**Оглавление**

[Аннотация 3](#_Toc516068080)

[Введение 4](#_Toc516068081)

[1 Общие сведения о программном средстве 5](#_Toc516068082)

[1.1 Основное функциональное назначение программного средства 5](#_Toc516068083)

[1.2 Полное наименование программного средства 5](#_Toc516068084)

[1.3 Условное обозначение программного средства 5](#_Toc516068085)

[1.4 Разработчики программного средства 5](#_Toc516068086)

[2 Техническое задание 6](#_Toc516068087)

[2.1 Основание для разработки 6](#_Toc516068088)

[2.2 Назначение разработки 6](#_Toc516068089)

[2.3 Требование к программному средству 6](#_Toc516068090)

[2.4 Требования к программной документации 6](#_Toc516068091)

[2.5 Требования к эргономике и технической эстетике 7](#_Toc516068092)

[2.6 Стадии и этапы разработки 7](#_Toc516068093)

[2.7 Порядок контроля и приемки 9](#_Toc516068094)

[3 Пояснительная записка к программному средству 10](#_Toc516068095)

[3.1 Декомпозиция поставленной задачи 10](#_Toc516068096)

[3.2 Общая архитектура программного средства 11](#_Toc516068097)

[3.3 Разработка алгоритма решения задачи и отдельных подзадач 13](#_Toc516068098)

[3.4 Реализация функционального назначения программного средства 14](#_Toc516068099)

[3.5 Структурная организация хранимых данных 15](#_Toc516068100)

[3.6 Разработка интерфейса программного средства 21](#_Toc516068101)

[3.7 Описание структуры выходной информации 32](#_Toc516068102)

[4 Руководство пользователя 33](#_Toc516068103)

[4.1 Назначение программного средства 33](#_Toc516068104)

[4.2 Условие выполнения программного средства 33](#_Toc516068105)

[4.3 Эксплуатация программного средства 34](#_Toc516068106)

[4.4 Сообщения пользователю 35](#_Toc516068107)

[5 Разработка тестового набора 36](#_Toc516068108)

[5.1 Определение верхней границы количества тестов 36](#_Toc516068109)

[5.2 Разработка тестовых вариантов 38](#_Toc516068110)

[5.3 Составление отчета о тестировании 46](#_Toc516068111)

[Заключение 47](#_Toc516068112)

[Оценка качества программного средства с помощью метрик 47](#_Toc516068113)

[Список использованных источников 50](#_Toc516068114)

[Приложение А Текст кода программного средства 51](#_Toc516068115)

[Приложение Б Спецификация программного средства 98](#_Toc516068117)

[Приложение B Диск с программным средством 100](#_Toc516068118)

**Аннотация**

На дипломную работу студента Медведева В.А. на тему: «Автоматизированная информационная система «Фирма по обслуживанию орг техники»». Дипломная работа представлена пояснительной запиской на 100 листах машинописного текста. Состоит из введения, 5 глав, заключения и списка литературных источников.

В работе содержится общее описание программного средства, изложены требования к программному средству, описаны его архитектура, реализация функций, интерфейс, приведено руководство пользователя, обоснованы тестовые пакеты и выполнена оценка качества программного продукта с помощью метрик.

**Введение**

В дипломной работе была поставлена задача разработать программный модуль ремонта организационной техники. Результатом работы должна быть программа, которая позволяет формировать отчет по контингенту. Программа должна иметь возможность водить и хранить данные о заявках на ремонт. Также программа должна выводить в печатной форме отчет по контингенту.

Для достижения поставленной цели был составлен основной алгоритм работы программы, а также разработаны алгоритмы вспомогательных процедур.

Для реализации алгоритмов была выбрана связка PostgreSQL + java. PostgreSQL был выбран как наиболее широко распространенная СУБД не требующая специальной установки. Для создания программы была выбрана бесплатная версия java 17 community edition.

При разработке интерфейса программы, я учитывал наиболее нужные и важные функциональные возможности, которые должна выполнять данная программа.

**1 Общие сведения о программном средстве**

**1. ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММНОМ СРЕДСТВЕ**

**1.1 Основное функциональное назначение программного средства**

Программа предназначена для хранения заявок на ремонт организационной техники. Программа не требует владения современными информационными технологиями для эффективного использования данного программного средства, т.е. подходит для любого пользователя.

**1.2 Полное наименование программного средства**

Разработка интегрированного программного модуля «Фирма по обслуживанию орг техники».

**1.3 Условное обозначение программного средства**

«Org-Tech»

**1.4 Разработчики программного средства**

Программное средство разработал студент Зеленодольского Механического колледжа группы 205 Медведев В.А.

**2 Техническое задание**

**2.1 Основание для разработки**

Работа ведётся на основании задания на дипломную работу по специальности 09.02.03 «Программирование в компьютерных системах».

**2.2 Назначение разработки**

Программа «Org-tech» работает под управлением операционных систем Windows 7 и выше. Программа предназначена для формирования отчета по контингенту.

**2.3 Требование к программному средству**

«Org-tech» устанавливается на персональный компьютер, имеющий представленные минимальные системные требования:

Оперативная память: 2048Мб;

Процессор: Intel ® Pentium ® Core i2 2000 МГц;

Разрешение экрана: не менее 1024х768

«Org-tech» написана на объектно-ориентированном языке программировании java. Программа не может подлежать сторонней оптимизации и редактированию программного кода.

**2.4 Требования к программной документации**

Состав программной документации должен включать в себя:

1) техническое задание;

2) сроки выполнения работ;

3) программу и методику испытаний;

4) эксплуатационные инструкции пользователю;

**2.5 Требования к эргономике и технической эстетике**

Система обеспечивает удобный для пользователей системный интерфейс, отвечающий следующим требованиям:

- единый стиль оформления для пользовательских интерфейсов;

- удобная, интуитивно понятная навигация в интерфейсе пользователя.

Пользовательские интерфейсы системы спроектированы и разработаны с применением единых принципов графического представления информации и организации доступа к функциональным возможностям и сервисам. Разработан графический дизайн пользовательских интерфейсов, цветовые, шрифтовые и композиционные решения для отображения текстов, изображений, таблиц, гиперссылок, управляющих и навигационных элементов (меню, кнопок, форм и т.п.).

Экранные формы спроектированы с учетом требований унификации:

- все экранные формы пользовательского интерфейса выполнены в едином графическом дизайне, с одинаковым расположением основных элементов управления и навигации;

- для обозначения сходных операций использованы сходные графические значки, кнопки и другие управляющие (навигационные) элементы. Термины, используемые для обозначения типовых операций (добавление информационной сущности, редактирование поля данных), а также последовательности действий пользователя при их выполнении унифицированы;

- внешнее поведение сходных элементов интерфейса (реакция на наведение указателя «мыши», переключение фокуса, нажатие кнопки) реализованы одинаково для однотипных элементов.

**2.6 Стадии и этапы разработки**

Разработка должна быть проведена в три стадии:

1) разработка технического задания;

2) рабочее проектирование;

3) внедрение.

На стадии разработки технического задания должен быть выполнен этап разработки, согласования и утверждения настоящего технического задания.

На стадии рабочего проектирования должны быть выполнены перечисленные ниже этапы работ:

1) разработка программы;

2) разработка программной документации;

3) испытания программы.

На этапе разработки технического задания должны быть выполнены перечисленные ниже работы:

1) постановка задачи;

2) определение и уточнение требований к техническим средствам;

3) определение требований к программе;

4) определение стадий, этапов и сроков разработки программы и документации на нее;

5) выбор языков программирования;

6) согласование и утверждение технического задания.

На этапе разработки программы должна быть выполнена работа по программированию (кодированию) и отладке программы.

На этапе разработки программной документации должна быть выполнена разработка программных документов в соответствии с пунктом Предварительный состав программной документации настоящего технического задания.

На этапе испытаний программы должны быть выполнены перечисленные ниже виды работ:

1) разработка, согласование и утверждение порядка и методики испытаний;

2) проведение испытаний;

3) корректировка программы и программной документации по результатам испытаний.

**2.7 Порядок контроля и приемки**

Сдача-приёмка работ производится поэтапно, в соответствии с рабочей программой и календарным планом. Сдача-приемка осуществляется комиссией, в состав которой входят представители ГАПОУ «ЗМК» и работодатель. По результатам приемки подписывается акт приемочной комиссии.

В указанном подразделе, согласно таблице «Сроки выполнения работ» технического задания, будет описаны перечень участвующих организаций, место и сроки проведения работ, согласно п. 2.8 ГОСТ 34.602-89.

Порядок согласования и утверждения приемочной документации должен регламентироваться организационно-распорядительной документацией организации, принимающей участие в создании программного продукта. Согласно разделу «Приемка результатов разработки» ГОСТ 15.001-88 для согласования и утверждения приемочной документации создается приемочная комиссия (приказом).

Статус приемочной комиссии определяется Заказчиком до проведения испытаний.

**3 Пояснительная записка к программному средству**

**3.1 Декомпозиция поставленной задачи**

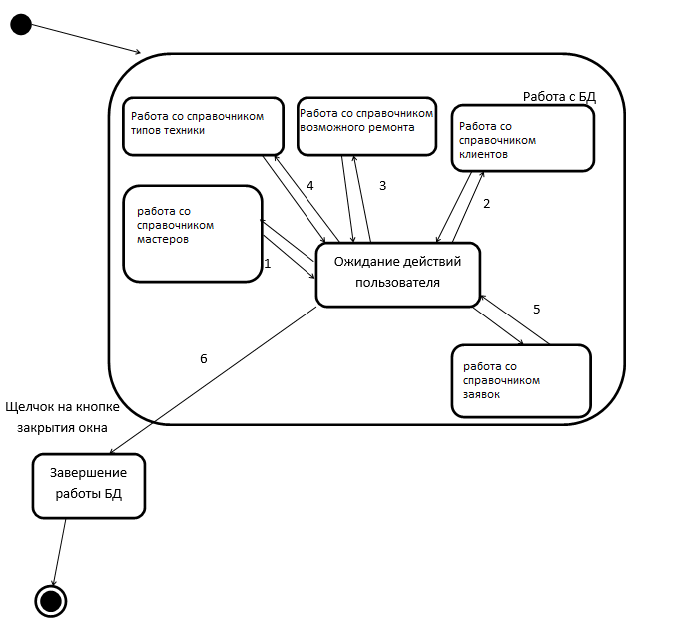
****

Рисунок 1

6

**3.2 Общая архитектура программного средства**



Рисунок 2. Диаграмма вариантов использования

В «Org-tech» используется событийно-ориентированная архитектура. Роль агента (источник событий) в программе выполняет Пользователь, за роль стоков (потребители событий) отвечают таблицы, входящие в базу данных. Например, когда пользователь выбирает определенное действие, добавить, удалить, редактировать, вывести отчет и т.д., система осуществит выбранные действия, и база отреагирует соответствующим образом: запись добавлена, удалена, отредактирована, либо был сформирован отчет. Системная архитектура пользователя рассматривает это изменение состояния как событие, создаваемое, публикуемое, определяемое и потребляемое различными приложениями в составе архитектуры.

Рисунок 3

Выбор действия

Добавить запись в БД

Удалить запись из БД

Редактировать запись в БД

Искать запись в БД

Выход

Рисунок 2

**3.3 Разработка алгоритма решения задачи и отдельных подзадач**

Программа «Org-tech» имеет определенный набор входных данных, такие как: справочник типов техники, справочник возможного ремонта, справочник Клиентов, справочник мастеров.

Выходными данными являются: сформированный отчет по контингенту.

**3.4 Реализация функционального назначения программного средства**

**Общий алгоритм**

Рисунок 4

Конец

Ввод данных в компоненты форм

Добавить запись

Ввод данных в компоненты форм

Редактировать запись

Начало

Установка

соединения с БД

Выбор действия

Удаление записи возможно?

Удалить

запись

**+**

**-**

Удалить запись

Завершить работу?

В поля формы введены корректные данные?

Добавить

запись

**+**

**-**

В поля формы введены корректные данные?

Изменить

запись

**+**

**-**

**+**

**-**

**3.5 Структурная организация хранимых данных**

Для создания БД необходимо определиться с данными, которые необходимы для полноценного функционирования системы.

Для организации более эффективной обработки данных применяется нормализация. Таблицы моей БД находятся в 3НФ:

* БД находится в форме -1НФ потому, что
  + - Таблица не имеет повторяющихся записей;
    - Каждый атрибут отношения хранит одно-единственное значение и не является списком, ни множеством значений;
    - Таблица не имеет повторяющихся групп полей.
    - Вторая нормальная форма(2НФ):
    - Устранены атрибуты, зависящие только от части уникального (первичного) идентификатора.
      * + Третья нормальная форма(3НФ):
    - Отсутствуют атрибуты, зависящие от атрибутов, не входящих в уникальный (первичный) ключ.

База данных была создана, с использованием средств Microsoft Access, на основании описания моего задания. На рисунке ниже представлена полная структура базы данных.

Изображение выглядит как текст, снимок экрана, число, диаграмма

Автоматически созданное описание

Рисунок 5

Ниже представлено подробное описание таблиц.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 6

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 7

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 8

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 9

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 10

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 11

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 12

**3.6 Разработка интерфейса программного средства**

Программное средство состоит из исполняемого файла Org-Tech.exe, файла конфигурации MyAppConfig.xml и базы данных orgtech.db.

При успешном запуске программы на экране появляется главное окно (рис. 13).

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

Рисунок 13

Окно авторизации состоит из следующих элементов:

* Два поля ввода: "Логин" и "Пароль". Пользователь должен ввести свои учетные данные в соответствующие поля для входа в систему.
* Кнопка "Войти". После ввода логина и пароля пользователь нажимает на эту кнопку для авторизации.
* Кнопка "Зарегистрироваться", расположенная в верхней части окна справа. При нажатии этой кнопки пользователь может перейти к процессу регистрации нового аккаунта.

Главное окно и другие элементы интерфейса станут доступны пользователю после успешной авторизации.

В случае отсутствия акта регистрации пользователь нажмет на кнопку Зарегистрироваться и перейдет на окно регистрации (рис. 13).

Изображение выглядит как текст, снимок экрана, программное обеспечение, дизайн

Автоматически созданное описание

Рисунок 14

* Три поля ввода для ввода имени, отчества и фамилии пользователя.
* Поле ввода для логина.
* Поле ввода для пароля.
* Выпадающий список для выбора пола пользователя.
* Две кнопки: "Зарегистрироваться" и "Назад". Кнопка "Зарегистрироваться" позволяет пользователю завершить процесс регистрации, а кнопка "Назад" возвращает к предыдущему окну или экрану.

Все элементы ввода являются обязательными для заполнения перед нажатием на кнопку "Зарегистрироваться". Системное меню и другие элементы интерфейса станут доступны пользователю после успешной регистрации и авторизации.

После успешной авторизации либо регистрации будет открыто главное окно (рис. 15)

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Автоматически созданное описание

Рисунок 15

Главное окно состоит из следующих элементов:

* В верхней части окна расположен заголовок "Ремонт орг техники" и кнопка "Личный кабинет" для перехода в личный профиль пользователя.
* В левой части окна находятся кнопки навигации:
  + **"Мои заявки"** – кнопка для перехода к списку заявок мастера.
  + **"Категории"** – кнопка для перехода к выбору категорий связанных с ремонтируемой техникой.
  + **"Отчеты"** – кнопка для создания отчетов.
  + **"Создать"** – кнопка для создания новой заявки на ремонт.
* Центральная часть окна содержит таблицу заявок с колонками:
  + Название заявки (например, "Ремонт телефона").
  + Статус заявки (например, "В работе").

Пользователь может видеть и управлять своими заявками с помощью данного интерфейса. По мере взаимодействия с программой, нажимая на соответствующие кнопки, пользователь может переходить к другим разделам программы, таким как "Категории" или "Отчеты".

При выборе пункта меню «Личный кабинет» открывается окно (рис. 16)

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 16

Страница "Информация о мастере" состоит из следующих элементов:

* Заголовок в верхней части окна "Информация о мастере".
* Формы для ввода и редактирования данных мастера, включая следующие поля:
  + **Логин** (например, " vladmed").
  + **Фамилия** (например, " Медведев").
  + **Имя** (например, " Владислав").
  + **Отчество** (например, " Александрович").
  + **Пол** (например, "Мужской").
* Три кнопки в нижней части окна:
  + **"Редактировать"** – кнопка для сохранения изменений в данных мастера.
  + **"Назад"** – кнопка для возврата на предыдущее окно (главное окно программы).
  + **"Выход"** – кнопка для выхода из учетной записи.

Пользователь может редактировать свои личные данные и сохранять изменения с помощью кнопки "Редактировать". Кнопка "Назад" позволяет вернуться на главное окно программы, а кнопка "Выход" завершает сессию пользователя.

