\_conn.Close();

MessageBox.Show("Группа была успешно добавлена");

Window\_Loaded(null, null);

}

private void Window\_Loaded(object sender, RoutedEventArgs e)

{

string baseName = "accounting.db";

DataSet dataSet = new DataSet();

string sql = @"SELECT \* FROM 'Group'";

SQLiteFactory factory = (SQLiteFactory)DbProviderFactories.GetFactory("System.Data.SQLite");

using (SQLiteConnection connection = (SQLiteConnection)factory.CreateConnection())

{

connection.ConnectionString = "Data Source = " + baseName;

connection.Open();

using (SQLiteDataAdapter dataAdapter = new SQLiteDataAdapter(sql, connection))

{

dataAdapter.Fill(dataSet);

StudentssGrid.ItemsSource = dataSet.Tables[0].DefaultView;

}

connection.Close();

}

}

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

{

AddStudent obj = new AddStudent();

obj.Show();

Close();

}

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

{

DataRowView dataRow = (DataRowView)StudentssGrid.SelectedItem;

string i = Convert.ToString(dataRow.Row.ItemArray[0]);

string a = Convert.ToString(dataRow.Row.ItemArray[1]);

DataRowView row = (DataRowView)StudentssGrid.SelectedItem;

row.Row.Delete();

sqlite\_conn.Open();

sqlite\_cmd.CommandText = $"DELETE FROM 'Group' WHERE номер = {i} AND специальность = {a}";

sqlite\_cmd.ExecuteNonQuery();

sqlite\_conn.Close();

MessageBox.Show("Группа была успешно удалена");

Window\_Loaded(null, null);

}

}

public partial class AddStudent : Window

{

public string s;

public int a;

public string g;

AppContext db;

public AddStudent()

{

InitializeComponent();

db = new AppContext();

}

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

{

StudentsList studentsList = new StudentsList();

studentsList.Show();

Close();

}

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

{

string studentName = Name.Text;

int age = Convert.ToInt32(Age.Text);

string gender = g;

string diagnosis = Diagnosis.Text;

string speciality = Spec.Text;

string notes = Notes.Text;

Student student = new Student(studentName, s, gender, age, speciality, a, diagnosis, notes);

db.students.Add(student);

db.SaveChanges();

Clear();

MessageBox.Show("Студент был успешно добавлен");

}

public void Clear()

{

Name.Clear();

Age.Clear();

Diagnosis.Clear();

Spec.Clear();

Notes.Clear();

}

private void GroupsList\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

ComboBox comboBox = (ComboBox)sender;

string selectedItem = (string)comboBox.SelectedItem;

s = Convert.ToString(selectedItem.ToString());

}

private void CourseList\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

ComboBox comboBox = (ComboBox)sender;

string selectedItem = (string)comboBox.SelectedItem;

a = Convert.ToInt32(selectedItem.ToString());

}

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

{

AddGroup obj = new AddGroup();

obj.Show();

Close();

}

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

{

AddCourse obj = new AddCourse();

obj.Show();

Close();

}

private void GenderList\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

ComboBox comboBox = (ComboBox)sender;

ComboBoxItem selectedItem = (ComboBoxItem)comboBox.SelectedItem;

g = selectedItem.Content.ToString();

}

private void Window\_Loaded(object sender, RoutedEventArgs e)

{

SQLiteConnection conn = new SQLiteConnection($"Data Source = { Environment.CurrentDirectory}\\accounting.db");

SQLiteCommand command = new SQLiteCommand("SELECT номер fROM Course", conn);

conn.Open();

DbDataReader reader = command.ExecuteReader();

while (reader.Read())

{

CourseList.Items.Add((string)reader["номер"]);

}

conn.Close();

GroupsLoad();

}

public void GroupsLoad()

{

SQLiteConnection conn = new SQLiteConnection($"Data Source = { Environment.CurrentDirectory}\\accounting.db");

SQLiteCommand command = new SQLiteCommand("SELECT номер fROM 'Group'", conn);

conn.Open();

DbDataReader reader = command.ExecuteReader();

while (reader.Read())

{

GroupsList.Items.Add((string)reader["номер"]);

}

conn.Close();

}

}

public partial class Edit : Window

{

public string n;

public Edit()

