Содержание

Введение

1. Постановка задачи

1.1 Организационно-экономическая сущность задачи

1.2 Структура входных и выходных данных

2. Вычислительная система

2.1 Используемые технические средства

2.2 Инструменты разработки

3. Проектирование задачи

3.1 Описание модели данных

3.2 Объектно-ориентированный анализ

3.3 Концептуальный прототип

4 Описание программы

4.1 Концептуальный прототип

4.2 Организация данных

4.3 Реализация функций

5. Тестирование

5.1 Тесты на использование

5.2 Функциональное тестирование

5.3 Полное тестирование

6. Описание применения

6.1 Требования к программному и аппаратному обеспечению компьютера

6.2 Требования к приложению

6.3 Функциональное назначение и область применения программы

6.4 Руководство пользователя

Заключение

Приложение

Введение

В современном мире огромную роль играют информационные технологии и сегодня уже не вызывает сомнения, что компьютеры будут играть важную роль в будущей цивилизации человечества. Уже сейчас их внедрение приводит к коренному изменению технологии во многих отраслях современного производства. И от того, в какой степени и как будут решены проблемы компьютеризации обучения детей и молодежи теперь, существенно зависит подготовленность подрастающего поколения к жизни в будущем обществе.

Компьютер быстро вошел в нашу жизнь. Еще несколько лет назад было редкостью увидеть какой-нибудь персональный компьютер – они были, но были очень дорогие, и даже не каждая фирма могла иметь у себя в офисе компьютер. Теперь в каждом третьем доме есть компьютер, который уже глубоко вошел в жизнь человека.

Современные вычислительные машины представляют одно из самых значительных достижений человеческой мысли, влияние, которого на развитие научно-технического прогресса трудно переоценить. Области применения ЭВМ непрерывно расширяются.

С помощью компьютерных систем осуществляется введение документации, обеспечивается электронная почта и связь с банками данных. Сети ЭВМ связывают разных пользователей, расположенных в одном учреждении или находящихся в различных регионах страны.

В данное время существует глобальная проблема за контролем документооборота учета сотрудников в какой-либо организации. Данные могут легко уничтожиться либо потеряться, что может привести к нежелательным последствиям. Эта проблема усугубляется при коллективном использовании информации о сотрудниках, когда надо найти документы, созданные другим сотрудником, и, наконец, она становится практически невыполнимой в том случае, если документы утеряны.

Огромный недостаток в бумажных носителях заключается в том, что много времени тратится на поиск необходимых для работы данных.

В такой ситуации единственно правильным решением является ведение учета и хранение информации о сотрудниках с помощью ПЭВМ в Базе данных, которой в необходимости можно легко обменяться и утеря информации будет практически невозможна.

Для более эффективного и полного учета состояния дисциплинарной практики в УВД Миноблисполкома мною было разработано ПС, очень удобное и легкое в применении и не требующего углубленных знаний компьютера.

Данная пояснительная записка является полным описанием разработанного ПС и разбита на ряд глав по содержательному признаку.

Первый раздел “Постановка задачи” исследует предметную область; определяет круг задач, которые должны быть автоматизированы или модифицированы; включает описание бизнес-процессов; определяет функциональные требования и как решается задача в настоящее время. Также первый раздел включает модель, которая отображает функциональную структуру объектов, то есть производимые ими действия и связи между этими действиями в виде диаграммы вариантов использования, диаграммы деятельности, диаграммы компонентов.

Во втором разделе “Вычислительная система” находятся данные о требованиях к аппаратному обеспечению и конфигурации, операционным ресурсам, приводится характеристика оперативной системы, выбор и обоснование среды разработки приложения.

В разделе “Проектирование задачи ” находятся сведения о требованиях к пользовательскому интерфейсу. Описывается структура пользовательского интерфейса – система меню, диалоговых окон и элементов управления. В разделе находятся сведения о логической и физической структуре данных в контексте среды разработки. Содержится информация о разработанных функциях, которые закреплены за определенными элементами управления, структура и описание функций пользователя в рамках среды разрабатываемого приложения. Содержится информация о взаимосвязях функций, о составе модулей и принципах объединения функций в модули. В конце раздела приводятся листинги всех модулей с комментариями и указанием элементов управления, которые инициируют их выполнение.