При выборе пункта меню «Редактировать» открывается окно (рис. 17)

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 17

Страница "Редактирование мастера" состоит из следующих элементов:

* Заголовок в верхней части окна "Редактирование мастера".
* Формы для редактирования данных мастера, включая следующие поля:
  + **Логин** (например, " vladmed").
  + **Фамилия** (например, " Медведев").
  + **Имя** (например, " Владислав").
  + **Отчество** (например, " Александрович").
  + **Пол** (например, "Мужской").
* Две кнопки в нижней части окна:
  + **"Сохранить"** – кнопка для сохранения изменений и возврата на главную страницу
  + **"Отмена"** – кнопка для отмены изменений и возврата на страницу "Информация о мастере"
* При выборе пункта главной страницы меню «Категории» открывается окно (рис. 18)

Изображение выглядит как текст, снимок экрана, Прямоугольник, дизайн

Автоматически созданное описание

Рисунок 18

В левой части окна представлены три кнопки: "Мои заявки", "Категории" и "Отчеты". Эти кнопки позволяют пользователю перейти в соответствующие разделы для управления заявками на ремонт, категориями и формирования отчетов.

В центральной части окна расположены три больших кнопки: "Ремонт", "Клиенты" и "Типы". Эти кнопки обеспечивают доступ к различным функциональным областям программы, связанным с управлением ремонтными работами, информацией о клиентах и типами ремонтируемой техники.

Описание элементов окна:

* **"Мои заявки"** - нажатием этой кнопки пользователь переходит в раздел управления своими заявками на ремонт.
* **"Категории"** - нажатием этой кнопки открывается раздел управления категориями.
* **"Отчеты"** - эта кнопка позволяет перейти к разделу для формирования и просмотра отчетов.
* **"Ремонт"** - нажатием на эту кнопку пользователь переходит в раздел управления информацией о ремонте.
* **"Клиенты"** - эта кнопка открывает раздел для управления информацией о клиентах.
* **"Типы"** - нажатием на эту кнопку открывается раздел для управления типами ремонтируемой техники.

Все представленные элементы обеспечивают интуитивно понятный интерфейс для эффективного управления заявками и информацией, связанной с ремонтными работами.

* При выборе пункта «Ремонт» страницы «Категории» открывается окно (рис. 19)

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Автоматически созданное описание

Рисунок 19

В левой части окна расположены четыре кнопки: "Мои заявки", "Категории", "Отчеты" и "Создать". Эти кнопки позволяют пользователю переходить в соответствующие разделы для управления заявками, категориями, отчетами и создания новых записей.

В центральной части окна находится заголовок "Отчеты" и под ним таблица со списком отчетов.

В правой части окна расположен основной список отчетов, который может быть прокручен для просмотра всех доступных отчетов. Поле ввода текста над таблицей позволяет фильтровать или искать конкретные отчеты по ключевым словам.

На верхней панели окна указано название программы "Ремонт орг техники", которое помогает пользователю ориентироваться в приложении.

Это окно предоставляет пользователю удобный интерфейс для управления различными аспектами ремонта оргтехники, обеспечивая доступ к основным функциям через интуитивно понятные элементы управления.

* При выборе пункта «Создать» страницы открывается окно (рис. 20)

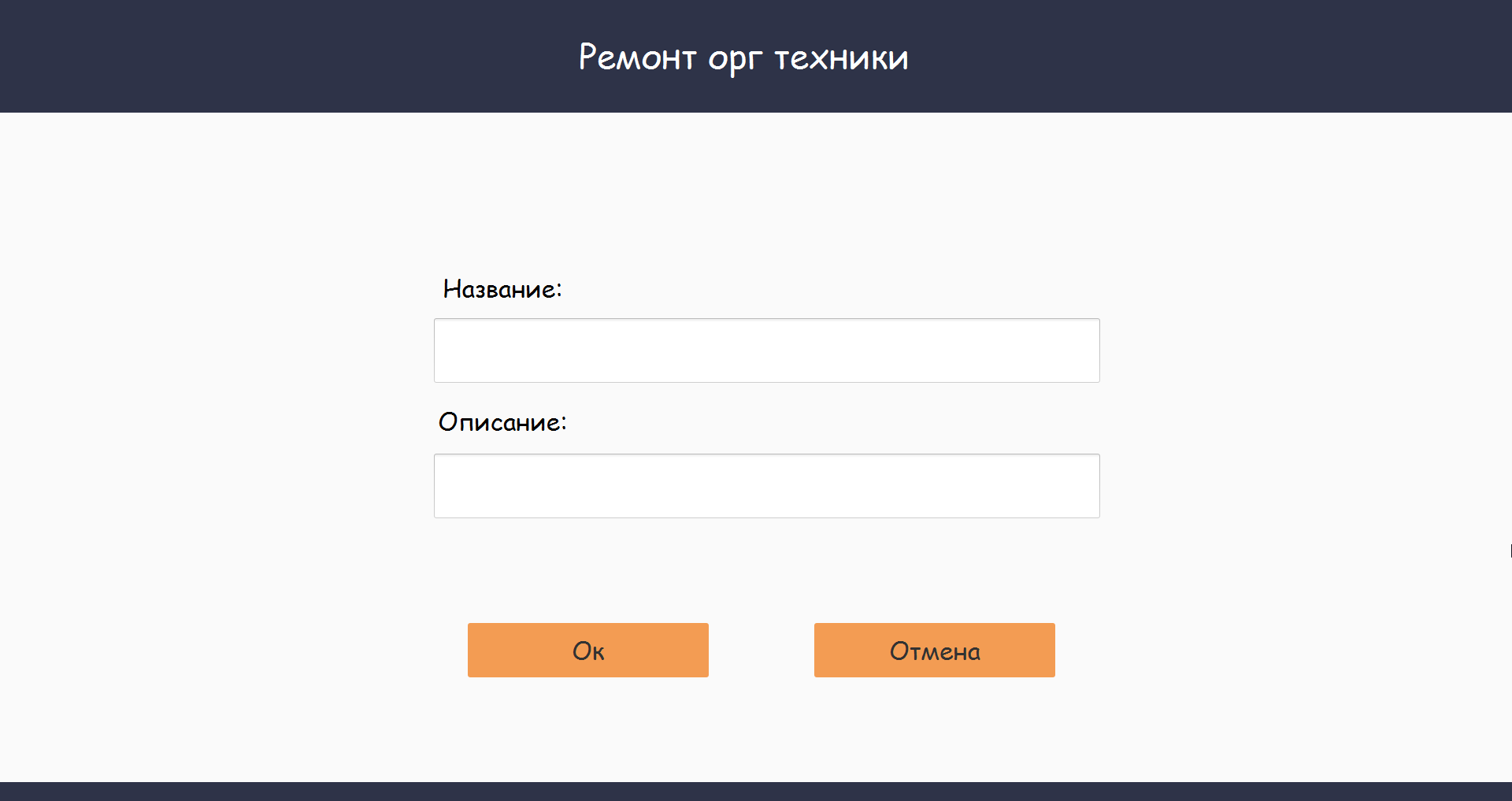


Рисунок 20

Основная часть окна состоит из следующих элементов:

* Поле для ввода названия категории с меткой "Название:", в котором указано текущее название категории - "Ремонт экрана".
* Поле для ввода описания категории с меткой "Описание:", в котором также указано текущее описание - "Ремонт экрана".

В нижней части окна расположены три кнопки:

* **"Ок"** - кнопка, предназначенная для сохранения изменений, внесенных в поля "Название" и "Описание".
* **"Отмена"** - кнопка для отмены внесенных изменений и возврата к предыдущему окну без сохранения изменений.

Эти элементы позволяют пользователю легко редактировать или удалять существующие записи, а также возвращаться к предыдущему окну, если изменения не требуются.

* При выборе ремонта на листе ремонтов открывается окно (рис. 21)

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 21

Основная часть окна состоит из следующих элементов:

* Поле для ввода названия категории с меткой "Название:", в котором указано текущее название категории - "Ремонт экрана".
* Поле для ввода описания категории с меткой "Описание:", в котором также указано текущее описание - "Ремонт экрана".

В нижней части окна расположены три кнопки:

* **"Редактировать"** - кнопка, предназначенная для сохранения изменений, внесенных в поля "Название" и "Описание".
* **"Удалить"** - кнопка для удаления выбранной категории.
* **"Отмена"** - кнопка для отмены внесенных изменений и возврата к предыдущему окну без сохранения изменений.

Эти элементы позволяют пользователю легко редактировать или удалять существующие записи, а также возвращаться к предыдущему окну, если изменения не требуются.

* При выборе пункта «Редактировать» открывается окно (рис. 22)

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рисунок 22

Основная часть окна состоит из следующих элементов:

* Поле для ввода названия с меткой "Название:". Это поле позволяет пользователю ввести название новой записи.
* Поле для ввода описания с меткой "Описание:". Это поле предназначено для ввода описания новой записи.

В нижней части окна расположены две кнопки:

* **"Ок"** - кнопка, предназначенная для сохранения введенной информации и добавления новой записи.
* **"Отмена"** - кнопка для отмены введенных изменений и закрытия окна без сохранения информации.

Эти элементы позволяют пользователю легко вводить и сохранять новые записи, а также отменять действия при необходимости.

* При выборе пункта «Клиенты» страницы «Категории» открывается окно (рис. 23)

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Автоматически созданное описание

Рисунок 23

Основная часть окна состоит из следующих элементов:

* Заголовок "Клиенты".
* Поле для ввода и поиска клиента с меткой "Клиенты:".
* Список клиентов, отображаемый в табличном виде

В нижней части окна расположена кнопка "Создать", которая позволяет добавить нового клиента в систему.

Описание функциональности:

* **"Мои заявки"** - раздел для просмотра и управления заявками пользователя на ремонт.
* **"Категории"** - раздел для добавления, редактирования и удаления категорий ремонтируемой техники.
* **"Отчеты"** - раздел, в котором пользователь может просматривать, создавать и управлять отчетами о ремонтах.
* **"Создать"** - кнопка для создания новой записи в текущем разделе (например, нового клиента или заявки).
* При выборе клиента на листе клиентов открывается окно (рис. 24)

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание

Рисунок 24

Основная часть окна состоит из следующих элементов:

* Поле для ввода имени клиента с меткой "Имя:", в котором указано текущее имя клиента - "Виктор Корнеплод".
* Поле для ввода номера телефона клиента с меткой "Номер:", в котором указан текущий номер - "88005553535".

В нижней части окна расположены три кнопки:

* **"Редактировать"** - кнопка, предназначенная для сохранения изменений, внесенных в поля "Имя" и "Номер".
* **"Удалить"** - кнопка для удаления информации о клиенте.
* **"Отмена"** - кнопка для отмены внесенных изменений и возврата к предыдущему окну без сохранения изменений.

Эти элементы позволяют пользователю легко редактировать или удалять существующие записи о клиентах, а также отменять действия при необходимости.

* При выборе пункта «Редактировать» открывается окно (рис. 25)

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание Рисунок 25

Основная часть окна состоит из следующих элементов:

* Поле для ввода имени клиента с меткой "Имя:", в котором указано текущее имя клиента - "Виктор Корнеплод".
* Поле для ввода номера телефона клиента с меткой "Номер:", в котором указан текущий номер - "88005553535".

В нижней части окна расположены две кнопки:

* **"Ок"** - кнопка, предназначенная для сохранения изменений, внесенных в поля "Имя" и "Номер".
* **"Отмена"** - кнопка для отмены внесенных изменений и возврата к предыдущему окну без сохранения изменений.

Эти элементы позволяют пользователю легко редактировать или удалять существующие записи о клиентах, а также отменять действия при необходимости.

* При выборе пункта «Создать» страницы открывается окно (рис. 26)

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рисунок 26

Основная часть окна состоит из следующих элементов:

* Поле для ввода имени клиента с меткой "Имя:". Это поле предназначено для ввода имени нового клиента.
* Поле для ввода номера телефона клиента с меткой "Номер:". Это поле предназначено для ввода номера телефона нового клиента.

В нижней части окна расположены две кнопки:

* **"Ок"** - кнопка, предназначенная для сохранения введенной информации и добавления нового клиента.
* **"Отмена"** - кнопка для отмены введенных данных и закрытия окна без сохранения информации.

Эти элементы позволяют пользователю легко вводить и сохранять информацию о новых клиентах, а также отменять действия при необходимости.

* При выборе пункта «Ремонт» страницы «Категории» открывается окно (рис. 27)

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Автоматически созданное описание

Рисунок 27

Основная часть окна состоит из следующих элементов:

* Заголовок "Типы".
* Поле для ввода и поиска типов с меткой "Типы:".
* Список типов.
* В нижней части окна расположена кнопка "Создать", которая позволяет добавить новый тип в систему.

Элементы интерфейса:

* **"Мои заявки"** - кнопка для перехода в раздел управления заявками на ремонт.
* **"Категории"** - кнопка для перехода в раздел управления категориями ремонтируемой техники.
* **"Отчеты"** - кнопка для перехода в раздел формирования и просмотра отчетов.
* **"Создать"** - кнопка для создания новой записи в текущем разделе (например, нового типа или заявки).

Это окно предоставляет пользователю удобный интерфейс для управления различными типами ремонтируемой техники, обеспечивая доступ к основным функциям через интуитивно понятные элементы управления.

* При выборе клиента на листе клиентов открывается окно (рис. 28)

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рисунок 28

Основная часть окна состоит из следующих элементов:

* Поле для ввода названия типа с меткой "Название:", в котором указано текущее название типа - "Тип 1".
* Поле для ввода описания типа с меткой "Описание:". Это поле предназначено для ввода описания текущего типа (в настоящее время пустое).

В нижней части окна расположены три кнопки:

* **"Редактировать"** - кнопка, предназначенная для сохранения изменений, внесенных в поля "Название" и "Описание".
* **"Удалить"** - кнопка для удаления выбранного типа.
* **"Отмена"** - кнопка для отмены внесенных изменений и возврата к предыдущему окну без сохранения изменений.

Эти элементы позволяют пользователю легко редактировать или удалять существующие типы, а также отменять действия при необходимости.

* При выборе пункта «Редактировать» страницы открывается окно (рис. 29)

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание

Рисунок 29

Основная часть окна состоит из следующих элементов:

* Поле для ввода названия типа с меткой "Название:", в котором указано текущее название типа - "Тип 1".
* Поле для ввода описания типа с меткой "Описание:". Это поле предназначено для ввода описания текущего типа.

В нижней части окна расположены две кнопки:

* **"Ок"** - кнопка, предназначенная для сохранения изменений, внесенных в поля "Название" и "Описание".
* **"Отмена"** - кнопка для отмены внесенных изменений и возврата к предыдущему окну без сохранения изменений.

Эти элементы позволяют пользователю легко редактировать существующие типы, а также отменять действия при необходимости.

* При выборе пункта «Создать» страницы открывается окно (рис. 30)

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 30

Основная часть окна состоит из следующих элементов:

* Поле для ввода названия типа с меткой "Название:". Это поле предназначено для ввода названия нового типа.
* Поле для ввода описания типа с меткой "Описание:". Это поле предназначено для ввода описания нового типа.

В нижней части окна расположены две кнопки:

* **"Ок"** - кнопка, предназначенная для сохранения введенной информации и добавления нового типа.
* **"Отмена"** - кнопка для отмены введенных данных и закрытия окна без сохранения информации.

Эти элементы позволяют пользователю легко вводить и сохранять информацию о новых типах ремонтируемой техники, а также отменять действия при необходимости.

* При выборе заявки на листе страницы «Заявки» клиентов открывается окно (рис. 31)

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 31

Основная часть окна состоит из следующих элементов:

* Поля для ввода информации о заявке с метками:
  + **"Имя:"** - поле для ввода названия заявки, в данном случае "Ремонт телефона".
  + **"Цена:"** - поле для ввода стоимости ремонта, в данном случае "1000.00".
  + **"Клиент:"** - поле для ввода имени клиента, в данном случае "Виктор Корнеплод".
  + **"Ремонт:"** - поле для ввода типа ремонта, в данном случае "Ремонт 1".
  + **"Тип:"** - поле для ввода типа техники, в данном случае "Тип 1".
  + **"Статус:"** - поле для ввода текущего статуса заявки, в данном случае "В работе".
  + **"Дата:"** - поле для ввода даты заявки, в данном случае "4.6.2024".

В нижней части окна расположены три кнопки:

* **"Редактировать"** - кнопка, предназначенная для сохранения изменений, внесенных в поля.
* **"Удалить"** - кнопка для удаления выбранной заявки.
* **"Назад"** - кнопка для отмены внесенных изменений и возврата к предыдущему окну без сохранения изменений.

Эти элементы позволяют пользователю легко просматривать, редактировать или удалять существующие заявки, а также возвращаться к предыдущему окну при необходимости.

* При выборе пункта «Редактировать» страницы открывается окно (рис. 32)

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 32

Основная часть окна состоит из следующих элементов:

* Поля для ввода информации о заявке с метками:
  + **"Имя:"** - поле для ввода названия заявки, в данном случае "Ремонт телефона".
  + **"Цена:"** - поле для ввода стоимости ремонта, в данном случае "1000.00".
  + **"Клиент:"** - выпадающее поле для выбора клиента, в данном случае "Виктор Корнелплод", и кнопка "Добавить" для добавления нового клиента.
  + **"Ремонт:"** - выпадающее поле для выбора типа ремонта, в данном случае "Ремонт 1", и кнопка "Добавить" для добавления нового типа ремонта.
  + **"Тип:"** - выпадающее поле для выбора типа техники, в данном случае "Тип 1", и кнопка "Добавить" для добавления нового типа техники.
  + **"Статус:"** - выпадающее поле для выбора текущего статуса заявки, в данном случае "В работе".