{

InitializeComponent();

}

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

{

string dataSource = "accounting.db";

using (SQLiteConnection connection = new SQLiteConnection())

{

connection.ConnectionString = "Data Source=" + dataSource;

connection.Open();

using (SQLiteCommand command = new SQLiteCommand(connection))

{

command.CommandText =

$@"UPDATE Students SET ФИО='{Name.Text}',Группа='{Group.Text}',Возвраст='{Age.Text}',Специальность='{Spec.Text}',Курс='{Course.Text}',Диагноз='{Diagnosis.Text}' WHERE ФИО = '{n}'; ";

command.ExecuteNonQuery();

if (command.ExecuteNonQuery() == 1)

{

MessageBox.Show("Информация обновлена");

}

}

StudentsList obj = new StudentsList();

obj.Show();

Close();

}

}

}

public partial class Menu : Window

{

public Menu()

{

InitializeComponent();

}

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

{

StudentsList studentsList = new StudentsList();

studentsList.Show();

Close();

}

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

{

Tests tests = new Tests();

tests.Show();

Close();

}

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

{

Otchet otchet = new Otchet();

otchet.Show();

Close();

}

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

{

Process.Start($"{Environment.CurrentDirectory}\\info.docx");

}

}

public partial class Otchet : Window

{

public Otchet()

{

InitializeComponent();

}

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

{

MessageBox.Show($"Отчет был создан по слеждующему пути {Environment.CurrentDirectory}//export.doc");

SourseGrid.SelectAllCells();

SourseGrid.ClipboardCopyMode = DataGridClipboardCopyMode.IncludeHeader;

ApplicationCommands.Copy.Execute(null, SourseGrid);

SourseGrid.UnselectAllCells();

var result = (string)Clipboard.GetData(DataFormats.Text);

dynamic wordApp = null;

try

{

var sw = new StreamWriter("export.doc");

sw.WriteLine(result);

sw.Close();

//var proc = Process.Start("export.doc");

Type wordType = Type.GetTypeFromProgID("Word.Application");

wordApp = Activator.CreateInstance(wordType);

wordApp.Documents.Add(System.AppDomain.CurrentDomain.BaseDirectory + "export.doc");

wordApp.ActiveDocument.Range.ConvertToTable(1, SourseGrid.Items.Count, SourseGrid.Columns.Count);

}

catch (Exception ex)

{

if (wordApp != null)

{

wordApp.Quit();

}

}

}

private void Window\_Loaded(object sender, RoutedEventArgs e)

{

SourseGrid.Visibility = Visibility.Hidden;

}

private void ChoiseTable\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (ChoiseTable.SelectedIndex == 0)

{

RefreshTable("Дата","Students");

}

else

{

RefreshTable("Выписан","Archieve");

}

}

private void RefreshTable(string param,string dataTable)

{

string baseName = "accounting.db";

DataSet dataSet = new DataSet();

string sql = $"SELECT \* FROM {dataTable}";

SQLiteFactory factory = (SQLiteFactory)DbProviderFactories.GetFactory("System.Data.SQLite");

using (SQLiteConnection connection = (SQLiteConnection)factory.CreateConnection())

{

connection.ConnectionString = "Data Source = " + baseName;

connection.Open();

using (SQLiteDataAdapter dataAdapter = new SQLiteDataAdapter(sql, connection))

{

dataAdapter.Fill(dataSet);

SourseGrid.ItemsSource = dataSet.Tables[0].DefaultView;

}

connection.Close();

}

}

}

internal class Student

{

public int id { get; set; }

public string ФИО { get; set; }

public string Группа { get; set; }

public string Пол { get; set; }

public int Возвраст { get; set; }

public string Специальность { get; set; }

public int Курс { get; set; }

public string Диагноз { get; set; }

public string Примечания { get; set; }

public string Дата { get; set; }

public Student() { }

public Student(string nName,string nGroup,string nGender,int nAge,string nSpec,int nCourse,string nDiagnosis,string nNotes)

{

ФИО = nName;

Группа = nGroup;

Пол = nGender;

Возвраст = nAge;

Специальность = nSpec;

Курс = nCourse;

Диагноз = nDiagnosis;

Примечания = nNotes;

DateTime obj = DateTime.Now;