Четвертый раздел “Описание программного средства” содержит общие сведения об обозначении и именовании приложения, программное обеспечение, необходимое для функционирования программы, принципы инсталляции программного средства. Находятся сведения о классах решаемых задач и назначение программы. Также четвертый раздел включает информацию о входных и выходных данных, их характер, организацию, формат и описание.

В пятом разделе “Методика испытаний” содержаться сведения о требованиях к техническим средствам для проведения испытаний, требования к характеристикам программы применительно к условиям эксплуатации, требования к информационной и программной совместимости. Также в разделе находится информация о порядке проведения испытаний, то есть о функциональном и полном тестировании.

Шестой раздел “Применение” содержит сведения о назначении программного средства, области применения, классе решаемых задач, ограничениях, накладываемых на область применения. Содержит требования к необходимым для данного программного средства техническим и программным ресурсам. В разделе находится информация о структуре и методике использования справочной системы.

Приложения к пояснительной записке содержат текст программы и отчет журнала “Приложение”.

В графической части проекта представлены: диаграмма вариантов использования, диаграмма классов, диаграмма последовательности и диаграмма деятельности.

1 Постановка задачи

* 1. Организационно-экономическая сущность задачи

Разрабатываемый программный продукт на тему: “Автоматизация учета правонарушений в УВД Миноблисполкома”, предназначен для учета сотрудников УВД, нарушивших дисциплинарные нормы.

Для конкретной постановки задачи и формулирования требований к разрабатываемому приложению требуется исследовать предметную область проекта и цели, поставленные перед проектом. Целью проекта является контроль сотрудников и принятия мер для прекращения нарушения дисциплины.

Программный продукт должен обрабатывать и хранить информацию. Приложение должно выполнять необходимый поиск по заданному критерию и выводить найденную информации на печать.

Проанализировав предметную область определяется круг задач, которые будут автоматизированы в разрабатываемой программе. Созданный продукт должен выполнять следующие функции: ведение учета сотрудников, поиск по заданному критерию, формирование отчетов, автоматическое снятие взыскания через год после его наложения.

Разрабатываемая программа будет использоваться в УВД Миноблисполкома постоянно. Это обусловлено тем, что благодаря ей будет осуществлен контроль работников организации. Широкое применение получат функции автоматического снятия взыскания и автоматизации ввода информации.

Исходя из анализа предметной области разрабатываемое приложение должно содержать:

- БД сотрудников, получивших взыскания;

- возможность добавления, редактирование и удаления записи;

- выбор должности, звания, областей, службы и подразделения из блока;

- запросы по блокам по фамилии, области, службе, подразделению, по должностям и по приказу;

- отчеты по запросам с возможностью предварительного просмотра и выводом на печать;

- автоматическое снятие взыскания через год;

- досрочное снятие взыскания.

Описание бизнес-процессов (функций) используемых в разрабатываемом продукте представлено на диаграмме вариантов использования, которая представлена в графической части на 1 листе.

Основное приложение должно широко использовать технологию диалоговых окон для приведения к минимуму ошибок пользователя при работе с приложением. Приложение должно иметь хорошо развитую справочную систему.

При разработке программы и построения алгоритмов для решения его задач разработчик должен соблюдать следующие правила:

- структура алгоритма не должна быть слишком сложной;

- алгоритмы программы должны быть легко модифицированы;

- алгоритмы не должны влиять на не корректную работоспособность всей программы.

В настоящее время в УВД Миноблисполкома учет сотрудников ведется в программе Excel, в которой формировать запросы, отчеты, а также снимать записи с учета необходимо вручную. В связи с этим данное ПС создавалось для улучшения оперативной работы учета сотрудников, нарушивших дисциплинарные нормы, экономии времени ведения учета, а также для автоматизации действия человека, т.е. замены ручного труда на действия с помощью ПЭВМ.

В данном программном продукте планируется создавать отчеты и диаграммы по результатам дисциплины, как по УВД, так и по области в целом, вести статистику изменения показателей по сравнению с прошлым годом, все это преследует цель постоянного контроля сотрудников и принятия мер для избежания нарушения дисциплины в УВД Миноблисполкома.