В нижней части окна расположены две кнопки:

* **"Сохранить"** - кнопка, предназначенная для сохранения введенной информации и добавления новой заявки.
* **"Назад"** - кнопка для отмены внесенных данных и возврата к предыдущему окну без сохранения информации.

Эти элементы позволяют пользователю легко вводить и сохранять информацию о новых заявках, а также возвращаться к предыдущему окну при необходимости.

* При выборе пункта «Создать» страницы открывается окно (рис. 33)

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 33

Основная часть окна состоит из следующих элементов:

* Поля для ввода информации о заявке с метками:
  + **"Имя:"** - поле для ввода названия заявки.
  + **"Цена:"** - поле для ввода стоимости ремонта.
  + **"Клиент:"** - выпадающее поле для выбора клиента с кнопкой "Добавить" для добавления нового клиента.
  + **"Ремонт:"** - выпадающее поле для выбора типа ремонта с кнопкой "Добавить" для добавления нового типа ремонта.
  + **"Тип:"** - выпадающее поле для выбора типа техники с кнопкой "Добавить" для добавления нового типа техники.
  + **"Статус:"** - выпадающее поле для выбора текущего статуса заявки.

В нижней части окна расположены две кнопки:

* **"Сохранить"** - кнопка, предназначенная для сохранения введенной информации и добавления новой заявки.
* **"Назад"** - кнопка для отмены внесенных данных и возврата к предыдущему окну без сохранения информации.

Эти элементы позволяют пользователю легко вводить и сохранять информацию о новых заявках, а также возвращаться к предыдущему окну при необходимости.

Выходными данными являются: сформированный отчет по заявкам.

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Автоматически созданное описание

Рисунок 34

**3.7 Описание структуры выходной информации**

**4 Руководство пользователя**

**4.1 Назначение программного средства**

Программа предназначена для формирования отчета по контингенту.

**4.2 Условие выполнения программного средства**

Условия, при соблюдении которых возможно выполнение работы с программным продуктом:

Компьютер с установленной на нём «Org-Tech» и PostgreSQL.

Подготовительные действия:

На иконке «Org-Tech» рабочего стола произвести двойной щелчок левой кнопкой мыши.

**4.3 Эксплуатация программного средства**

Этот раздел содержит обзор приложений «Org-Tech» и ее основных параметров.

*Установка*

Прежде чем приступить к процессу установки, нужно убедиться, что Вы начали сеанс работы Windows с правами администратора.

Создайте в желаемом месте папку, в которую будет установлена «Org-Tech». Скопируйте в эту папку, с носителя с дистрибутивом «Org-Tech», все содержимое. Если вы хотите хранить файл базы данных в другом месте, то перенесите его туда.

*Вводные сведения об интерфейсе пользователя*

В окнах «Org-Tech» используются разнообразные элементы интерфейса, такие как:

* однострочные текстовые редакторы;
* текстовые таблицы;
* меню;
* управляющие кнопки;
* кнопки работы с таблицами базы данных.

Однострочные текстовые редакторы предназначены для ввода произвольного текста длиной не более 255 символов.

Текстовые таблицы используются для отображения табличных данных.

Управляющие кнопки предназначены для выполнения указанных на них действий по нажатию кнопки.

Кнопки работы с таблицами базы данных предназначены для выполнения определенных операций с таблицей (рис. 35 слева на право):

* Добавить запись в таблицу;
* Удалить запись из таблицы;
* Редактировать запись.

Изображение выглядит как Шрифт, текст, логотип, Графика

Автоматически созданное описание Изображение выглядит как текст, Шрифт, логотип, Графика

Автоматически созданное описание Изображение выглядит как текст, Шрифт, Графика, логотип

Автоматически созданное описание

Рисунок 35

*Работа с «*Org-Tech*»*

Назначение основных кнопок управления базами данных приведены в таблице 1.

Таблица 1

|  |  |
| --- | --- |
| **Кнопка** | **Назначение** |
| Изображение выглядит как Шрифт, текст, логотип, Графика  Автоматически созданное описание | Предназначена для добавления новой записи в таблицу. |
| Изображение выглядит как текст, Шрифт, логотип, Графика  Автоматически созданное описание | Предназначена для удаления записи из таблицы. |
| Изображение выглядит как текст, Шрифт, Графика, логотип  Автоматически созданное описание | Предназначена для подтверждения изменений, сделанных в таблице. |

**4.4 Сообщения пользователю**

Сообщения пользователю представлены на рис. 36, 37.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Автоматически созданное описание

Рисунок 36

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Автоматически созданное описание

Рисунок 37

**5 Разработка тестового набора**

**5.1 Определение верхней границы количества тестов**

Для определения верхней границы количества тестов был разработан укрупненный алгоритм работы программы.

начало

Файл конфигурации существует?

Структура файла правильная?

**+**

**-**

Файл базы данных существует?

**++**

**--**

Запуск программы

Выход из программы?

**+**

**-**

конец

Рисунок 38

Для разработанного укрупненного алгоритма работы программы был построен граф (см. рис. 37).

Рисунок 39

Необходимое количество тестов определяется по формуле V(G) = E-N+2, где E – это число ребер, а N–это число вершин данного графа. Согласно представленному графу E=9, N=7. Далее по формуле рассчитаем необходимое количество тестовых сценариев (тест-кейсов).

V(G)=9-7+2=4.

Для данного программного средства необходимо разработать 4 тестовых сценария.

Так как каждый сценарий имеет два варианта завершения, то необходимо 8 тестовых прогонов.

**5.2 Разработка тестовых вариантов**

Тест №1

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Наличие файла конфигурации | | |
| **Функция:** | Условие | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Файл конфигурации существует в месте запуска программы | |  |  |
| **Шаги теста:** | |  |  |
| Программа запущена | | Сообщение об ошибке не выведено. Программа перешла к проверке следующего условия | пройден |
| **Постусловие:** | |  |  |
|  | |  |  |

Тест №2

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Наличие файла конфигурации | | |
| **Функция:** | Условие | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Файл конфигурации отсутствует в месте запуска программы | |  |  |
| **Шаги теста:** | |  |  |
| Программа запущена | | Выведено сообщение об ошибке. Программа закрыта | пройден |

Тест №3

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Корректность структуры файла конфигурации | | |
| **Функция:** | Условие | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Файл конфигурации существует в месте запуска программы. Структура файла корректна. | |  |  |
| **Шаги теста:** | |  |  |
| Программа запущена | | Сообщение об ошибке не выведено. Программа перешла к проверке следующего условия | пройден |

Тест №4

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Корректность структуры файла конфигурации | | |
| **Функция:** | Условие | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Файл конфигурации существует в месте запуска программы. Структура файла не корректна. | |  |  |
| **Шаги теста:** | |  |  |
| Программа запущена | | Выведено сообщение об ошибке. Программа закрыта. | пройден |

Тест №5

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Наличие файла базы данных | | |
| **Функция:** | Условие | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Файл конфигурации существует в месте запуска программы. Структура файла корректна. Файл базы данных существует по пути указанному в файле конфигурации. | |  |  |
| **Шаги теста:** | |  |  |
| Программа запущена | | Сообщение об ошибке не выведено. Программа перешла к проверке следующего условия | пройден |

Тест №6

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Наличие файла базы данных | | |
| **Функция:** | Условие | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Файл конфигурации существует в месте запуска программы. Структура файла корректна. Файл базы данных не существует по пути указанному в файле конфигурации. | |  |  |
| **Шаги теста:** | |  |  |
| Программа запущена | | Выведено сообщение об ошибке. Программа закрыта. | пройден |

Тест №7

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование завершения программы | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Программа запущена. | |  |  |
| **Шаги теста:** | |  |  |
| Была нажата кнопка выхода из программы. | | Программа закрыта. Все данные успешно сохранены. | пройден |

Тест №8

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование завершения программы | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Программа запущена. | |  |  |
| **Шаги теста:** | |  |  |
| Кнопка выхода из программы не нажата. | | Программа функционирует в штатном режиме | пройден |

**5.3 Составление отчета о тестировании**

Для форм программы были созданы и проведены тесты, результаты сведены в таблицу2:

Таблица 2

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Прохождение | |
|  | Число ТК | пройдено | не пройдено |
| Наличие файла конфигурации | 2 | 2 | 0 |
| Корректность структуры файла конфигурации | 2 | 2 | 0 |
| Тестирование завершения программы | 2 | 2 | 0 |
| ИТОГО | 8 | 8 | 0 |

По результатам тестирования проблем выявлено не было.

**Заключение**

**Оценка качества программного средства с помощью метрик**

Оценка качества программного средства проводится исходя из стандарта оценки качества ISО 9126.

На компьютере следующей конфигурации:

Установленная операционная система –Microsoft Windows7 32 бит.

Процессор: Intel Pentium G2030 3.0 GHz.

ОЗУ: 2 Gb

Таблица 3 - Требования к количественным характеристикам качества программного средства

|  |  |  |  |
| --- | --- | --- | --- |
| **Характеристики качества** | **Мера** | **Требуемое значение** | **Реальное значение** |
| **Надежность** |  | | |
| *Завершенность:*  *-* наработка на отказ при отсутствии рестарта; | Часы | 10 | 10 |
| *Устойчивость:*  - относительные ресурсы на обеспечение надежности и рестарта. | % | 0,5 | 0,5 |
| *Восстанавливаемость:*  -длительность восстановления. | Минуты | 2 | 2 |
| *Доступность-готовность:*  -относительное время работоспособного функционирования. | Вероятность | 0,998 | 0,998 |

окончание таблицы 3

|  |  |  |  |
| --- | --- | --- | --- |
| **Эффективность** |  | | |
| *Временная эффективность:*  *-*время отклика - получение результатов на типовое задание;  - пропускная способность число типовых заданий, исполняемых в единицу времени. | Секунды  Число в минуту | 3  60 | 3  60 |
| *Используемость ресурсов:*  *-* относительная величина использования ресурсов ЭВМ при нормальном функционировании программного средства. |  | 0,001 | 0,001 |

Таблица 4- Требования к качественным характеристикам программного средства.

|  |  |  |  |
| --- | --- | --- | --- |
| **Характеристики качества** | **Мера** | **Требуемое значение** | **Реальное значение** |
| **Практичность** |  | | |
| *Простота использования:*  **-**среднее время ввода заданий;  -среднее время отклика на задание. | Секунды  Секунды | 10  3 | 10  3 |
| *Изучаемость :*  - трудоемкость изучения применения ПС;  -продолжительность изучения;  -объем эксплуатационной документации; | Чел.-часы  Часы  Страницы | 5  2  15 | 5  2  15 |

Окончание таблицы 4

|  |  |  |  |
| --- | --- | --- | --- |
| **Сопровождаемость** |  | | |
| *Изменяемость:*  -трудоемкость подготовки изменений;  -длительность подготовки изменений. | Чел.-часы  Часы | 5  5 | 5  5 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Тестируемость:*  -трудоемкость тестирования изменений;  - длительность тестирования изменений. | Чел.-часы  Часы | 3  3 | 3  3 |
| **Мобильность** |  | | |
| *Адаптируемость:*  - трудоемкость адаптации;  -длительность адаптации. | Чел.-часы  Часы | 0,2  1 | 0,2  1 |
| *Простота установки:*  -трудоемкость инсталляции;  -длительность инсталляции. | Чел.-часы  Часы | ,2  1 | 0,2  1 |
| *Замещаемость:*  -трудоемкость замены компонентов;  - длительность замены компонентов. | Чел.-часы  Часы | 0,2  0,2 | 0,2  0,2 |

**Список использованных источников**

1. Базы данных. В 2-х кн. Кн. 2. Распределенные и удаленные базы данных: Учебник / В.П. Агальцов. - М.: ИД ФОРУМ: НИЦ Инфра-М, 2020. - 272 с.
2. Осипов Д. Л. Базы данных и Delphi. Теория и практика. — СПб.: БХВ-Петербург, 2018. — 746 с.
3. Базы данных: Учебное пособие / О.Л. Голицына, Н.В. Максимов, И.И. Попов. - 2-e изд., испр. и доп. - М.: Форум: ИНФРА-М, 2019. - 400 с
4. Кириллов, В. В. Введение в реляционные базы данных / В. В. Кириллов, Г. Ю. Громов. — СПб.: БХВ-Петербург, 2019. — 450 с.
5. Рудикова Л.В. Базы данных. Разработка приложений. - СПб.: БХВ-Петербург, 2018. 491 с
6. Пирогов, В. Ю. Информационные системы и базы данных: организация и проектирование: учеб. пособие / В. Ю. Пирогов. — СПб.: БХВ-Петербург, 2019. — 528 с
7. ГОСТ 19.201-78.ЕСПД. Техническое задание. Требования к содержанию и оформлению
8. ГОСТ 19.102-77.ЕСПД. Стадии разработки
9. ГОСТ 19.505-79.ЕСПД. Руководство оператора. Требования к содержанию и оформлению
10. ГОСТ 19.002-80.ЕСПД. Схемы алгоритмов и программ. Правила выполнения
11. ГОСТ 19.404-79.ЕСПД. Пояснительная записка. Требования к содержанию и оформлению.
12. Электронный ресурс http://www.cyberforum.ru
13. Электронный ресурс https://delphisources.ru/
14. Электронный ресурс https://coderbook.ru/learn-delphi/
15. Электронный ресурс https://sourceforge.net/

**Приложение А**

Разработка программного модуля ремонт организационной техники

Текст кода программного средства

460.ЗМК.00003-01 12 01

**Текст кода программного средства**

unit uadres;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, DBCtrls, Vcl.ExtCtrls;

type

TfAdres = class(TForm)

Button1: TButton;

Panel1: TPanel;

Label1: TLabel;

Label2: TLabel;

Label3: TLabel;

Label4: TLabel;

Label5: TLabel;

edCountry: TEdit;

btnCountry: TButton;

edRegion: TEdit;

btnRegion: TButton;

edDistrict: TEdit;

btnDistrict: TButton;

edlocalityType: TEdit;

btnLocalityType: TButton;

edLocality: TEdit;

btnLocality: TButton;

btnCountryClear: TButton;

btnRegionClear: TButton;

btnDistrictClear: TButton;

btnLocalityTypeClear: TButton;

btnLocalityClear: TButton;

Label11: TLabel;

Label12: TLabel;

Panel2: TPanel;

Label6: TLabel;

Label7: TLabel;

Label8: TLabel;

Label9: TLabel;

Label10: TLabel;

edStreet: TEdit;

btnStreet: TButton;

edHouse: TEdit;

edHousePart: TEdit;

edBuilding: TEdit;

edApartment: TEdit;

btnStreetClear: TButton;

Label13: TLabel;