DateTime dateOnly = obj.Date;

Дата = dateOnly.ToShortDateString();

}

}

public partial class StudentsArchieve : Window

{

public StudentsArchieve()

{

InitializeComponent();

}

private void Window\_Loaded(object sender, RoutedEventArgs e)

{

string baseName = "accounting.db";

DataSet dataSet = new DataSet();

string sql = @"SELECT \* FROM archieve";

SQLiteFactory factory = (SQLiteFactory)DbProviderFactories.GetFactory("System.Data.SQLite");

using (SQLiteConnection connection = (SQLiteConnection)factory.CreateConnection())

{

connection.ConnectionString = "Data Source = " + baseName;

connection.Open();

using (SQLiteDataAdapter dataAdapter = new SQLiteDataAdapter(sql, connection))

{

dataAdapter.Fill(dataSet);

ArchieveGrid.ItemsSource = dataSet.Tables[0].DefaultView;

ArchieveGrid.Columns[9].Visibility = Visibility.Hidden;

}

connection.Close();

}

}

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

{

StudentsList obj = new StudentsList();

obj.Show();

Close();

}

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

{

SQLiteConnection conn = new SQLiteConnection("Data Source=accounting.db");

conn.Open();

SQLiteCommand cmd = conn.CreateCommand();

string sql\_delete = "DELETE FROM archieve;";

cmd.CommandText = sql\_delete;

try

{

cmd.ExecuteNonQuery();

}

catch (SQLiteException ex)

{

Console.WriteLine(ex.Message);

}

conn.Close();

Window\_Loaded(null, null);

}

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

{

Transport(sender,e);

}

public void Transport(object sender, RoutedEventArgs e)

{

DataRowView dataRow = (DataRowView)ArchieveGrid.SelectedItem;

string sur = Convert.ToString(dataRow.Row.ItemArray[0]);

string Gender = Convert.ToString(dataRow.Row.ItemArray[1]);

string Group = Convert.ToString(dataRow.Row.ItemArray[2]);

string spec = Convert.ToString(dataRow.Row.ItemArray[4]);

int age = Convert.ToInt32(dataRow.Row.ItemArray[3]);

int course = Convert.ToInt32(dataRow.Row.ItemArray[5]);

string diagnosis = Convert.ToString(dataRow.Row.ItemArray[6]);

string notes = Convert.ToString(dataRow.Row.ItemArray[7]);

SQLiteConnection sqlite\_conn = new SQLiteConnection("Data Source=accounting.db;Version=3;New=False;Compress=True;");

sqlite\_conn.Open();

SQLiteCommand sqlite\_cmd = sqlite\_conn.CreateCommand();

DateTime obj = DateTime.Now;

DateTime dateOnly = obj.Date;

string date = dateOnly.ToShortDateString();

sqlite\_cmd.CommandText =

$@"INSERT INTO students

(

ФИО,

Группа,

Пол,

Возвраст,

Специальность,

Курс,

Диагноз,

Примечания,

Дата

)

VALUES

(

'{sur}',

'{Group}',

'{Gender}',

{age},

'{spec}',

{course},

'{diagnosis}',

'{notes}',

'{date}'

)";

sqlite\_cmd.ExecuteNonQuery();

sqlite\_conn.Close();

Delete();

}

public void Delete()

{

DataRowView dataRow = (DataRowView)ArchieveGrid.SelectedItem;

string i = Convert.ToString(dataRow.Row.ItemArray[0]);

DataRowView row = (DataRowView)ArchieveGrid.SelectedItem;

row.Row.Delete();

SQLiteConnection conn = new SQLiteConnection("Data Source=accounting.db");

conn.Open();

SQLiteCommand cmd = conn.CreateCommand();

string sql\_delete = $"DELETE FROM archieve WHERE ФИО = '{i}';";

cmd.CommandText = sql\_delete;

try

{

cmd.ExecuteNonQuery();

}

catch (SQLiteException ex)

{

Console.WriteLine(ex.Message);

}

conn.Close();

Window\_Loaded(null, null);

}

}

public partial class StudentsList : Window

{

public string i;

AppContext DB = new AppContext();

public StudentsList()

{

InitializeComponent();

}

private void GoBackLabel\_MouseDoubleClick(object sender, MouseButtonEventArgs e)