* 1. Структура входных и выходных данных

При проектировании модели приложения было выявлено, что наиболее предпочтительным будет хранение информации о сотрудниках в таблицах Microsoft Office Access. Данные файлы будут типизированными, то есть строго определенного типа, состоящие из записей, которые в свою очередь состоят из полей определенного типа. Количество полей и их тип определяются на этапе проектирования исходя из требований к хранимой в файлах информации.

В приложении будут созданы две таблицы для хранения информации.

Первый таблица будет содержать сведения о сотрудниках управления и области, а вторая информацию о снятых с учета сотрудниках. Данные таблицы будут содержать следующие поля:

- ФИО;

- звание;

- область ил УВД;

- служба;

- подразделение;

- должность;

- взыскание;

- чей приказ;

- номер приказа;

- дата приказа;

- краткое описание проступка.

В приложении “DisPract” входными данными будут являться информация о сотрудниках, в соответствии с полями таблиц.

Выходными документами в данной программе являются отчёты, а также данные, отображаемые в DBGrid. По нажатию на кнопку “Отчет” или подменю “Печать” пользователь может вывести отчёт на печать или предварительно его просмотреть, пример отчета приведен на рисунке 1. В отчётах по запросам будут входить только те записи, которые удовлетворяют заданному критерию.

Для создания отчетов используется технология Rave Reports, которая является авторизованным членом Комитета открытых систем и разрабатывает программное обеспечение для Delphi. В ее основе лежит идея отделения процесса разработки от процесса создания отчета за счет использования промежуточного документа, который называется проектом отчета. Проект отчета создается с помощью среды Rave (Report Authoring Visual Environment — визуальная среда автора отчета), а в создаваемую программу внедряется так называемая машина генератора отчета, которая по данным, полученным из файла проекта, создает документ отчета. Пример отчета



Рисунок - 1

2. Вычислительная система

2.1 Используемые технические средства

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

- процессор: Pentium II 400 МГц и выше;

объем оперативной памяти: 32 Мбайт и выше;

- операционная система: Windows 98, Windows Me, Windows XP;

- свободное место на жестком диске 20 Мбайт и выше;

- наличие монитора VGA с разрешением не менее 800x600 точек;

- наличие манипулятора “мышь”;

- CD-ROM 10х;

- клавиатура IBM PC любой модификации;

- принтер.

2.2 Инструменты разработки

Для разработки программы разработчиком были выбраны следующие инструменты разработки:

- операционная система Microsoft Windows XP Professional;

- интегрированная среда Delphi 7;

- пакет Rational Rose.

Возможности операционной системы Windows XP Professional позволяют использовать ее как основную операционную систему для современных настольных компьютеров и компьютеров Notebook, используемых на предприятиях любого типа. При создании этой системы корпорация Майкрософт сохранила все полезные возможности Windows 98 - технологию Plug and Play, простой и понятный пользовательский интерфейс, широкие возможности управления. Кроме того, их дополнили система безопасности, средства управления и обеспечения надежности, характерные для системы Windows NT. Развертывание системы Windows XP Professional как на одном компьютере, так и в рамках всемирной сети позволяет повысить эффективность использования компьютерных технологий при одновременном снижении общей стоимости владения.

Delphi – это мощная среда визуального, объектно-ориентированного программирования, позволяющая решать множество задач, в частности это комбинация нескольких важнейших технологий:

– Высокопроизводительный компилятор в машинный код;

– объектно-ориентированная модель компонент;

– визуальное (а, следовательно, и скоростное) построение приложений из программных прототипов;

– Масштабируемые средства для построения баз данных.

Компилятор, встроенный в Delphi, обеспечивает высокую производительность, необходимую для построения приложений в архитектуре “клиент-сервер”. Этот компилятор в настоящее время является самым быстрым в мире, его скорость компиляции составляет свыше 120 тысяч строк в минуту на компьютере 486DX33. Он предлагает легкость разработки и быстрое время проверки готового программного блока, характерного для языков четвертого поколения (4GL) и в то же время обеспечивает качество кода, характерного для компилятора 3GL.