Label14: TLabel;

procedure btnCountryClick(Sender: TObject);

procedure btnRegionClick(Sender: TObject);

procedure btnDistrictClick(Sender: TObject);

procedure btnLocalityTypeClick(Sender: TObject);

procedure btnLocalityClick(Sender: TObject);

procedure btnStreetClick(Sender: TObject);

procedure btnCountryClearClick(Sender: TObject);

procedure btnRegionClearClick(Sender: TObject);

procedure btnDistrictClearClick(Sender: TObject);

procedure btnLocalityTypeClearClick(Sender: TObject);

procedure btnLocalityClearClick(Sender: TObject);

procedure btnStreetClearClick(Sender: TObject);

procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);

procedure Button1Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

idCountry:integer;

idRegion:integer;

idDistrict:integer;

idLocalityType:integer;

idLocality:integer;

idStreet:integer;

school:boolean;

admin:boolean;

pass:string;

end;

var

fAdres: TfAdres;

implementation

{$R \*.dfm}

uses uAdresElement, udm;//, uzayavlenie;

procedure TfAdres.btnCountryClearClick(Sender: TObject);

begin

idCountry:=-1;

edCountry.Text:='';

end;

procedure TfAdres.btnCountryClick(Sender: TObject);

begin

dm.ADOadrElement.Active:=false;

dm.ADOadrElement.TableName:='adrCountry';

dm.ADOadrElement.IndexFieldNames:='nazv';

dm.ADOadrElement.Active:=true;

fAdrElement.DBGrid1.Columns[1].Title.Caption:='Страна';

fAdrElement.DBGrid1.Columns[1].Width:=fAdrElement.DBGrid1.ClientWidth;

fAdrElement.admin:=fAdres.admin;

fAdrElement.pass:=fAdres.pass;

fAdrElement.ShowModal;

if (fAdrElement.id<>-1)or(edCountry.Text='') then

begin

edCountry.Text:=fAdrElement.nazv;

idCountry:=fAdrElement.id;

end;

end;

procedure TfAdres.btnDistrictClearClick(Sender: TObject);

begin

idDistrict:=-1;

edDistrict.Text:='';

end;

procedure TfAdres.btnDistrictClick(Sender: TObject);

begin

dm.ADOadrElement.Active:=false;

dm.ADOadrElement.TableName:='adrDistrict';

dm.ADOadrElement.IndexFieldNames:='nazv';

dm.ADOadrElement.Active:=true;

fAdrElement.DBGrid1.Columns[1].Title.Caption:='Район';

fAdrElement.DBGrid1.Columns[1].Width:=fAdrElement.DBGrid1.ClientWidth;

fAdrElement.admin:=fAdres.admin;

fAdrElement.pass:=fAdres.pass;

fAdrElement.ShowModal;

if (fAdrElement.id<>-1)or(edCountry.Text='') then

begin

edDistrict.Text:=fAdrElement.nazv;

idDistrict:=fAdrElement.id;

end;

end;

procedure TfAdres.btnLocalityClearClick(Sender: TObject);

begin

idLocality:=-1;

edLocality.Text:='';

end;

procedure TfAdres.btnLocalityClick(Sender: TObject);

begin

dm.ADOadrElement.Active:=false;

dm.ADOadrElement.TableName:='adrLocality';

dm.ADOadrElement.IndexFieldNames:='nazv';

dm.ADOadrElement.Active:=true;

fAdrElement.DBGrid1.Columns[1].Title.Caption:='Населенный пункт';

fAdrElement.DBGrid1.Columns[1].Width:=fAdrElement.DBGrid1.ClientWidth;

fAdrElement.admin:=fAdres.admin;

fAdrElement.pass:=fAdres.pass;

fAdrElement.ShowModal;

if (fAdrElement.id<>-1)or(edCountry.Text='') then

begin

edLocality.Text:=fAdrElement.nazv;

idLocality:=fAdrElement.id;

end;

end;

procedure TfAdres.btnLocalityTypeClearClick(Sender: TObject);

begin

idLocalityType:=-1;

edLocalityType.Text:='';

end;

procedure TfAdres.btnLocalityTypeClick(Sender: TObject);

begin

dm.ADOadrElement.Active:=false;

dm.ADOadrElement.TableName:='adrLocalityType';

dm.ADOadrElement.IndexFieldNames:='nazv';

dm.ADOadrElement.Active:=true;

fAdrElement.DBGrid1.Columns[1].Title.Caption:='Тип населенного пункта';

fAdrElement.DBGrid1.Columns[1].Width:=fAdrElement.DBGrid1.ClientWidth;

fAdrElement.admin:=fAdres.admin;

fAdrElement.pass:=fAdres.pass;

fAdrElement.ShowModal;

if (fAdrElement.id<>-1)or(edCountry.Text='') then

begin

edLocalityType.Text:=fAdrElement.nazv;

idLocalityType:=fAdrElement.id;

end;

end;

procedure TfAdres.btnRegionClearClick(Sender: TObject);

begin

idRegion:=-1;

edRegion.Text:='';

end;

procedure TfAdres.btnRegionClick(Sender: TObject);

begin

dm.ADOadrElement.Active:=false;

dm.ADOadrElement.TableName:='adrRegion';

dm.ADOadrElement.IndexFieldNames:='nazv';

dm.ADOadrElement.Active:=true;

fAdrElement.DBGrid1.Columns[1].Title.Caption:='Регион';

fAdrElement.DBGrid1.Columns[1].Width:=fAdrElement.DBGrid1.ClientWidth;

fAdrElement.admin:=fAdres.admin;

fAdrElement.pass:=fAdres.pass;

fAdrElement.ShowModal;

if (fAdrElement.id<>-1)or(edCountry.Text='') then

begin

edRegion.Text:=fAdrElement.nazv;

idRegion:=fAdrElement.id;

end;

end;

procedure TfAdres.btnStreetClearClick(Sender: TObject);

begin

idStreet:=-1;

edStreet.Text:='';

end;

procedure TfAdres.btnStreetClick(Sender: TObject);

begin

dm.ADOadrElement.Active:=false;

dm.ADOadrElement.TableName:='adrStreet';

dm.ADOadrElement.IndexFieldNames:='nazv';

dm.ADOadrElement.Active:=true;

fAdrElement.DBGrid1.Columns[1].Title.Caption:='Улица';

fAdrElement.DBGrid1.Columns[1].Width:=fAdrElement.DBGrid1.ClientWidth;

fAdrElement.admin:=fAdres.admin;

fAdrElement.pass:=fAdres.pass;

fAdrElement.ShowModal;

if (fAdrElement.id<>-1)or(edCountry.Text='') then

begin

edStreet.Text:=fAdrElement.nazv;

idStreet:=fAdrElement.id;

end;

end;

procedure TfAdres.Button1Click(Sender: TObject);

begin

Close;

end;

procedure TfAdres.FormCloseQuery(Sender: TObject; var CanClose: Boolean);

begin

CanClose:=false;

if not school then

begin

if (idCountry<>-1)and(idLocality<>-1)and(idStreet<>-1)and(edHouse.Text<>'') then CanClose:=true

else ShowMessage('Заполните обязательные поля');

end

else

begin

if (idCountry<>-1)and(idLocality<>-1) then CanClose:=true

else ShowMessage('Заполните обязательные поля');

end;

end;

end.

unit uAdresElement;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Data.DB, Vcl.StdCtrls, Vcl.DBCtrls,

Vcl.Grids, Vcl.DBGrids, Vcl.ExtCtrls;

type

TfAdrElement = class(TForm)

Panel2: TPanel;

DBGrid1: TDBGrid;

btnSelect: TButton;

GroupBox1: TGroupBox;

Edit1: TEdit;

DBNavigator1: TDBNavigator;

procedure FormResize(Sender: TObject);

procedure btnSelectClick(Sender: TObject);

procedure FormActivate(Sender: TObject);

procedure Edit1Change(Sender: TObject);

procedure DBNavigator1BeforeAction(Sender: TObject; Button: TNavigateBtn);

private

{ Private declarations }

public

{ Public declarations }

id:integer;

nazv:string;

admin:boolean;

pass:string;

end;

var

fAdrElement: TfAdrElement;

implementation

{$R \*.dfm}

uses ustudmain,umain;

procedure TfAdrElement.btnSelectClick(Sender: TObject);

begin

id:=DBGrid1.DataSource.DataSet.FieldByName('id').Value;

nazv:=DBGrid1.DataSource.DataSet.FieldByName('nazv').Value;

Close;

end;

procedure TfAdrElement.DBNavigator1BeforeAction(Sender: TObject;

Button: TNavigateBtn);

var s:string;

begin

if (Button=nbInsert) then

begin

if not admin then

begin

s:=InputBox('Пароль', 'Введите пароль','');

if s=pass then

begin

s:=InputBox('Ввод данных', '','');

if s<>'' then

if not dbGrid1.DataSource.DataSet.Locate('nazv',s,[loCaseInsensitive, loPartialKey]) then

begin

dbGrid1.DataSource.DataSet.Append;

dbGrid1.DataSource.DataSet.FieldByName('nazv').AsString:=s;

dbGrid1.DataSource.DataSet.Post;

Abort;

end

else Abort;

end;

end;

end

end;

procedure TfAdrElement.Edit1Change(Sender: TObject);

begin

dbGrid1.DataSource.DataSet.Locate('nazv',Edit1.Text,[loCaseInsensitive, loPartialKey])

end;

procedure TfAdrElement.FormActivate(Sender: TObject);

begin

id:=-1;

nazv:='';

Edit1.Text:='';

fAdrElement.btnSelect.Visible:=true;

DBGrid1.ReadOnly:=not admin;

if admin then fAdrElement.DBNavigator1.VisibleButtons:=[nbInsert,nbDelete,nbPost,nbCancel]

else fAdrElement.DBNavigator1.VisibleButtons:=[nbInsert];

end;

procedure TfAdrElement.FormResize(Sender: TObject);

begin

DBGrid1.Columns[1].Width:=DBGrid1.ClientWidth;

//if fAdresElement.Width<btnSelect.Width then fAdresElement.Width:=btnSelect.Width;

end;

end.

unit udm;

interface

uses

System.SysUtils, System.Classes, Data.DB, Data.Win.ADODB;

type

Tdm = class(TDataModule)

ADOCon1: TADOConnection;

ADOtStudents: TADOTable;

dsStudents: TDataSource;

ADOtGroups: TADOTable;

dsGroups: TDataSource;

ADOtTypes: TADOTable;

dsTypes: TDataSource;

ADOtLevel: TADOTable;

dsLevel: TDataSource;

ADOtLevelid: TAutoIncField;

ADOtLevelshortname: TWideStringField;

ADOtLevelfullname: TWideStringField;

ADOtLevelequivalentAmount: TBCDField;

ADOtSpec: TADOTable;

dsSpec: TDataSource;

ADOtSpecid: TAutoIncField;

ADOtSpecnumspec: TWideStringField;

ADOtSpecnamespec: TWideStringField;

ADOtSpeclevel: TIntegerField;

ADOtSpecshortnamelevel: TStringField;

ADOtGroupsid: TAutoIncField;

ADOtGroupsgroup: TWideStringField;

ADOtGroupsspec: TIntegerField;

ADOtStudentsid: TAutoIncField;

ADOtStudentsF: TWideStringField;

ADOtStudentsI: TWideStringField;

ADOtStudentsO: TWideStringField;

ADOtStudentsGroup: TIntegerField;

ADOtStudentsgender: TWideStringField;

ADOtStudentsbdate: TDateTimeField;

ADOtStudentssnils: TWideStringField;

ADOtTypesid: TAutoIncField;

ADOtTypestypename: TWideStringField;

ADOtStudentsegissotype: TIntegerField;

ADOtStudentsegissotypename: TStringField;

ADOtStudentssnilscheck: TBooleanField;

ADOtStudentsFIO: TStringField;

ADOTable1: TADOTable;

ADOTable1id: TAutoIncField;

ADOTable1F: TWideStringField;

ADOTable1I: TWideStringField;

ADOTable1O: TWideStringField;

ADOTable1Group: TIntegerField;

ADOTable1snilscheck: TBooleanField;

ADOTable1gender: TWideStringField;

ADOTable1bdate: TDateTimeField;

ADOTable1snils: TWideStringField;

ADOTable1egissotype: TIntegerField;

ADOTable1gruppa: TStringField;

ADOTable1gruppaspec: TIntegerField;

ADOTable1specnum: TStringField;

DataSource1: TDataSource;

ADOTable1egissoname: TStringField;

ADOTable1speclevel: TIntegerField;

ADOTable1levelshortname: TStringField;

ADOtLevelcategoryid: TWideStringField;

ADOtTypescategoryid: TWideStringField;

ADOadrElement: TADOTable;

dsAdrElement: TDataSource;

ADOadrElementnazv: TWideStringField;

ADOadrElementid: TAutoIncField;

ADOtStudentsstatus: TIntegerField;

ADOtStudentsZayavDate: TDateTimeField;

ADOtStudentsF\_rp: TWideStringField;

ADOtStudentsI\_rp: TWideStringField;

ADOtStudentsO\_rp: TWideStringField;

ADOtStudentsF\_dp: TWideStringField;

ADOtStudentsI\_dp: TWideStringField;

ADOtStudentsO\_dp: TWideStringField;

ADOtStudentsphone: TWideStringField;

ADOtStudentsInKours: TWideStringField;

ADOtStudentsInSpec: TIntegerField;

ADOtStudentsInEduLvl: TWideStringField;

ADOtStudentsInPriceType: TWideStringField;

ADOtStudentsbPlace: TWideStringField;

ADOtStudentsPassport\_ser: TWideStringField;

ADOtStudentsPassport\_num: TWideStringField;

ADOtStudentsPassport\_vidan: TWideStringField;

ADOtStudentsPassport\_vidan\_data: TDateTimeField;

ADOtStudentsPassport\_kodp: TWideStringField;

ADOtStudentsPassport\_Other: TWideStringField;

ADOtStudentsadrpCountry: TIntegerField;

ADOtStudentsadrpRegion: TIntegerField;

ADOtStudentsadrpDistrict: TIntegerField;

ADOtStudentsadrpLocalityType: TIntegerField;

ADOtStudentsadrpLocality: TIntegerField;

ADOtStudentsadrpStreet: TIntegerField;

ADOtStudentsadrpHouse: TWideStringField;

ADOtStudentsadrpHousePart: TWideStringField;

ADOtStudentsadrpBuilding: TWideStringField;

ADOtStudentsadrpApartment: TWideStringField;

ADOtStudentsadrfCountry: TIntegerField;

ADOtStudentsadrfRegion: TIntegerField;

ADOtStudentsadrfDistrict: TIntegerField;

ADOtStudentsadrfLocalityType: TIntegerField;

ADOtStudentsadrfLocality: TIntegerField;

ADOtStudentsadrfStreet: TIntegerField;

ADOtStudentsadrfHouse: TWideStringField;

ADOtStudentsadrfHousePart: TWideStringField;

ADOtStudentsadrfBuilding: TWideStringField;

ADOtStudentsadrsCountry: TIntegerField;

ADOtStudentsadrsRegion: TIntegerField;

ADOtStudentsadrsDistrict: TIntegerField;

ADOtStudentsadrsLocalityType: TIntegerField;

ADOtStudentsadrsLocality: TIntegerField;

ADOtStudentseduSchool: TIntegerField;

ADOtStudentseduDocType: TIntegerField;

ADOtStudentseduDocDate: TDateTimeField;

ADOtStudentseduDocSer: TWideStringField;

ADOtStudentseduDocNum: TWideStringField;

ADOtStatus: TADOTable;

dsStatus: TDataSource;

ADOtStudentsadrfApartment: TWideStringField;

ADOtStudentseduEndYear: TWideStringField;

ADOtStudentseduKlass: TWideStringField;

ADOtStudentsr1Type: TWideStringField;

ADOtStudentsr1F: TWideStringField;

ADOtStudentsr1I: TWideStringField;

ADOtStudentsr1O: TWideStringField;

ADOtStudentsr1Work: TWideStringField;

ADOtStudentsr1D: TWideStringField;

ADOtStudentsr1Phone: TWideStringField;

ADOtStudentsr2Type: TWideStringField;

ADOtStudentsr2F: TWideStringField;

ADOtStudentsr2I: TWideStringField;

ADOtStudentsr2O: TWideStringField;

ADOtStudentsr2Work: TWideStringField;

ADOtStudentsr2D: TWideStringField;

ADOtStudentsr2Phone: TWideStringField;

ADOtStudentsDeti: TBooleanField;

ADOtStudentsOpeka: TBooleanField;

ADOtStudentsOpekaFIO: TWideStringField;

ADOtStudentsSirota: TBooleanField;

ADOtStudentsBezRoditel: TBooleanField;

ADOtStudentsInvalid: TBooleanField;

ADOtStudentsDopSvedenia: TWideMemoField;

ADOtStudentsInvalidDiagnoz: TWideStringField;

ADOtStudentsPassport\_type: TWideStringField;

tPredmet: TADOTable;

dsPredmet: TDataSource;

tPrepod: TADOTable;

dsPrepod: TDataSource;

dsGNagr: TDataSource;

tGNagr: TADOTable;

tGNagrid: TAutoIncField;

tGNagridGroup: TIntegerField;

tGNagridPredmet: TIntegerField;

tGNagrHours: TSmallintField;

tGNagrPredmet: TStringField;

tGNagridPrepod: TIntegerField;

tGNagrPrepod: TStringField;

dsPNagr: TDataSource;

tPNagr: TADOTable;

AutoIncField1: TAutoIncField;

IntegerField1: TIntegerField;

IntegerField2: TIntegerField;

StringField1: TStringField;

SmallintField1: TSmallintField;

tPNagrGroup: TStringField;

tPNagridPrepod: TIntegerField;

ADOtStudentspoms: TWideStringField;

ADOtStudentsinn: TWideStringField;

ADOtStudentsstatuschangereason: TWideStringField;

ADOtStudentsInFinalDate: TDateTimeField;

ADOtStudentsInFinalReason: TWideStringField;

q1: TADOQuery;

ds1: TDataSource;

tStudPriem: TADOTable;

AutoIncField2: TAutoIncField;

WideStringField1: TWideStringField;

WideStringField2: TWideStringField;

WideStringField3: TWideStringField;

IntegerField3: TIntegerField;

StringField3: TStringField;

IntegerField5: TIntegerField;

DateTimeField2: TDateTimeField;

WideStringField13: TWideStringField;

IntegerField6: TIntegerField;

WideStringField15: TWideStringField;

dsStudPriem: TDataSource;

tStudPriemGruppa: TStringField;

ADOtStudentseduScore: TWideStringField;

tStudPriemeduScore: TWideStringField;

tStudPriemstatuschangereason: TWideStringField;

tStudPriemInFinalDate: TDateTimeField;

tStudPriemInFinalReason: TWideStringField;

tStudPriemgroupchangereason: TWideStringField;

ADOtStudentsgroupchangereason: TWideStringField;

ADOtStudentsGruppa: TStringField;

ADOtGroupskurs: TSmallintField;

ADOtStudentsFinalPriceType: TWideStringField;

ADOtStatusid: TAutoIncField;

ADOtStatusstatus: TWideStringField;

ADOtStudentsstatusname: TStringField;