{

Menu menu = new Menu();

menu.Show();

Close();

}

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

{

AddStudent student = new AddStudent();

student.Show();

Close();

}

public void Window\_Loaded(object sender, RoutedEventArgs e)

{

string baseName = "accounting.db";

DataSet dataSet = new DataSet();

string sql = @"SELECT \* FROM Students";

SQLiteFactory factory = (SQLiteFactory)DbProviderFactories.GetFactory("System.Data.SQLite");

using (SQLiteConnection connection = (SQLiteConnection)factory.CreateConnection())

{

connection.ConnectionString = "Data Source = " + baseName;

connection.Open();

using (SQLiteDataAdapter dataAdapter = new SQLiteDataAdapter(sql, connection))

{

dataAdapter.Fill(dataSet);

StudentssGrid.ItemsSource = dataSet.Tables[0].DefaultView;

StudentssGrid.Columns[9].Visibility = Visibility.Hidden;

}

connection.Close();

}

}

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

{

if(!Check())

{

MessageBox.Show("Не выбрана запись!");

}

else

{

Transport(sender, e);

}

}

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

{

Window\_Loaded(sender, e);

}

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

{

StudentsArchieve archieve = new StudentsArchieve();

archieve.Show();

Close();

}

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

{

string param = TextBoxName.Text;

string value = "ФИО";

SearchFunc(value, param);

}

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

{

string param = TextBoxGroup.Text;

string value = "Группа";

SearchFunc(value, param);

}

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

{

string param = TextBoxCourse.Text;

string value = "Курс";

SearchFunc(value, param);

}

public void SearchFunc(string Value,string param)

{

if(!SeasrchCheck(param))

{

MessageBox.Show("Нужно заполнить поле");

}

else

{

try

{

string baseName = "accounting.db";

DataSet dataSet = new DataSet();

string sql = $@"SELECT\*FROM Students WHERE {Value}='{param}'";

SQLiteFactory factory = (SQLiteFactory)DbProviderFactories.GetFactory("System.Data.SQLite");

using (SQLiteConnection connection = (SQLiteConnection)factory.CreateConnection())

{

connection.ConnectionString = "Data Source = " + baseName;

connection.Open();

using (SQLiteDataAdapter dataAdapter = new SQLiteDataAdapter(sql, connection))

{

dataAdapter.Fill(dataSet);

DataTable at = new DataTable("Students");

dataAdapter.Fill(at);

if(at.Rows.Count < 1)

{

MessageBox.Show("Ничего не найденно");

}

StudentssGrid.ItemsSource = at.DefaultView;

}

connection.Close();

}

Clear();

}

catch(Exception e)

{

MessageBox.Show(e.Message);

Clear();

}

}

}

public bool Check()

{

return StudentssGrid.SelectedItems.Count >= 1;

}

public bool SeasrchCheck(string param)

{

return param.Length != 0;

}

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

{

if (!Check())

{

MessageBox.Show("Не выбрана запись!");

}

else

{

DataRowView dataRow = (DataRowView)StudentssGrid.SelectedItem;

i = Convert.ToString(dataRow.Row.ItemArray[0]);

Edit obj = new Edit();

obj.Spec.Text = Convert.ToString(dataRow.Row.ItemArray[4]);

obj.Name.Text = Convert.ToString(dataRow.Row.ItemArray[0]);

obj.Group.Text = Convert.ToString(dataRow.Row.ItemArray[1]);

obj.Diagnosis.Text = Convert.ToString(dataRow.Row.ItemArray[6]);

obj.Course.Text = Convert.ToString(dataRow.Row.ItemArray[5]);

obj.Age.Text = Convert.ToString(dataRow.Row.ItemArray[3]);

obj.Show();

obj.n = Convert.ToString(dataRow.Row.ItemArray[0]);

Close();

}

}

public void Clear()

{

TextBoxName.Clear();

TextBoxGroup.Clear();

TextBoxCourse.Clear();

}

public void Transport(object sender, RoutedEventArgs e)

{

MessageBoxResult result = MessageBox.Show("Переместить запись в архив?", " ", MessageBoxButton.YesNoCancel, MessageBoxImage.Question);

if (result == MessageBoxResult.Yes)