Основной упор в модели Delphi делается на максимальном использовании кода. Это позволяет разработчикам строить приложения весьма быстро из заранее подготовленных объектов, а также дает им возможность создавать свои собственные объекты для среды Delphi. Никаких ограничений по типам объектов, которые могут создавать разработчики, не существует. Действительно, все в Delphi написано на нем же, поэтому разработчики имеют доступ к тем же объектам и инструментам, которые использовались для создания среды разработки. В результате нет никакой разницы между объектами, поставляемыми Borland или третьими фирмами, и объектами, которые вы можете создать.

Объекты БД в Delphi основаны на SQL и включают в себя полную мощь Borland Database Engine. В состав Delphi также включен Borland SQL Link, поэтому доступ к СУБД Oracle, Sybase, Informix и InterBase происходит с высокой эффективностью. Кроме того, Delphi включает в себя локальный сервер Interbase для того, чтобы можно было разработать расширяемые на любые внешние SQL-сервера приложения в офлайновом режиме.

Данный дипломный проект был выполнен на языке программирования Borland Delphi 7.0 на основе такой технологии как ADO. Таблицы и отчеты разрабатывались в приложении Rave Reports. Для более понятного и удобного интерфейса использовались такие компоненты как DBGrid, DBedit, Dbtext. Запросы создавались с помощью компонента ADOQuery, а доступ к набору данных через ADOTable. А отчёты на основе запросов с помощью компонентов RvQueryConnection.

CASE-средство Rational Rose было выбрано, так как оно позволяет разрабатывать проектную документацию в виде диаграмм и спецификаций. Rational Rose - CASE-средство фирмы Rational Software Corporation - предназначено для автоматизации этапов анализа и проектирования ПО, а также для генерации кодов на различных языках и выпуска проектной документации. Rational Rose использует синтез-методологию объектно-ориентированного анализа и проектирования, основанную на подходах трех ведущих специалистов в данной области: Буча, Рамбо и Джекобсона. Разработанная ими универсальная нотация для моделирования объектов (UML - Unified Modeling Language) претендует на роль стандарта в области объектно-ориентированного анализа и проектирования. Кроме того, Rational Rose содержит средства реинжиниринга программ, обеспечивающие повторное использование программных компонент в новых проектах.

3. Проектирование задачи

3.1 Описание модели данных

Теоретической основой этой модели стала теория отношений, основу которой заложили два логика — американец Чарльз Содерс Пирс (1839-1914) и немец Эрнст Шредер (1841-1902). В руководствах по теории отношений было показано, что множество отношений замкнуто относительно некоторых специальных операций, то есть образует вместе с этими операциями абстрактную алгебру. Это важнейшее свойство отношений было использовано в реляционной модели для разработки языка манипулирования данными, связанного с исходной алгеброй. Американский математик Э. Ф. Кодд в 1970 году впервые сформулировал основные понятия и ограничения реляционной модели, ограничив набор операций в ней семью основными и одной дополнительной операцией. Предложения Кодда были настолько эффективны для систем баз данных, что за эту модель он был удостоен престижной премии Тьюринга в области теоретических основ вычислительной техники.

Основной структурой данных в модели является отношение, именно поэтому модель получила название реляционной (от английского relation — отношение).

В соответствии с наиболее распространенной трактовкой, принадлежащей К. Дейту, реляционная модель данных состоит из трех частей – структурной части, целостной части и манипуляционной.

Структурная часть описывает, какие объекты рассматриваются реляционной моделью. Определяется, что единственной структурой данных, используемой в реляционной модели, являются n-арные отношения.

Целостная часть описывает ограничения специального вида, которые должны выполнять для любых отношений в любых реляционных базах данных – это целостность сущностей и целостность ссылок или внешних ключей.

Манипуляционная часть описывает два эквивалентных способа манипулирования реляционными данными – реляционную алгебру и реляционное исчисление.

Данных относятся логические, строковые и численные. Однако это жесткое требование практически не накладывает никаких ограничений, поскольку трактуется следующим образом: в реляционных операциях не должна учитываться внутренняя структура данных. Тем не менее, должны быть описаны действия, которые можно производить с определенным типом данных как с единым целым. Так для данных числового типа разрешена операция сложения, для строковых же данных аналогом сложения будет операция конкатенации или присоединения.