ADOtStudentsFinalPriceTypeChangeReason: TWideStringField;

tUspMain: TADOTable;

dsUspMain: TDataSource;

tUspGroup: TADOTable;

dsUspGroup: TDataSource;

dsUspPredmet: TDataSource;

tUspPredmet: TADOTable;

tUspStudent: TADOTable;

dsUspStudent: TDataSource;

tUspGroupid: TAutoIncField;

tUspGroupiduspmain: TIntegerField;

tUspGroupidgroup: TIntegerField;

tUspGroupGroup: TStringField;

tUspMainid: TAutoIncField;

tUspMainuspYear: TWideStringField;

tUspMainuspSemestr: TWideStringField;

tUspPredmetid: TAutoIncField;

tUspPredmetiduspgroup: TIntegerField;

tUspPredmetidpredmet: TIntegerField;

tUspPredmettype: TWideStringField;

tUspPredmetPredmet: TStringField;

tUspStudentid: TAutoIncField;

tUspStudentidusppredmet: TIntegerField;

tUspStudentidstudent: TIntegerField;

tUspStudentrez: TWideStringField;

tUspStudentFIO: TStringField;

qUsp: TADOQuery;

dsUsp: TDataSource;

tUspStudentsort: TSmallintField;

tSotr: TADOTable;

dsSotr: TDataSource;

dsDolznost: TDataSource;

tDolznost: TADOTable;

tDolznostid: TAutoIncField;

tDolznostnazv: TWideStringField;

tDolznostMainType: TWideStringField;

tDolznostSubType: TWideStringField;

tSotrid: TAutoIncField;

tSotrF: TWideStringField;

tSotrI: TWideStringField;

tSotrO: TWideStringField;

tSotrBDate: TDateTimeField;

tSotrGender: TWideStringField;

tSotrEduType: TWideStringField;

tSotrEduPed: TWideStringField;

tSotrEduStepen: TWideStringField;

tSotrEduZvanie: TWideStringField;

tSotrKvalif: TWideStringField;

tSotrDolznost: TIntegerField;

tSotrFIO: TStringField;

tSotrNazvDolznost: TStringField;

procedure ADOtStudentsCalcFields(DataSet: TDataSet);

procedure tStudPriemCalcFields(DataSet: TDataSet);

procedure tSotrCalcFields(DataSet: TDataSet);

private

{ Private declarations }

public

{ Public declarations }

end;

var

dm: Tdm;

implementation

{%CLASSGROUP 'Vcl.Controls.TControl'}

{$R \*.dfm}

procedure Tdm.ADOtStudentsCalcFields(DataSet: TDataSet);

var s:string;

begin

if ADOtStudents.RecordCount>0 then

begin

s:=ADOtStudents.FieldByName('f').AsString+' '+copy(ADOtStudents.FieldByName('i').AsString,1,1)+'.';

if ADOtStudents.FieldByName('o').AsString<>'' then

s:=s+copy(ADOtStudents.FieldByName('o').AsString,1,1)+'.';

ADOtStudents.FieldByName('FIO').AsString:=s;

end;

end;

procedure Tdm.tSotrCalcFields(DataSet: TDataSet);

var s:string;

begin

if tSotr.RecordCount>0 then

begin

s:=tSotr.FieldByName('f').AsString+' '+copy(tSotr.FieldByName('i').AsString,1,1)+'.';

if tSotr.FieldByName('o').AsString<>'' then

s:=s+copy(tSotr.FieldByName('o').AsString,1,1)+'.';

tSotr.FieldByName('FIO').AsString:=s;

end;

end;

procedure Tdm.tStudPriemCalcFields(DataSet: TDataSet);

var s:string;

begin

if tStudPriem.RecordCount>0 then

begin

s:=tStudPriem.FieldByName('f').AsString+' '+copy(tStudPriem.FieldByName('i').AsString,1,1)+'.';

if tStudPriem.FieldByName('o').AsString<>'' then

s:=s+copy(tStudPriem.FieldByName('o').AsString,1,1)+'.';

tStudPriem.FieldByName('FIO').AsString:=s;

end;

end;

end.

unit ugroup;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.ExtCtrls, Vcl.DBCtrls, Data.DB,

Vcl.Grids, Vcl.DBGrids;

type

TfGroup = class(TForm)

DBGrid1: TDBGrid;

Panel1: TPanel;

DBGrid2: TDBGrid;

DBNavigator1: TDBNavigator;

procedure DBGrid1TitleClick(Column: TColumn);

procedure FormActivate(Sender: TObject);

procedure FormClose(Sender: TObject; var Action: TCloseAction);

procedure DBNavigator1BeforeAction(Sender: TObject; Button: TNavigateBtn);

private

{ Private declarations }

public

{ Public declarations }

end;

var

fGroup: TfGroup;

implementation

{$R \*.dfm}

uses udm;

procedure TfGroup.DBGrid1TitleClick(Column: TColumn);

begin

case Column.Index of

0:dm.ADOtSpec.IndexFieldNames:='numspec';

1:dm.ADOtSpec.IndexFieldNames:='namespec';

end;

end;

procedure TfGroup.DBNavigator1BeforeAction(Sender: TObject;

Button: TNavigateBtn);

begin

if Button=nbDelete then

if dm.ADOtStudents.Locate('group',dm.ADOtGroups.FieldByName('id').Value,[]) then

begin

ShowMessage('Удаление невозможно.'+#13+'Сначала удалите связанные записи из справочника "Студенты"');

Abort;

end;

end;

procedure TfGroup.FormActivate(Sender: TObject);

begin

dm.ADOtGroups.MasterSource:=dm.dsSpec;

dm.ADOtGroups.MasterFields:='id';

dm.ADOtGroups.IndexFieldNames:='spec';

end;

procedure TfGroup.FormClose(Sender: TObject; var Action: TCloseAction);

begin

dm.ADOtGroups.MasterSource:=nil;

dm.ADOtGroups.MasterFields:='';

dm.ADOtGroups.IndexFieldNames:='';

end;

end.

unit ulevel;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.DBCGrids, Vcl.ExtCtrls, Vcl.DBCtrls,

Data.DB, Vcl.Grids, Vcl.DBGrids;

type

TfLevel = class(TForm)

DBNavigator1: TDBNavigator;

DBGrid1: TDBGrid;

procedure DBNavigator1BeforeAction(Sender: TObject; Button: TNavigateBtn);

private

{ Private declarations }

public

{ Public declarations }

end;

var

fLevel: TfLevel;

implementation

{$R \*.dfm}

uses udm;

procedure TfLevel.DBNavigator1BeforeAction(Sender: TObject;

Button: TNavigateBtn);

begin

if Button=nbDelete then

if dm.ADOtSpec.Locate('level',dm.ADOtLevel.FieldByName('id').Value,[]) then

begin

ShowMessage('Удаление невозможно.'+#13+'Сначала удалите связанные записи из справочника "Специальности"');

Abort;

end;

end;

end.

unit uspec;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.ExtCtrls, Vcl.DBCtrls, Data.DB,

Vcl.Grids, Vcl.DBGrids, Vcl.StdCtrls, Vcl.Mask;

type

TfSpec = class(TForm)

DBNavigator1: TDBNavigator;

DBGrid1: TDBGrid;

Panel1: TPanel;

DBEdit1: TDBEdit;

DBEdit2: TDBEdit;

DBLookupComboBox1: TDBLookupComboBox;

procedure DBGrid1TitleClick(Column: TColumn);

procedure DBNavigator1BeforeAction(Sender: TObject; Button: TNavigateBtn);

private

{ Private declarations }

public

{ Public declarations }

end;

var

fSpec: TfSpec;

implementation

{$R \*.dfm}

uses udm;

procedure TfSpec.DBGrid1TitleClick(Column: TColumn);

begin

//ShowMessage(IntToStr(Column.Index));

case Column.Index of

0:dm.ADOtSpec.IndexFieldNames:='numspec';

1:dm.ADOtSpec.IndexFieldNames:='namespec';

2:dm.ADOtSpec.IndexFieldNames:='level';

end;

end;

procedure TfSpec.DBNavigator1BeforeAction(Sender: TObject;

Button: TNavigateBtn);

begin

if Button=nbDelete then

if dm.ADOtGroups.Locate('spec',dm.ADOtSpec.FieldByName('id').Value,[]) then

begin

ShowMessage('Удаление невозможно.'+#13+'Сначала удалите связанные записи из справочника "Группы"');

Abort;

end;

end;

end.

unit ustud1;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.Mask, Vcl.ComCtrls, DBCtrls, Data.DB;

type

TfStud1 = class(TForm)

PageControl1: TPageControl;

TabSheet1: TTabSheet;

TabSheet2: TTabSheet;

TabSheet3: TTabSheet;

edF\_i: TEdit;

edI\_i: TEdit;

edO\_i: TEdit;

Label1: TLabel;

Label2: TLabel;

Label3: TLabel;

Label6: TLabel;

edO\_r: TEdit;

edI\_r: TEdit;

edF\_r: TEdit;

Label7: TLabel;

edO\_d: TEdit;

edI\_d: TEdit;

edF\_d: TEdit;

Label10: TLabel;

Label8: TLabel;

edPhone: TMaskEdit;

GroupBox1: TGroupBox;

Label22: TLabel;

cbPasport: TComboBox;

lSer: TLabel;

edSer: TMaskEdit;

lNum: TLabel;

edNum: TMaskEdit;

lVidanDate: TLabel;

dtpVidanDate: TDateTimePicker;

lKod: TLabel;

edKod: TMaskEdit;

lVidanKem: TLabel;

edVidanKem: TEdit;

mInoi: TMemo;

edSnils: TMaskEdit;

lSnils: TLabel;

Label28: TLabel;

mAdrProp: TMemo;

btnAdrProp: TButton;

Label30: TLabel;

btnAdrFact: TButton;

mAdrFact: TMemo;

Button1: TButton;

Button2: TButton;

Button4: TButton;

Button5: TButton;

Label9: TLabel;

cbGender: TComboBox;

Label49: TLabel;

edINN: TMaskEdit;

Label50: TLabel;

edPOMS: TEdit;

Label18: TLabel;

dtpBDate: TDateTimePicker;

Label19: TLabel;

edBPlace: TEdit;

GroupBox2: TGroupBox;

Label5: TLabel;

Label11: TLabel;

edNewGroup: TEdit;

edNewGroupReason: TEdit;

Button6: TButton;

Label4: TLabel;

DBEdit1: TDBEdit;

Label38: TLabel;

edSchoolName: TEdit;

btnSchool: TButton;

btnSchoolClear: TButton;

Button3: TButton;

mAdrSchool: TMemo;

Label37: TLabel;

btnAdrSchool: TButton;

lScore: TLabel;

edScore: TMaskEdit;

GroupBox3: TGroupBox;

Label14: TLabel;

edAkademReason: TEdit;

Button7: TButton;

GroupBox4: TGroupBox;

Label15: TLabel;

Label16: TLabel;

edRestoreGroup: TEdit;

edRestoreReason: TEdit;

Button8: TButton;

GroupBox6: TGroupBox;

Label17: TLabel;

edPriceTypeReason: TEdit;

Button10: TButton;

Label12: TLabel;

DBEdit2: TDBEdit;

cbPlata: TComboBox;

Label13: TLabel;

procedure AdrP;

procedure AdrF;

procedure AdrS;

procedure cbPasportChange(Sender: TObject);

procedure FormActivate(Sender: TObject);

procedure btnAdrPropClick(Sender: TObject);

procedure btnAdrFactClick(Sender: TObject);

procedure btnSchoolClick(Sender: TObject);

procedure btnSchoolClearClick(Sender: TObject);

procedure btnAdrSchoolClick(Sender: TObject);

procedure Button3Click(Sender: TObject);

procedure Button1Click(Sender: TObject);

procedure Button2Click(Sender: TObject);

procedure Button5Click(Sender: TObject);

procedure Button4Click(Sender: TObject);

procedure Button6Click(Sender: TObject);

procedure Button7Click(Sender: TObject);

procedure Button8Click(Sender: TObject);

procedure Button10Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

idStud:integer;

idCountry:integer;

idRegion:integer;

idDistrict:integer;

idLocalityType:integer;

idLocality:integer;

idStreet:integer;

edCountry:string;

edRegion:string;

edDistrict:string;

edLocalityType:string;

edLocality:string;

edStreet:string;

edHouse:string;

edHousePart:string;

edBuilding:string;

edApartment:string;

idfCountry:integer;

idfRegion:integer;

idfDistrict:integer;

idfLocalityType:integer;

idfLocality:integer;

idfStreet:integer;

edfCountry:string;

edfRegion:string;

edfDistrict:string;

edfLocalityType:string;

edfLocality:string;

edfStreet:string;

edfHouse:string;

edfHousePart:string;

edfBuilding:string;

edfApartment:string;

idSchool:integer;

edSchool:string;

idsCountry:integer;

idsRegion:integer;

idsDistrict:integer;

idsLocalityType:integer;

idsLocality:integer;

idsStreet:integer;

edsCountry:string;

edsRegion:string;

edsDistrict:string;

edsLocalityType:string;

edsLocality:string;

end;

var

fStud1: TfStud1;

implementation

{$R \*.dfm}

uses uadres, udm, uAdresElement, ustudmain;

procedure TfStud1.AdrP;

var adr:string;

begin

adr:='';mAdrProp.Lines.Clear;

if edCountry<>'' then adr:=adr+edCountry;

if (edRegion<>'') then

begin

if (length(adr)+length(edRegion)<250) then adr:=adr+', '+edRegion

else

begin

mAdrProp.Lines.Append(adr);

adr:=edRegion;

end;

end;

if (edDistrict<>'') then

begin

if (length(adr)+length(edDistrict)<240) then adr:=adr+', район '+edDistrict

else

begin

mAdrProp.Lines.Append(adr);

adr:='район '+edDistrict

end;

end;

if (edLocalityType<>'') then //Есть тип населенного пункта

begin

if (edLocality<>'') then //Есть населенный пункт

begin

if (length(adr)+length(edLocalityType)+length(edLocality)>=250) then

begin

mAdrProp.Lines.Append(adr);

adr:=edLocalityType+' '+edLocality;

end

else adr:=adr+', '+edLocalityType+' '+edLocality;

end;

end

else //Нет типа населенного пункта

begin

if (edLocality<>'') then //Есть населенный пункт

begin

if (length(adr)+length(edLocality)>=250) then

begin

mAdrProp.Lines.Append(adr);

adr:=', '+edLocality;

end

else adr:=adr+', '+edLocality;

end;

end;

if (edStreet<>'') then

begin

if (length(adr)+length(edStreet)<250) then adr:=adr+', улица '+edStreet

else

begin

mAdrProp.Lines.Append(adr);

adr:='улица '+edStreet;

end;

end;

if (edHouse<>'') then

begin

if (length(adr)+length(edHouse)<250) then adr:=adr+', дом '+edHouse

else

begin

mAdrProp.Lines.Append(adr);

adr:='дом '+edHouse;

end;

end;

if (edHousePart<>'') then

begin

if (length(adr)+length(edHousePart)<240) then adr:=adr+', корпус '+edHousePart

else

begin

mAdrProp.Lines.Append(adr);

adr:='корпус '+edHousePart;

end;

end;

if (edBuilding<>'') then

begin

if (length(adr)+length(edBuilding)<240) then adr:=adr+', строение '+edBuilding

else

begin

mAdrProp.Lines.Append(adr);

adr:='строение '+edBuilding;

end;

end;

if (edApartment<>'') then

begin

if (length(adr)+length(edApartment)<240) then adr:=adr+', квартира '+edApartment

else

begin

mAdrProp.Lines.Append(adr);

adr:='квартира '+edApartment;

end;

end;

mAdrProp.Lines.Append(adr);

end;

procedure TfStud1.AdrF;

var adr:string;

begin

adr:='';mAdrFact.Lines.Clear;

if edfCountry<>'' then adr:=adr+edfCountry;

if (edfRegion<>'') then

begin

if (length(adr)+length(edfRegion)<250) then adr:=adr+', '+edfRegion

else

begin

mAdrFact.Lines.Append(adr);

adr:=edfRegion;

end;

end;

if (edfDistrict<>'') then

begin

if (length(adr)+length(edfDistrict)<240) then adr:=adr+', район '+edfDistrict

else

begin

mAdrFact.Lines.Append(adr);

adr:='район '+edfDistrict

end;

end;

if (edfLocalityType<>'') then //Есть тип населенного пункта

begin

if (edfLocality<>'') then //Есть населенный пункт

begin

if (length(adr)+length(edfLocalityType)+length(edfLocality)>=250) then

begin

mAdrFact.Lines.Append(adr);

adr:=edfLocalityType+' '+edfLocality;

end

else adr:=adr+', '+edfLocalityType+' '+edfLocality;

end;

end

else //Нет типа населенного пункта

begin

if (edfLocality<>'') then //Есть населенный пункт

begin

if (length(adr)+length(edfLocality)>=250) then

begin

mAdrFact.Lines.Append(adr);

adr:=', '+edfLocality;

end

else adr:=adr+', '+edfLocality;

end;

end;

if (edfStreet<>'') then

begin

if (length(adr)+length(edfStreet)<250) then adr:=adr+', улица '+edfStreet

else

begin

mAdrFact.Lines.Append(adr);

adr:='улица '+edfStreet;

end;