{

DataRowView dataRow = (DataRowView)StudentssGrid.SelectedItem;

string sur = Convert.ToString(dataRow.Row.ItemArray[0]);

string Gender = Convert.ToString(dataRow.Row.ItemArray[1]);

string Group = Convert.ToString(dataRow.Row.ItemArray[2]);

string spec = Convert.ToString(dataRow.Row.ItemArray[4]);

string age = Convert.ToString(dataRow.Row.ItemArray[3]);

string course = Convert.ToString(dataRow.Row.ItemArray[5]);

string diagnosis = Convert.ToString(dataRow.Row.ItemArray[6]);

string notes = Convert.ToString(dataRow.Row.ItemArray[7]);

MessageBox.Show($"{sur}, {Gender}, {Group},{spec},{age},{course},{diagnosis},{notes}");

SQLiteConnection sqlite\_conn = new SQLiteConnection("Data Source=accounting.db;Version=3;New=False;Compress=True;");

sqlite\_conn.Open();

SQLiteCommand sqlite\_cmd = sqlite\_conn.CreateCommand();

DateTime obj = DateTime.Now;

DateTime dateOnly = obj.Date;

string date = dateOnly.ToShortDateString();

sqlite\_cmd.CommandText =

$@"INSERT INTO archieve

(

ФИО,

Группа,

Пол,

Возвраст,

Специальность,

Курс,

Диагноз,

Примечания,

Выписан

)

VALUES

('{sur}', '{Gender}', '{Group}',{age},'{spec}',{course},'{diagnosis}','{notes}','{date}')";

sqlite\_cmd.ExecuteNonQuery();

sqlite\_conn.Close();

Delete();

}

else if (result == MessageBoxResult.No)

{

Delete();

}

}

public void Delete()

{

DataRowView dataRow = (DataRowView)StudentssGrid.SelectedItem;

string a = Convert.ToString(dataRow.Row.ItemArray[0]);

DataRowView row = (DataRowView)StudentssGrid.SelectedItem;

row.Row.Delete();

SQLiteConnection conn = new SQLiteConnection("Data Source=accounting.db");

conn.Open();

SQLiteCommand cmd = conn.CreateCommand();

string sql\_delete = $"DELETE FROM Students WHERE ФИО = '{a}'";

cmd.CommandText = sql\_delete;

try

{

cmd.ExecuteNonQuery();

}

catch (SQLiteException ex)

{

MessageBox.Show(ex.Message);

}

conn.Close();

Window\_Loaded(null, null);

}

}

Приложение Б

(справочное)