С понятием типа данных тесно связано понятие домена. Домен является семантическим понятием и рассматривается как подмножество значений некоторого типа данных, имеющих определенный смысл. Домен характеризуется следующими свойствами:

- домен имеет в пределах базы данных уникальное имя;

- домен определен на некотором простом типе данных или на другом домене;

- домен может иметь некоторое логическое условие, позволяющее описать подмножества данных, допустимых для данного домена;

- домен несет определенную смысловую нагрузку.

Число атрибутов в отношении называют степенью или арностью отношения.

Мощность множества кортежей отношения называют мощностью отношения.

Таким образом, заголовок отношения описывает декартово произведение доменов, на котором задано отношение. Заголовок статичен и не меняется во время работы с базой данных. Если в отношении изменены, добавлены или удалены атрибуты, то результатом этих действий будет появление нового отношения. Тело отношения представляет собой набор кортежей, то есть подмножество декартова произведения доменов. Именно тело отношения является отношением в математическом смысле. Тело отношения может изменяться во время работы с базой данных.

Реляционной базой данных называется набор отношений.

Схемой реляционной базы данных называется набор заголовков входящих в неё отношений.

3.2 Объектно-ориентированный анализ

Основным классом, использующемся в проекте, будет класс TForm. Данный класс будет использоваться для размещения на нем компонентов – элементов создаваемого приложения. Класс TForm будет включать следующие классы, предназначенные для создания интерфейса в программе: TLabel, TEdit, TMemo, TButton, TStringGrid и другие. Класс используется для обозначения множества объектов, которые обладают одинаковой структурой, поведением и отношениями с объектами из других классов.

Диаграмма классов служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. Диаграмма может отражать различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывать их внутреннюю структуру и типы отношений. На данной диаграмме не указывается информация о временных аспектах функционирования системы. Диаграмма классов представлена в графической части на листе 2.

Для моделирования процесса выполнения операций используются диаграмма деятельности. Каждое состояние на диаграмме деятельности соответствует выполнению некоторой элементарной операции, переход в следующее состояние срабатывает только при завершении этой операции. Диаграмма деятельности представлена в графической части на листе 4.

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

На главной форме имеется три вкладки:

на первой компонент DataSource1 типа TDataSource является промежуточным звеном между компонентом ADOQuery1 типа ADOQuery, соединенным с таблицей БД сотрудников Аппарата УВД, и визуальным компонентом DBGrid1, с помощью которого пользователь взаимодействует с этой таблицей; на второй компонент DataSource2 типа TDataSource является промежуточным звеном между компонентом ADOQuery2 типа ADOQuery, соединенным с таблицей БД сотрудников всех областей УВД и визуальным компонентом DBGrid2, с помощью которого пользователь взаимодействует с этой таблицей; на третьей компонент DataSource3 типа TDataSource является промежуточным звеном между компонентом ADOTable2 типа ADOTable, соединенным с таблицей БД сотрудников Аппарата УВД, и визуальным компонентом DBGrid3, с помощью которого пользователь взаимодействует с этой таблицей;

Компоненты DBGrid1, DBGrid2, DBGrid3 отображает содержимое таблицы БД в виде сетки, в которой столбцы соответствуют полям, а строки – записям таблицы. Эти компоненты связываются со своим источниками данных через свойство DataSource. Аналогично взаимодействует DBGrid1 формы поиска данных с компонентом ADOQuery3 через DataSource4.

Для компонента ADOConnection1 также с помощью свойства ConnectionString нужно установить соединение с хранилищем данных.

В таблице 3.2 приведены компоненты, используемые для работы с таблицами и запросами БД, их основные свойства и значения этих свойств.