end;

if (edfHouse<>'') then

begin

if (length(adr)+length(edfHouse)<250) then adr:=adr+', дом '+edfHouse

else

begin

mAdrFact.Lines.Append(adr);

adr:='дом '+edfHouse;

end;

end;

if (edfHousePart<>'') then

begin

if (length(adr)+length(edfHousePart)<240) then adr:=adr+', корпус '+edfHousePart

else

begin

mAdrFact.Lines.Append(adr);

adr:='корпус '+edfHousePart;

end;

end;

if (edfBuilding<>'') then

begin

if (length(adr)+length(edfBuilding)<240) then adr:=adr+', строение '+edfBuilding

else

begin

mAdrFact.Lines.Append(adr);

adr:='строение '+edfBuilding;

end;

end;

if (edfApartment<>'') then

begin

if (length(adr)+length(edfApartment)<240) then adr:=adr+', квартира '+edfApartment

else

begin

mAdrFact.Lines.Append(adr);

adr:='квартира '+edfApartment;

end;

end;

mAdrFact.Lines.Append(adr);

end;

procedure TfStud1.AdrS;

var adr:string;

begin

adr:='';mAdrSchool.Lines.Clear;

if edsCountry<>'' then adr:=adr+edsCountry;

if (edsRegion<>'') then

begin

if (length(adr)+length(edsRegion)<250) then adr:=adr+', '+edsRegion

else

begin

mAdrSchool.Lines.Append(adr);

adr:=edsRegion;

end;

end;

if (edsDistrict<>'') then

begin

if (length(adr)+length(edsDistrict)<240) then adr:=adr+', район '+edsDistrict

else

begin

mAdrSchool.Lines.Append(adr);

adr:='район '+edsDistrict

end;

end;

if (edsLocalityType<>'') then //Есть тип населенного пункта

begin

if (edsLocality<>'') then //Есть населенный пункт

begin

if (length(adr)+length(edsLocalityType)+length(edsLocality)>=250) then

begin

mAdrSchool.Lines.Append(adr);

adr:=edsLocalityType+' '+edsLocality;

end

else adr:=adr+', '+edsLocalityType+' '+edsLocality;

end;

end

else //Нет типа населенного пункта

begin

if (edsLocality<>'') then //Есть населенный пункт

begin

if (length(adr)+length(edsLocality)>=250) then

begin

mAdrSchool.Lines.Append(adr);

adr:=', '+edsLocality;

end

else adr:=adr+', '+edsLocality;

end;

end;

mAdrSchool.Lines.Append(adr);

end;

procedure TfStud1.btnAdrFactClick(Sender: TObject);

var adr:string;

begin

fAdres.edCountry.Text:=edfCountry;

fAdres.edRegion.Text:=edfRegion;

fAdres.edDistrict.Text:=edfDistrict;

fAdres.edLocalityType.Text:=edfLocalityType;

fAdres.edLocality.Text:=edfLocality;

fAdres.edStreet.Text:=edfStreet;

fAdres.edHouse.Text:=edfHouse;

fAdres.edHousePart.Text:=edfHousepart;

fAdres.edBuilding.Text:=edfBuilding;

fAdres.edApartment.Text:=edfApartment;

fAdres.idCountry:=idfCountry;

fAdres.idRegion:=idfRegion;

fAdres.idDistrict:=idfDistrict;

fAdres.idLocalityType:=idfLocalityType;

fAdres.idLocality:=idfLocality;

fAdres.idStreet:=idfStreet;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

fAdres.school:=false;

fAdres.Panel2.Visible:=true;

fAdres.admin:=fStudMain.admin;

fAdres.pass:=fStudMain.pass;

fAdres.ShowModal;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

edfCountry:=fAdres.edCountry.Text;

edfRegion:=fAdres.edRegion.Text;

edfDistrict:=fAdres.edDistrict.Text;

edfLocalityType:=fAdres.edLocalityType.Text;

edfLocality:=fAdres.edLocality.Text;

edfStreet:=fAdres.edStreet.Text;

edfHouse:=fAdres.edHouse.Text;

edfHousePart:=fAdres.edHousepart.Text;

edfBuilding:=fAdres.edBuilding.Text;

edfApartment:=fAdres.edApartment.Text;

idfCountry:=fAdres.idCountry;

idfRegion:=fAdres.idRegion;

idfDistrict:=fAdres.idDistrict;

idfLocalityType:=fAdres.idLocalityType;

idfLocality:=fAdres.idLocality;

idfStreet:=fAdres.idStreet;

AdrF;

end;

procedure TfStud1.btnAdrPropClick(Sender: TObject);

var adr:string;

begin

fAdres.edCountry.Text:=edCountry;

fAdres.edRegion.Text:=edRegion;

fAdres.edDistrict.Text:=edDistrict;

fAdres.edLocalityType.Text:=edLocalityType;

fAdres.edLocality.Text:=edLocality;

fAdres.edStreet.Text:=edStreet;

fAdres.edHouse.Text:=edHouse;

fAdres.edHousePart.Text:=edHousepart;

fAdres.edBuilding.Text:=edBuilding;

fAdres.edApartment.Text:=edApartment;

fAdres.idCountry:=idCountry;

fAdres.idRegion:=idRegion;

fAdres.idDistrict:=idDistrict;

fAdres.idLocalityType:=idLocalityType;

fAdres.idLocality:=idLocality;

fAdres.idStreet:=idStreet;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

fAdres.school:=false;

fAdres.Panel2.Visible:=true;

fAdres.admin:=fStudMain.admin;

fAdres.pass:=fStudMain.pass;

fAdres.ShowModal;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

edCountry:=fAdres.edCountry.Text;

edRegion:=fAdres.edRegion.Text;

edDistrict:=fAdres.edDistrict.Text;

edLocalityType:=fAdres.edLocalityType.Text;

edLocality:=fAdres.edLocality.Text;

edStreet:=fAdres.edStreet.Text;

edHouse:=fAdres.edHouse.Text;

edHousePart:=fAdres.edHousepart.Text;

edBuilding:=fAdres.edBuilding.Text;

edApartment:=fAdres.edApartment.Text;

idCountry:=fAdres.idCountry;

idRegion:=fAdres.idRegion;

idDistrict:=fAdres.idDistrict;

idLocalityType:=fAdres.idLocalityType;

idLocality:=fAdres.idLocality;

idStreet:=fAdres.idStreet;

AdrP;

end;

procedure TfStud1.btnSchoolClearClick(Sender: TObject);

begin

idSchool:=-1;

edSchoolName.Text:='';

edSchool:='';

end;

procedure TfStud1.btnSchoolClick(Sender: TObject);

begin

dm.ADOadrElement.Active:=false;

dm.ADOadrElement.TableName:='School';

dm.ADOadrElement.IndexFieldNames:='nazv';

dm.ADOadrElement.Active:=true;

fAdrElement.DBGrid1.Columns[1].Title.Caption:='Школа';

fAdrElement.DBGrid1.Columns[1].Width:=fAdrElement.DBGrid1.ClientWidth;

fAdrElement.btnSelect.Visible:=true;

fAdrElement.DBNavigator1.VisibleButtons:=[nbInsert];

fAdrElement.ShowModal;

if (fAdrElement.id<>-1)or(edSchoolName.Text='') then

begin

edSchoolName.Text:=fAdrElement.nazv;

idSchool:=fAdrElement.id;

end;

end;

procedure TfStud1.Button10Click(Sender: TObject);

begin

//Изменение условий обучения

if edPriceTypeReason.Text<>'' then

begin

dm.ADOtStudents.Edit;

dm.ADOtStudents.FieldByName('FinalPriceType').Value:=cbPlata.Text;

dm.ADOtStudents.FieldByName('FinalPriceTypeChangeReason').Value:=edPriceTypeReason.Text;

dm.ADOtStudents.Post;

edPriceTypeReason.Text:='';

ShowMessage('Условия обучения успешно изменены');

end

else ShowMessage('Укажите основание');

end;

procedure TfStud1.Button1Click(Sender: TObject);

begin

mAdrProp.Lines.Clear;

edCountry:='';

edRegion:='';

edDistrict:='';

edLocalityType:='';

edLocality:='';

edStreet:='';

edHouse:='';

edHousePart:='';

edBuilding:='';

edApartment:='';

idCountry:=-1;

idRegion:=-1;

idDistrict:=-1;

idLocalityType:=-1;

idLocality:=-1;

idStreet:=-1;

end;

procedure TfStud1.Button2Click(Sender: TObject);

begin

mAdrFact.Lines.Clear;

edfCountry:='';

edfRegion:='';

edfDistrict:='';

edfLocalityType:='';

edfLocality:='';

edfStreet:='';

edfHouse:='';

edfHousePart:='';

edfBuilding:='';

edfApartment:='';

idfCountry:=-1;

idfRegion:=-1;

idfDistrict:=-1;

idfLocalityType:=-1;

idfLocality:=-1;

idfStreet:=-1;

end;

procedure TfStud1.Button3Click(Sender: TObject);

begin

mAdrSchool.Lines.Clear;

edsCountry:='';

edsRegion:='';

edsDistrict:='';

edsLocalityType:='';

edsLocality:='';

idsCountry:=-1;

idsRegion:=-1;

idsDistrict:=-1;

idsLocalityType:=-1;

idsLocality:=-1;

end;

procedure TfStud1.Button4Click(Sender: TObject);

var ok:boolean;

begin

ok:=true;

if not ok then ShowMessage('Заполните все обязательные поля, отмеченные символом красной звездочки')

else

begin

if idstud=-1 then

begin

dm.ADOtStudents.Append;

dm.ADOtStatus.Active:=true;

if dm.ADOtStatus.Locate('status','Абитуриент',[loCaseInsensitive, loPartialKey]) then

dm.ADOtStudentsStatus.Value:=dm.ADOtStatus.FieldByName('Id').Value

else dm.ADOtStudentsStatus.Value:=0;

dm.ADOtStatus.Active:=false;

dm.ADOtStudentsZayavDate.Value:=Date();

end

else dm.ADOtStudents.Edit;

dm.ADOtStudentsF.Value:=edF\_i.Text;

dm.ADOtStudentsI.Value:=edI\_i.Text;

dm.ADOtStudentsO.Value:=edO\_i.Text;

dm.ADOtStudentsF\_rp.Value:=edF\_r.Text;

dm.ADOtStudentsI\_rp.Value:=edI\_r.Text;

dm.ADOtStudentsO\_rp.Value:=edO\_r.Text;

dm.ADOtStudentsF\_dp.Value:=edF\_d.Text;

dm.ADOtStudentsI\_dp.Value:=edI\_d.Text;

dm.ADOtStudentsO\_dp.Value:=edO\_d.Text;

dm.ADOtStudentsPhone.Value:=edPhone.Text;

dm.ADOtStudentsGender.Value:=cbGender.Text;

dm.ADOtStudentsBDate.Value:=dtpbDate.Date;

dm.ADOtStudentsBPlace.Value:=edBPlace.Text;

dm.ADOtStudentsPassport\_Type.Value:=cbPasport.Text;

dm.ADOtStudentsPassport\_Ser.Value:=edSer.Text;

dm.ADOtStudentsPassport\_Num.Value:=edNum.Text;

dm.ADOtStudentsPassport\_Vidan.Value:=edVidanKem.Text;

dm.ADOtStudentsPassport\_Vidan\_Data.Value:=dtpVidanDate.Date;

dm.ADOtStudentsPassport\_Kodp.Value:=edKod.Text;

dm.ADOtStudentsPassport\_Other.Value:=mInoi.Text;

dm.ADOtStudentsSnils.Value:=edSnils.Text;

dm.ADOtStudentsINN.Value:=edINN.Text;

dm.ADOtStudentsPOMS.Value:=edPOMS.Text;

dm.ADOtStudentseduScore.AsString:=edScore.Text;

if idCountry<>-1 then dm.ADOtStudentsAdrPCountry.Value:=idCountry;

if idRegion<>-1 then dm.ADOtStudentsAdrPRegion.Value:=idRegion;

if idDistrict<>-1 then dm.ADOtStudentsAdrPDistrict.Value:=idDistrict;

if idLocalityType<>-1 then dm.ADOtStudentsAdrPLocalityType.Value:=idLocalityType;

if idLocality<>-1 then dm.ADOtStudentsAdrPLocality.Value:=idLocality;

if idStreet<>-1 then dm.ADOtStudentsAdrPStreet.Value:=idStreet;

dm.ADOtStudentsAdrPHouse.Value:=edHouse;

dm.ADOtStudentsAdrPHousePart.Value:=edHousePart;

dm.ADOtStudentsAdrPBuilding.Value:=edBuilding;

dm.ADOtStudentsAdrPApartment.Value:=edApartment;

if idfCountry<>-1 then dm.ADOtStudentsAdrFCountry.Value:=idfCountry;

if idfRegion<>-1 then dm.ADOtStudentsAdrFRegion.Value:=idfRegion;

if idfDistrict<>-1 then dm.ADOtStudentsAdrFDistrict.Value:=idfDistrict;

if idfLocalityType<>-1 then dm.ADOtStudentsAdrFLocalityType.Value:=idfLocalityType;

if idfLocality<>-1 then dm.ADOtStudentsAdrFLocality.Value:=idfLocality;

if idfStreet<>-1 then dm.ADOtStudentsAdrFStreet.Value:=idfStreet;

dm.ADOtStudentsAdrFHouse.Value:=edfHouse;

dm.ADOtStudentsAdrFHousePart.Value:=edfHousePart;

dm.ADOtStudentsAdrFBuilding.Value:=edfBuilding;

dm.ADOtStudentsAdrFApartment.Value:=edfApartment;

if idSchool<>-1 then dm.ADOtStudentseduSchool.Value:=idSchool;

if idsCountry<>-1 then dm.ADOtStudentsAdrSCountry.Value:=idsCountry;

if idsRegion<>-1 then dm.ADOtStudentsAdrSRegion.Value:=idsRegion;

if idsDistrict<>-1 then dm.ADOtStudentsAdrSDistrict.Value:=idsDistrict;

if idsLocalityType<>-1 then dm.ADOtStudentSAdrPLocalityType.Value:=idsLocalityType;

if idsLocality<>-1 then dm.ADOtStudentsAdrSLocality.Value:=idsLocality;

dm.ADOtStudents.Post;

idStud:=dm.ADOtStudentsId.Value;

Close;

end;

end;

procedure TfStud1.Button5Click(Sender: TObject);

begin

mAdrFact.Lines:=mAdrProp.Lines;

edfCountry:=edCountry;

edfRegion:=edRegion;

edfDistrict:=edDistrict;

edfLocalityType:=edLocalityType;

edfLocality:=edLocality;

edfStreet:=edStreet;

edfHouse:=edHouse;

edfHousePart:=edHousepart;

edfBuilding:=edBuilding;

edfApartment:=edApartment;

idfCountry:=idCountry;

idfRegion:=idRegion;

idfDistrict:=idDistrict;

idfLocalityType:=idLocalityType;

idfLocality:=idLocality;

idfStreet:=idStreet;

end;

procedure TfStud1.Button6Click(Sender: TObject);

begin

//Перевод в новую группу

if dm.ADOtStudents.FieldByName('status').Value=2 then

begin

dm.ADOtGroups.Filtered:=false;

if dm.ADOtGroups.Locate('Group',edNewGroup.Text,[]) then

begin

if edNewGroupReason.Text<>'' then

begin

dm.ADOtStudents.Edit;

dm.ADOtStudents.FieldByName('Group').Value:=dm.ADOtGroups.FieldByName('id').Value;

dm.ADOtStudents.FieldByName('GroupChangeReason').Value:=edNewGroupReason.Text;

dm.ADOtStudents.Post;

edNewGroupReason.Text:='';

edNewGroup.Text:='';

ShowMessage('Перевод совершен успешно');

end

else ShowMessage('Укажите основание');

end

else ShowMessage('Такой группы не существует');

end

else ShowMessage('Перевод не возможен');

end;

procedure TfStud1.Button7Click(Sender: TObject);

begin

//Академка

if dm.ADOtStudents.FieldByName('status').Value=2 then

begin

if edAkademReason.Text<>'' then

begin

dm.ADOtStudents.Edit;

dm.ADOtStudents.FieldByName('status').Value:=3;

dm.ADOtStudents.FieldByName('StatusChangeReason').Value:=edAkademReason.Text;

dm.ADOtStudents.Post;

edAkademReason.Text:='';

ShowMessage('Студент успешно отправлен в академический отпуск');

end

else ShowMessage('Укажите основание');

end

else ShowMessage('Отправить студента в академический отпуск не возможно');

end;

procedure TfStud1.Button8Click(Sender: TObject);

begin

//Восстановление из академки

if dm.ADOtStudents.FieldByName('status').Value=3 then

begin

dm.ADOtGroups.Filtered:=false;

if dm.ADOtGroups.Locate('Group',edRestoreGroup.Text,[]) then

begin

if edRestoreReason.Text<>'' then

begin

dm.ADOtStudents.Edit;

dm.ADOtStudents.FieldByName('Group').Value:=dm.ADOtGroups.FieldByName('id').Value;

dm.ADOtStudents.FieldByName('status').Value:=2;

dm.ADOtStudents.FieldByName('StatusChangeReason').Value:=edRestoreReason.Text;

dm.ADOtStudents.Post;

edRestoreReason.Text:='';

edRestoreGroup.Text:='';