Формы входных и выходных документов



Рисунок Б.1

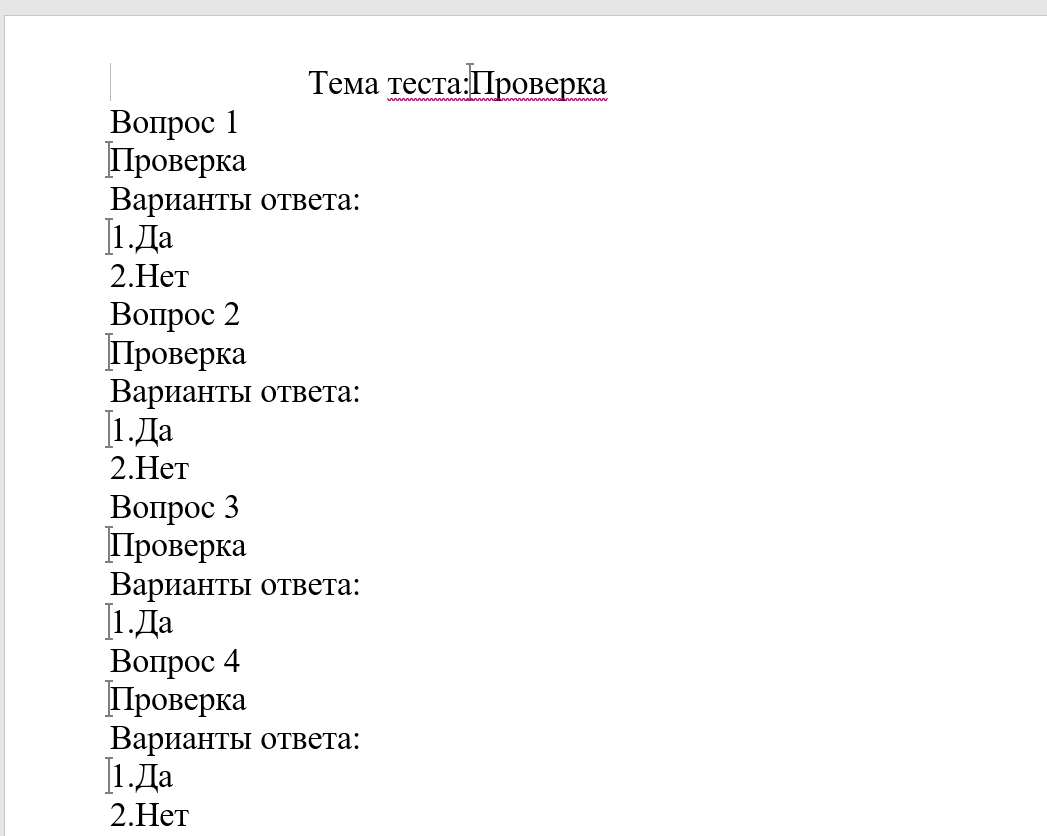


Рисунок Б.2

Приложение В

(справочное)

Результаты работы программы

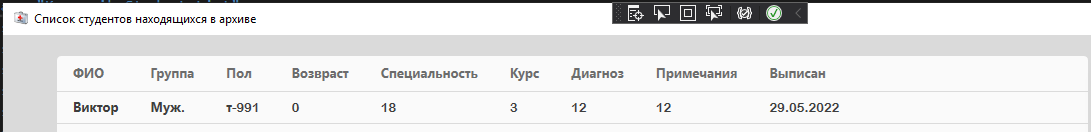


Рисунок В.1

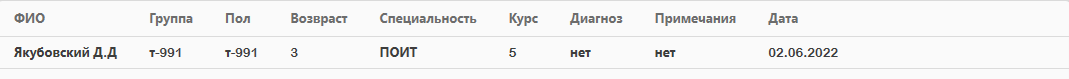


Рисунок В.2

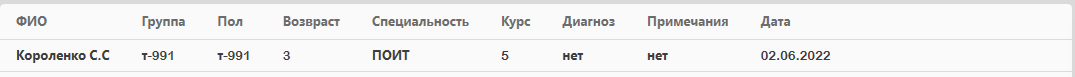


Рисунок В.3

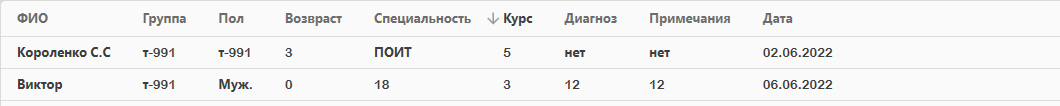


Рисунок В.4

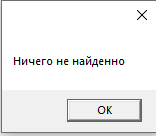


Рисунок В.5

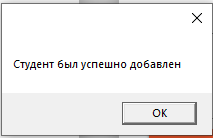


Рисунок В.6



Рисунок В.7

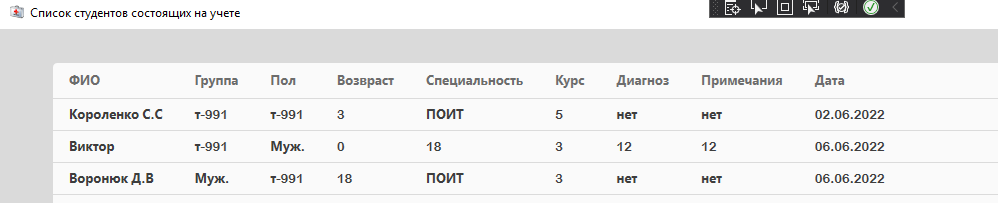


Рисунок В.8



Рисунок В.9

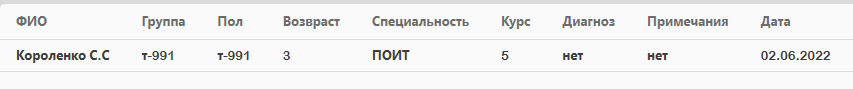


Рисунок В.10