ShowMessage('Студент восстановлен');

end

else ShowMessage('Укажите основание');

end

else ShowMessage('Такой группы не существует');

end

else ShowMessage('Студент не находится в академическом отпуске');

end;

procedure TfStud1.btnAdrSchoolClick(Sender: TObject);

var adr:string;

begin

fAdres.edCountry.Text:=edsCountry;

fAdres.edRegion.Text:=edsRegion;

fAdres.edDistrict.Text:=edsDistrict;

fAdres.edLocalityType.Text:=edsLocalityType;

fAdres.edLocality.Text:=edsLocality;

fAdres.idCountry:=idsCountry;

fAdres.idRegion:=idsRegion;

fAdres.idDistrict:=idsDistrict;

fAdres.idLocalityType:=idsLocalityType;

fAdres.idLocality:=idsLocality;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

fAdres.school:=true;

fAdres.Panel2.Visible:=false;

fAdres.admin:=fStudMain.admin;

fAdres.pass:=fStudMain.pass;

fAdres.ShowModal;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

edsCountry:=fAdres.edCountry.Text;

edsRegion:=fAdres.edRegion.Text;

edsDistrict:=fAdres.edDistrict.Text;

edsLocalityType:=fAdres.edLocalityType.Text;

edsLocality:=fAdres.edLocality.Text;

idsCountry:=fAdres.idCountry;

idsRegion:=fAdres.idRegion;

idsDistrict:=fAdres.idDistrict;

idsLocalityType:=fAdres.idLocalityType;

idsLocality:=fAdres.idLocality;

AdrS;

end;

procedure TfStud1.cbPasportChange(Sender: TObject);

var passp:boolean;

begin

case cbPasport.ItemIndex of

0:passp:=true;

1:passp:=false;

end;

//иной документ

mInoi.Visible:=not passp;

//Паспорт гражданина РФ

lSer.Visible:=passp;

lNum.Visible:=passp;

lVidanDate.Visible:=passp;

lKod.Visible:=passp;

lVidanKem.Visible:=passp;

edSer.Visible:=passp;

edNum.Visible:=passp;

dtpVidanDate.Visible:=passp;

edKod.Visible:=passp;

edVidanKem.Visible:=passp;

end;

procedure TfStud1.FormActivate(Sender: TObject);

begin

//Загрузка начальных данных

edNewGroupReason.Text:='';

edNewGroup.Text:='';

edAkademReason.Text:='';

edRestoreReason.Text:='';

edRestoreGroup.Text:='';

edPriceTypeReason.Text:='';

if idstud=-1 then

begin

ShowMessage('Студент не найден');Close;

{

pageControl1.ActivePageIndex:=0;

idCountry:=-1;idRegion:=-1;idDistrict:=-1;idLocalityType:=-1;idLocality:=-1;idStreet:=-1;

edCountry:='';edRegion:='';edDistrict:='';edLocalityType:='';edLocality:='';edStreet:='';

edHouse:='';edHousePart:='';edBuilding:='';edApartment:='';

idfCountry:=-1;idfRegion:=-1;idfDistrict:=-1;idfLocalityType:=-1;idfLocality:=-1;

idfStreet:=-1;edfCountry:='';edfRegion:='';edfDistrict:='';edfLocalityType:='';

edfLocality:='';edfStreet:='';edfHouse:='';edfHousePart:='';edfBuilding:='';edfApartment:='';

idsCountry:=-1;idsRegion:=-1;idsDistrict:=-1;idsLocalityType:=-1;idsLocality:=-1;edsCountry:='';

edsRegion:='';edsDistrict:='';edsLocalityType:='';edsLocality:='';idSchool:=-1;edSchool:='';

mAdrProp.Lines.Clear;mAdrFact.Lines.Clear;mAdrSchool.Lines.Clear;mInoi.Lines.Clear;

edF\_i.Text:='';edI\_i.Text:='';edO\_i.Text:='';

edF\_r.Text:='';edI\_r.Text:='';edO\_r.Text:='';

edF\_d.Text:='';edI\_d.Text:='';edO\_d.Text:='';

edPhone.Text:='';edBPlace.Text:='';cbPasport.ItemIndex:=0;lSer.Visible:=true;lNum.Visible:=true;

lVidanDate.Visible:=true;lKod.Visible:=true;lVidanKem.Visible:=true;edSer.Visible:=true;

edNum.Visible:=true;dtpVidanDate.Visible:=true;edKod.Visible:=true;edVidanKem.Visible:=true;mInoi.Visible:=false;

edSer.Text:='';edNum.Text:='';edKod.Text:='';edVidanKem.Text:='';

edSnils.Text:='';edINN.Text:='';edPOMS.Text:='';edScore.Text:='';lScore.Visible:=false;edScore.Visible:=false;

cbGender.ItemIndex:=0;cbPlata.ItemIndex:=0;dtpBDate.Date:=Date();cbPasport.ItemIndex:=0;dtpVidanDate.Date:=Date();

}

end

else

begin

edF\_i.Text:=dm.ADOtStudentsF.Value;

edI\_i.Text:=dm.ADOtStudentsI.Value;

edO\_i.Text:=dm.ADOtStudentsO.Value;

edF\_r.Text:=dm.ADOtStudentsF\_rp.Value;

edI\_r.Text:=dm.ADOtStudentsI\_rp.Value;

edO\_r.Text:=dm.ADOtStudentsO\_rp.Value;

edF\_d.Text:=dm.ADOtStudentsF\_dp.Value;

edI\_d.Text:=dm.ADOtStudentsI\_dp.Value;

edO\_d.Text:=dm.ADOtStudentsO\_dp.Value;

edPhone.Text:=dm.ADOtStudentsPhone.Value;

cbGender.ItemIndex:=-1;

while cbGender.ItemIndex<cbGender.Items.Count-1 do

begin

cbGender.ItemIndex:=cbGender.ItemIndex+1;

if cbGender.Text=dm.ADOtStudentsGender.AsString then Break;

end;

dtpbDate.Date:=dm.ADOtStudentsBDate.Value;

edBPlace.Text:=dm.ADOtStudentsBPlace.Value;

cbPasport.ItemIndex:=-1;

while cbPasport.ItemIndex<cbPasport.Items.Count-1 do

begin

cbPasport.ItemIndex:=cbPasport.ItemIndex+1;

if cbPasport.Text=dm.ADOtStudentsPassport\_Type.AsString then Break;

end;

case cbPasport.ItemIndex of

0:begin //Паспорт гражданина РФ

lSer.Visible:=true;

lNum.Visible:=true;

lVidanDate.Visible:=true;

lKod.Visible:=true;

lVidanKem.Visible:=true;

edSer.Visible:=true;

edNum.Visible:=true;

dtpVidanDate.Visible:=true;

edKod.Visible:=true;

edVidanKem.Visible:=true;

mInoi.Visible:=false;

end;

1:begin //иной документ

mInoi.Visible:=true;

lSer.Visible:=false;

lNum.Visible:=false;

lVidanDate.Visible:=false;

lKod.Visible:=false;

lVidanKem.Visible:=false;

edSer.Visible:=false;

edNum.Visible:=false;

dtpVidanDate.Visible:=false;

edKod.Visible:=false;

edVidanKem.Visible:=false;

end;

end;

edSer.Text:=dm.ADOtStudentsPassport\_Ser.Value;

edNum.Text:=dm.ADOtStudentsPassport\_Num.Value;

edVidanKem.Text:=dm.ADOtStudentsPassport\_Vidan.Value;

dtpVidanDate.Date:=dm.ADOtStudentsPassport\_Vidan\_Data.Value;

edKod.Text:=dm.ADOtStudentsPassport\_Kodp.Value;

mInoi.Text:=dm.ADOtStudentsPassport\_Other.Value;

edSnils.Text:=dm.ADOtStudentsSnils.Value;

edINN.Text:=dm.ADOtStudentsINN.Value;

edPOMS.Text:=dm.ADOtStudentsPOMS.Value;

if dm.ADOtStudentsAdrPCountry.Value<>NULL then idCountry:=dm.ADOtStudentsAdrPCountry.Value

else idCountry:=-1;

if dm.ADOtStudentsAdrPRegion.Value<>NULL then idRegion:=dm.ADOtStudentsAdrPRegion.Value

else idRegion:=-1;

if dm.ADOtStudentsAdrPDistrict.Value<>NULL then idDistrict:=dm.ADOtStudentsAdrPDistrict.Value

else idDistrict:=-1;

if dm.ADOtStudentsAdrPLocalityType.Value<>NULL then idLocalityType:=dm.ADOtStudentsAdrPLocalityType.Value

else idLocalityType:=-1;

if dm.ADOtStudentsAdrPLocality.Value<>NULL then idLocality:=dm.ADOtStudentsAdrPLocality.Value

else idLocality:=-1;

if dm.ADOtStudentsAdrPStreet.Value<>NULL then idStreet:=dm.ADOtStudentsAdrPStreet.Value

else idStreet:=-1;

edHouse:=dm.ADOtStudentsAdrPHouse.Value;

edHousePart:=dm.ADOtStudentsAdrPHousePart.Value;

edBuilding:=dm.ADOtStudentsAdrPBuilding.Value;

edApartment:=dm.ADOtStudentsAdrPApartment.Value;

if dm.ADOtStudentsAdrFCountry.Value<>NULL then idfCountry:=dm.ADOtStudentsAdrFCountry.Value

else idfCountry:=-1;

if dm.ADOtStudentsAdrFRegion.Value<>NULL then idfRegion:=dm.ADOtStudentsAdrFRegion.Value

else idfRegion:=-1;

if dm.ADOtStudentsAdrFDistrict.Value<>NULL then idfDistrict:=dm.ADOtStudentsAdrFDistrict.Value

else idfDistrict:=-1;

if dm.ADOtStudentsAdrFLocalityType.Value<>NULL then idfLocalityType:=dm.ADOtStudentsAdrFLocalityType.Value

else idfLocalityType:=-1;

if dm.ADOtStudentsAdrFLocality.Value<>NULL then idfLocality:=dm.ADOtStudentsAdrFLocality.Value

else idfLocality:=-1;

if dm.ADOtStudentsAdrFStreet.Value<>NULL then idfStreet:=dm.ADOtStudentsAdrFStreet.Value

else idfStreet:=-1;

edfHouse:=dm.ADOtStudentsAdrFHouse.Value;

edfHousePart:=dm.ADOtStudentsAdrFHousePart.Value;

edfBuilding:=dm.ADOtStudentsAdrFBuilding.Value;

edfApartment:=dm.ADOtStudentsAdrFApartment.Value;

if dm.ADOtStudentseduSchool.Value<>NULL then idSchool:=dm.ADOtStudentseduSchool.Value

else idSchool:=-1;

if dm.ADOtStudentsAdrSCountry.Value<>NULL then idsCountry:=dm.ADOtStudentsAdrSCountry.Value

else idsCountry:=-1;

if dm.ADOtStudentsAdrSRegion.Value<>NULL then idsRegion:=dm.ADOtStudentsAdrSRegion.Value

else idsRegion:=-1;

if dm.ADOtStudentsAdrSDistrict.Value<>NULL then idsDistrict:=dm.ADOtStudentsAdrSDistrict.Value

else idsDistrict:=-1;

if dm.ADOtStudentSAdrPLocalityType.Value<>NULL then idsLocalityType:=dm.ADOtStudentSAdrPLocalityType.Value

else idsLocalityType:=-1;

if dm.ADOtStudentsAdrSLocality.Value<>NULL then idsLocality:=dm.ADOtStudentsAdrSLocality.Value

else idsLocality:=-1;

dm.ADOadrElement.Active:=false;

dm.ADOadrElement.TableName:='School';

//dm.ADOadrElement.IndexFieldNames:='nazv';

dm.ADOadrElement.Active:=true;

if dm.ADOadrElement.Locate('id',idSchool,[]) then edSchool:=dm.ADOadrElementNazv.Value

else edSchool:='';

edSchoolName.Text:=edSchool;

dm.ADOadrElement.Active:=false;

dm.ADOadrElement.TableName:='adrCountry';

//dm.ADOadrElement.IndexFieldNames:='nazv';

dm.ADOadrElement.Active:=true;

if dm.ADOadrElement.Locate('id',idCountry,[]) then edCountry:=dm.ADOadrElementNazv.Value

else edCountry:='';

if dm.ADOadrElement.Locate('id',idfCountry,[]) then edfCountry:=dm.ADOadrElementNazv.Value

else edfCountry:='';

if dm.ADOadrElement.Locate('id',idsCountry,[]) then edsCountry:=dm.ADOadrElementNazv.Value

else edsCountry:='';

dm.ADOadrElement.Active:=false;

dm.ADOadrElement.TableName:='adrRegion';

//dm.ADOadrElement.IndexFieldNames:='nazv';

dm.ADOadrElement.Active:=true;

if dm.ADOadrElement.Locate('id',idRegion,[]) then edRegion:=dm.ADOadrElementNazv.Value

else edRegion:='';

if dm.ADOadrElement.Locate('id',idfRegion,[]) then edfRegion:=dm.ADOadrElementNazv.Value

else edfRegion:='';

if dm.ADOadrElement.Locate('id',idsRegion,[]) then edsRegion:=dm.ADOadrElementNazv.Value

else edsRegion:='';

dm.ADOadrElement.Active:=false;

dm.ADOadrElement.TableName:='adrDistrict';

//dm.ADOadrElement.IndexFieldNames:='nazv';

dm.ADOadrElement.Active:=true;

if dm.ADOadrElement.Locate('id',idDistrict,[]) then edDistrict:=dm.ADOadrElementNazv.Value

else edDistrict:='';

if dm.ADOadrElement.Locate('id',idfDistrict,[]) then edfDistrict:=dm.ADOadrElementNazv.Value

else edfDistrict:='';

if dm.ADOadrElement.Locate('id',idsDistrict,[]) then edsDistrict:=dm.ADOadrElementNazv.Value

else edsDistrict:='';

dm.ADOadrElement.Active:=false;

dm.ADOadrElement.TableName:='adrLocalityType';

//dm.ADOadrElement.IndexFieldNames:='nazv';

dm.ADOadrElement.Active:=true;

if dm.ADOadrElement.Locate('id',idLocalityType,[]) then edLocalityType:=dm.ADOadrElementNazv.Value

else edLocalityType:='';

if dm.ADOadrElement.Locate('id',idfLocalityType,[]) then edfLocalityType:=dm.ADOadrElementNazv.Value

else edfLocalityType:='';

if dm.ADOadrElement.Locate('id',idsLocalityType,[]) then edsLocalityType:=dm.ADOadrElementNazv.Value

else edsLocalityType:='';

dm.ADOadrElement.Active:=false;

dm.ADOadrElement.TableName:='adrLocality';

//dm.ADOadrElement.IndexFieldNames:='nazv';

dm.ADOadrElement.Active:=true;

if dm.ADOadrElement.Locate('id',idLocality,[]) then edLocality:=dm.ADOadrElementNazv.Value

else edLocality:='';

if dm.ADOadrElement.Locate('id',idfLocality,[]) then edfLocality:=dm.ADOadrElementNazv.Value

else edfLocality:='';

if dm.ADOadrElement.Locate('id',idsLocality,[]) then edsLocality:=dm.ADOadrElementNazv.Value

else edsLocality:='';

dm.ADOadrElement.Active:=false;

dm.ADOadrElement.TableName:='adrStreet';

//dm.ADOadrElement.IndexFieldNames:='nazv';

dm.ADOadrElement.Active:=true;

if dm.ADOadrElement.Locate('id',idStreet,[]) then edStreet:=dm.ADOadrElementNazv.Value

else edStreet:='';

if dm.ADOadrElement.Locate('id',idfStreet,[]) then edfStreet:=dm.ADOadrElementNazv.Value

else edfStreet:='';

edScore.Text:=dm.ADOtStudentseduScore.AsString;

AdrP;AdrF;AdrS; pageControl1.ActivePageIndex:=0;

end;

end;

end.

unit ustud2;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.Mask, Vcl.ComCtrls,

Vcl.DBCtrls, Data.DB;

type

TfStud2 = class(TForm)

PageControl1: TPageControl;

TabSheet1: TTabSheet;

TabSheet2: TTabSheet;

edPhone1: TMaskEdit;

Label1: TLabel;

Label3: TLabel;

cb1: TComboBox;

Label4: TLabel;

cb2: TComboBox;

Label5: TLabel;

Label6: TLabel;

Label7: TLabel;

edO1: TEdit;

edI1: TEdit;

edF1: TEdit;

edWork1: TEdit;

Label11: TLabel;

Label12: TLabel;

edD1: TEdit;

Label8: TLabel;

Label13: TLabel;

Label14: TLabel;

edO2: TEdit;

edI2: TEdit;

edF2: TEdit;

Label16: TLabel;

Label17: TLabel;

edWork2: TEdit;

Label18: TLabel;

edD2: TEdit;

Label19: TLabel;

edPhone2: TMaskEdit;

TabSheet3: TTabSheet;

chbDeti: TCheckBox;

chbOpeka: TCheckBox;

Label21: TLabel;

edOpekaFIO: TEdit;

chbSirota: TCheckBox;

chbBezRoditel: TCheckBox;

chbInvalid: TCheckBox;

Label22: TLabel;

edInvalidDiagnoz: TEdit;

Memo1: TMemo;

Label23: TLabel;

DBMemo1: TDBMemo;

Button4: TButton;

Label27: TLabel;

Label28: TLabel;

Label29: TLabel;

Label30: TLabel;

procedure chbOpekaClick(Sender: TObject);

procedure chbInvalidClick(Sender: TObject);

procedure Button4Click(Sender: TObject);

procedure FormActivate(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

fStud2: TfStud2;

implementation

{$R \*.dfm}

uses udm,ustud1;

procedure TfStud2.chbOpekaClick(Sender: TObject);

begin

edOpekaFIO.Enabled:=chbOpeka.Checked;

end;

procedure TfStud2.FormActivate(Sender: TObject);

begin

PageControl1.ActivePageIndex:=0;

cb1.ItemIndex:=-1;

while cb1.ItemIndex<cb1.Items.Count-1 do

begin

cb1.ItemIndex:=cb1.ItemIndex+1;

if cb1.Text=dm.ADOtStudentsR1Type.AsString then Break;

end;

edF1.Text:=dm.ADOtStudentsR1F.Value;

edI1.Text:=dm.ADOtStudentsR1I.Value;

edO1.Text:=dm.ADOtStudentsR1O.Value;

edWork1.Text:=dm.ADOtStudentsR1Work.Value;

edD1.Text:=dm.ADOtStudentsR1D.Value;

edPhone1.Text:=dm.ADOtStudentsR1Phone.Value;

cb2.ItemIndex:=-1;

while cb2.ItemIndex<cb2.Items.Count-1 do

begin

cb2.ItemIndex:=cb2.ItemIndex+1;

if cb2.Text=dm.ADOtStudentsR2Type.AsString then Break;

end;

edF2.Text:=dm.ADOtStudentsR2F.Value;

edI2.Text:=dm.ADOtStudentsR2I.Value;

edO2.Text:=dm.ADOtStudentsR2O.Value;

edWork2.Text:=dm.ADOtStudentsR2Work.Value;

edD2.Text:=dm.ADOtStudentsR2D.Value;

edPhone2.Text:=dm.ADOtStudentsR2Phone.Value;

chbDeti.Checked:=dm.ADOtStudentsDeti.Value;

chbOpeka.Checked:=dm.ADOtStudentsOpeka.Value;

edOpekaFIO.Text:=dm.ADOtStudentsOpekaFIO.Value;

chbSirota.Checked:=dm.ADOtStudentsSirota.Value;

chbBezRoditel.Checked:=dm.ADOtStudentsBezRoditel.Value;

chbInvalid.Checked:=dm.ADOtStudentsInvalid.Value;

edInvalidDiagnoz.Text:=dm.ADOtStudentsInvalidDiagnoz.Value;

//Memo1.Lines.Append(dm.ADOtStudentsDopSvedenia.Value);

end;

procedure TfStud2.Button4Click(Sender: TObject);

var ok:boolean;

begin

ok:=true;

if Cb1.ItemIndex<>5 then

begin

if edF1.Text='' then ok:=false;

if edI1.Text='' then ok:=false;

if edWork1.Text='' then ok:=false;

if edD1.Text='' then ok:=false;

if edPhone1.Text='' then ok:=false;

end;

if Cb2.ItemIndex<>5 then

begin

if edF2.Text='' then ok:=false;

if edI2.Text='' then ok:=false;

if edWork2.Text='' then ok:=false;

if edD2.Text='' then ok:=false;

if edPhone2.Text='' then ok:=false;

end;

if not ok then ShowMessage('Заполните все обязательные поля, отмеченные символом красной звездочки')

else

if fStud1.idStud<>-1 then

//if dm.ADOtStudents.Locate('id',fZayav.idStud,[loCaseInsensitive, loPartialKey]) then

begin

if dm.ADOtStudents.State in [dsEdit, dsInsert] then

dm.ADOtStudents.Post;

dm.ADOtStudents.Active:=false;

dm.ADOtStudents.Active:=true;

dm.ADOtStudents.Locate('id',fStud1.idStud,[]);

dm.ADOtStudents.Edit;

dm.ADOtStudentsR1Type.Value:=cb1.Text;

dm.ADOtStudentsR1F.Value:=edF1.Text;

dm.ADOtStudentsR1I.Value:=edI1.Text;

dm.ADOtStudentsR1O.Value:=edO1.Text;

dm.ADOtStudentsR1Work.Value:=edWork1.Text;

dm.ADOtStudentsR1D.Value:=edD1.Text;

dm.ADOtStudentsR1Phone.Value:=edPhone1.Text;

dm.ADOtStudentsR2Type.Value:=cb2.Text;

dm.ADOtStudentsR2F.Value:=edF2.Text;

dm.ADOtStudentsR2I.Value:=edI2.Text;

dm.ADOtStudentsR2O.Value:=edO2.Text;

dm.ADOtStudentsR2Work.Value:=edWork2.Text;

dm.ADOtStudentsR2D.Value:=edD2.Text;

dm.ADOtStudentsR2Phone.Value:=edPhone2.Text;

dm.ADOtStudentsDeti.Value:=chbDeti.Checked;

dm.ADOtStudentsOpeka.Value:=chbOpeka.Checked;

dm.ADOtStudentsOpekaFIO.Value:=edOpekaFIO.Text;

dm.ADOtStudentsSirota.Value:=chbSirota.Checked;

dm.ADOtStudentsBezRoditel.Value:=chbBezRoditel.Checked;

dm.ADOtStudentsInvalid.Value:=chbInvalid.Checked;

dm.ADOtStudentsInvalidDiagnoz.Value:=edInvalidDiagnoz.Text;

//dm.ADOtStudentsDopSvedenia.Value:=Memo1.Lines;

//DBMemo1.DataSource:=dm.dsStudents;

//DBMemo1.DataField:='DopSvedenia';

//DBMemo1.Lines:=Memo1.Lines;

dm.ADOtStudents.Post;

Close;

end;

end;

procedure TfStud2.chbInvalidClick(Sender: TObject);

begin

edInvalidDiagnoz.Enabled:=chbInvalid.Checked;

end;

end.

unit ustudmain;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.Menus, Vcl.StdCtrls, Vcl.ExtCtrls,

Data.DB, Vcl.Grids, Vcl.DBGrids, ComObj;

type

TfStudMain = class(TForm)

MainMenu1: TMainMenu;

N1: TMenuItem;

N2: TMenuItem;

N3: TMenuItem;

N4: TMenuItem;

N5: TMenuItem;

DBGrid1: TDBGrid;

Button1: TButton;

DBGrid2: TDBGrid;

N6: TMenuItem;

procedure FormActivate(Sender: TObject);

procedure N4Click(Sender: TObject);

procedure N2Click(Sender: TObject);

procedure N3Click(Sender: TObject);

procedure Button1Click(Sender: TObject);

procedure Button2Click(Sender: TObject);

procedure N5Click(Sender: TObject);

procedure DBGrid2CellClick(Column: TColumn);

procedure N6Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

exepath:string;

cfgpath:string;

dbpath:string;

admin:boolean;

pass:string;

end;

var

fStudMain: TfStudMain;

implementation

{$R \*.dfm}

uses udm, ulevel, uspec, ugroup, ustudpriem, ustud1, ustud2;

procedure TfStudMain.Button1Click(Sender: TObject);

begin

//Абитуриенты

fStud1.idStud:=dm.ADOtStudents.FieldByName('id').Value;

fStud1.ShowModal;

fStud2.ShowModal;

end;

procedure TfStudMain.Button2Click(Sender: TObject);

begin

//Студенты

end;

procedure TfStudMain.DBGrid2CellClick(Column: TColumn);

begin

dm.ADOtStudents.Filter:='group='+dm.ADOtGroups.FieldByName('id').AsString;

dm.ADOtStudents.Filtered:=true;

end;

procedure TfStudMain.FormActivate(Sender: TObject);

var f:textfile;

s:string;

begin

admin:=true;

pass:='911';

exepath:=ExtractFilePath(Application.ExeName);

cfgpath:=exepath+'config.cfg';

if FileExists(cfgpath) then

begin

//ShowMessage('OK');

assignfile(f,cfgpath);

reset(f);

repeat

readln(f,s);

until copy(s,1,2)<>'/\*';

//ShowMessage(s);

dbpath:=copy(s,length('DBFILENAME=')+1,255);

//ShowMessage(dbpath);

closefile(f);

if FileExists(dbpath) then

begin

//ShowMessage('DB OK');

if not dm.ADOCon1.Connected then

begin

dm.ADOCon1.ConnectionString:='Provider=Microsoft.Jet.OLEDB.4.0;Jet OLEDB:Database Password="090203";Data Source='+dbpath+';Mode=Share Deny None;Persist Security Info=False';

dm.ADOCon1.Connected:=true;

dm.ADOtLevel.Active:=true;

dm.ADOtSpec.Active:=true;

dm.ADOtGroups.Active:=true;

dm.ADOtStudents.Active:=true;

dm.tStudPriem.Active:=true;

//dm.ADOtTypes.Active:=true;

//dm.ADOTable1.Active:=true;

dm.ADOtStudents.Filter:='status>1';

dm.ADOtStudents.Filtered:=true;

dm.ADOtGroups.Sort:='KURS, spec, group';

dm.ADOtStudents.Sort:='f,i, o';

dm.ADOtStudents.Filter:='group='+dm.ADOtGroups.FieldByName('id').AsString;

dm.ADOtStudents.Filtered:=true;

end;

end

else

begin

//ShowMessage('DB NOT OK');

ShowMessage('Не обнаружена база данных. Программа будет закрыта.'+#13+'Обратитесь к системному администратору');

Close;

end;

end

else

begin

ShowMessage('Файл конфигурации не найден. Программа будет закрыта'+#13+'Обратитесь к системному администратору');

Close;

end;

end;

procedure TfStudMain.N2Click(Sender: TObject);

begin

//Специальности

fSpec.ShowModal;

end;

procedure TfStudMain.N3Click(Sender: TObject);

begin

//Группы

fGroup.ShowModal;

end;

procedure TfStudMain.N4Click(Sender: TObject);

begin

//Уровни образования

fLevel.DBGrid1.Columns[2].Visible:=false;

fLevel.DBGrid1.Columns[3].Visible:=false;

fLevel.Width:=600;

fLevel.ShowModal;

end;

procedure TfStudMain.N5Click(Sender: TObject);

begin

//Приказ на зачисление

fPriem.ShowModal;

end;

procedure TfStudMain.N6Click(Sender: TObject);

var xl,c1,c2:OLEvariant;

sp,gr,ku:integer;

b,k,ab,ak:integer;

begin

//Отчет по контингенту

xl:=CreateOLEObject('Excel.Application');

xl.Workbooks.Open(exepath+'Студенты.xlsx');//Создаем файл

c1:=xl.ActiveWorkBook.WorkSheets[1];

dm.ADOtSpec.Filter:='level=1';

dm.ADOtSpec.Filtered:=true;

dm.ADOtSpec.First;sp:=7;

while not dm.ADOtSpec.Eof do

begin

c1.Cells[3,sp]:=dm.ADOtSpec.FieldByName('numspec').AsString+' '+dm.ADOtSpec.FieldByName('namespec').AsString;

for ku:= 1 to 4 do

begin

dm.ADOtGroups.Filter:='(Spec='+dm.ADOtSpec.FieldByName('id').AsString+') and (kurs='+IntToStr(ku)+')';

dm.ADOtGroups.Filtered:=true;

if dm.ADOtGroups.RecordCount>0 then

begin

dm.ADOtGroups.First;gr:=1;

while not dm.ADOtGroups.Eof do

begin

dm.ADOtStudents.Filter:='group='+dm.ADOtGroups.FieldByName('id').AsString;

dm.ADOtStudents.Filtered:=true;

dm.ADOtStudents.First;b:=0;k:=0;ab:=0;ak:=0;

while not dm.ADOtStudents.Eof do

begin

if dm.ADOtStudents.FieldByName('status').Value=2 then

begin

if dm.ADOtStudents.FieldByName('FinalPriceType').AsString='бюджетное место' then b:=b+1

else k:=k+1;

end;

if dm.ADOtStudents.FieldByName('status').Value=3 then

begin

if dm.ADOtStudents.FieldByName('FinalPriceType').AsString='бюджетное место' then ab:=ab+1

else ak:=ak+1;

end;

dm.ADOtStudents.Next;

end;

c1.Cells[6+(ku-1)\*4+gr,sp]:=dm.ADOtGroups.FieldByName('group').AsString;

c1.Cells[6+(ku-1)\*4+gr,sp+2]:=IntToStr(b);

c1.Cells[6+(ku-1)\*4+gr,sp+3]:=IntToStr(k);

c1.Cells[6+(ku-1)\*4+gr,sp+4]:=IntToStr(ab);

c1.Cells[6+(ku-1)\*4+gr,sp+5]:=IntToStr(ak);

dm.ADOtGroups.Next;gr:=gr+1;

end;

end;

end;

dm.ADOtSpec.Next;sp:=sp+6;

end;

//Скрываем неиспользуемые столбцы

for ku:=sp to 114 do c1.Columns[ku].EntireColumn.Hidden:=True;

//Скрываем неиспользуемые столбцы

dm.ADOtSpec.Filter:='level=2';

dm.ADOtSpec.Filtered:=true;

dm.ADOtSpec.First;sp:=120;

while not dm.ADOtSpec.Eof do

begin

c1.Cells[3,sp]:=dm.ADOtSpec.FieldByName('numspec').AsString+' '+dm.ADOtSpec.FieldByName('namespec').AsString;

for ku:= 1 to 4 do

begin

dm.ADOtGroups.Filter:='(Spec='+dm.ADOtSpec.FieldByName('id').AsString+') and (kurs='+IntToStr(ku)+')';

dm.ADOtGroups.Filtered:=true;

if dm.ADOtGroups.RecordCount>0 then

begin

dm.ADOtGroups.First;gr:=1;

while not dm.ADOtGroups.Eof do

begin

dm.ADOtStudents.Filter:='group='+dm.ADOtGroups.FieldByName('id').AsString;

dm.ADOtStudents.Filtered:=true;

dm.ADOtStudents.First;b:=0;k:=0;ab:=0;ak:=0;

while not dm.ADOtStudents.Eof do

begin

if dm.ADOtStudents.FieldByName('status').Value=2 then

begin

if dm.ADOtStudents.FieldByName('FinalPriceType').AsString='бюджетное место' then b:=b+1

else k:=k+1;

end;

if dm.ADOtStudents.FieldByName('status').Value=3 then

begin

if dm.ADOtStudents.FieldByName('FinalPriceType').AsString='бюджетное место' then ab:=ab+1

else ak:=ak+1;

end;

dm.ADOtStudents.Next;

end;

c1.Cells[6+(ku-1)\*4+gr,sp]:=dm.ADOtGroups.FieldByName('group').AsString;

c1.Cells[6+(ku-1)\*4+gr,sp+2]:=IntToStr(b);

c1.Cells[6+(ku-1)\*4+gr,sp+3]:=IntToStr(k);

c1.Cells[6+(ku-1)\*4+gr,sp+4]:=IntToStr(ab);

c1.Cells[6+(ku-1)\*4+gr,sp+5]:=IntToStr(ak);

dm.ADOtGroups.Next;gr:=gr+1;

end;

end;

end;

dm.ADOtSpec.Next;sp:=sp+6;

end;

//Скрываем неиспользуемые столбцы

for ku:=sp to 155 do c1.Columns[ku].EntireColumn.Hidden:=True;

//Скрываем неиспользуемые столбцы

dm.ADOtSpec.Filtered:=false;

dm.ADOtGroups.Filtered:=false;

dm.ADOtStudents.Filter:='group='+dm.ADOtGroups.FieldByName('id').AsString;

dm.ADOtStudents.Filtered:=true;

xl.Visible:=true;

end;

end.

**Приложение Б**

Разработка программного модуля студент колледжа

Спецификация

460.ЗМК.00003-01 01

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Обозначение** | | |  | | --- | | **Наименование** | | |  | | --- | | **Примечание** | |
| 460.ЗМК.00003-01 12 01 | Разработка программного |  |
|  | модуля студент колледжа |  |
|  | Текст программы |  |
|  |  |  |
|  |  |  |
| 460.ЗМК.00003-01 13 01 | Разработка программного |  |
|  | модуля студент колледжа |  |
|  | Описание программы |  |
|  |  |  |
|  |  |  |
| 460.ЗМК.00003-01 34 01 | Разработка программного |  |
|  | модуля студент колледжа |  |
|  | Руководство |  |
|  | пользователя |  |
|  |  |  |
|  |  |  |
| 460.ЗМК.00003-01 81 01 | Разработка программного |  |
|  | модуля студент колледжа |  |
|  | Пояснительная записка |  |
|  |  |  |
|  |  |  |

**Приложение B**

**Диск с программным средством**