Таблица 3.2 - Значения свойств компонентов

|  |  |  |
| --- | --- | --- |
| *Компонент* | *Свойства* | *Значения* |
| *ADOConnection1* | *ConnectionString* | *Pr.mdb* |
| *ADOQuery1* | *Connection* | *ADOConnection1* |
| *ADOQuery2* | *Connection* | *ADOConnection1* |
| *ADOQuery3* | *Connection* | *ADOConnection1* |
| *ADOTable2* | *Connection* | *ADOConnection1* |
| *ADOTable1* | *Connection* | *ADOConnection1* |
| *DataSource1* | *DataSet* | *ADOQuery1* |
| *DataSource2* | *DataSet* | *ADOQuery2* |
| *DataSource3* | *DataSet* | *ADOTable2* |
| *DataSource4* | *DataSet* | *ADOQuery3* |
| *DBGrid1 (Главная форма)* | *DataSource* | *DataSource1* |
| *DBGrid2 (Главная форма)* | *DataSource* | *DataSource2* |
| *DBGrid3 (Главная форма)* | *DataSource* | *DataSource3* |
| *DBGrid1 (Форма поиска данных)* | *DataSource* | *dsrType* |

Диаграмма компонентов представлена в графической части на листе 3.

3.3 Концептуальный прототип

Разработанный программный продукт “Учет дисциплинарной практики в УВД Миноблисполкома” представляет собой файл “DisPract.exe”.

При двойном нажатии кнопки мыши на файл запускается файл, состоящий из главного окна, изображенного на рисунке 3.1

Рисунок 3.1 – Главное окно программы

На главном окне есть три вкладки с тремя таблицами: информация о сотрудниках УВД, областных сотрудниках и сотрудниках снятых с учета. В зависимости от того, на какой вы вкладке в данный момент находитесь, с той вы будите и работать, а именно редактировать запись, удалять ее, добавлять новый, осуществлять поиски и выводить данные на печать.

Для добавления нового сотрудника нужно нажать кнопку “Добавление новой записи”, находящуюся в нижней части главного окна, после чего откроется новое окно “Добавление записи”, изображенное на рисунке 3.2.

Для редактирования сотрудника нужно нажать кнопку “Редактирование записи”, после чего на экране появится окно “Редактирование записи” аналогичное окну представленному на рисунке 3.2.

Рисунок 3.2 – Окно “Добавление записи”

Для удаления записи нажмите кнопку “Удаление записи”, после чего программа спросит о том, действительно ли вы хотите удалить запись, если вы нажмите “Ok”, то запись будет удалена.

В данном программном продукте предусмотрена такая возможность как досрочное снятие. Оно предназначено для того, чтобы раньше установленного срока (равняется одному году с даты подписания приказа) снять запись с учета, после чего она будет автоматически занесена в Базу данных сотрудников снятых с учета. Если сотрудника не снимать досрочно, он будет снят через год после того, как его поставили на учет.

Досрочное снятие сотрудника осуществляется двумя способами:

– выбрать необходимую запись на нужной вкладке и нажать “Файл-Досрочное снятие”;

– выбрать “Поиск-по фамилии”, затем ввести фамилию сотрудника, которого необходимо снять с учета и нажать кнопку “Выбрать”, после чего в новом окне нажать на кнопку “Досрочное снятие”.

Далее программа откроет новое окно, где можно ввести новую дату и номер снятия приказа (рисунок 3.3). После внесения изменений нажмите на кнопку “Записать” и внесенные изменения применятся к записи и вы можете это увидеть, перейдя на третью вкладку главного окна.

Рисунок 3.3 – Окно “Досрочное снятие”

Данный программный продукт осуществляет поиск по фамилии, по званию, по области, по службе, по подразделения, по должности и по месту приказа. Для выполнения поиска по фамилии следует выбрать “Поиск-по фамилии”, после выбора откроется окно ( рисунок 3.4), где необходимо ввести фамилию или выбрать ее из списка.

После выбора фамилии программа выдаст результата поиска в новом окне (рис. 3.5) Если ни одной записи не найдено, то программа выдаст сообщение, в котором будет сказано, что в Базе данных нет сотрудников, удовлетворяющих критерию запроса. Аналогично действуют и выше перечисленные запросы.

Рис. 3.3.4 Окно “Выбор фамилии для поиска”

Рис. 3.3.5 Окно “Выбор фамилии для поиска”

3.4 Функции: логическая и физическая организация и элементы управления

При исследовании диаграммы вариантов использования были выявлены основные функциональные требования поставленной задачи, которые нужно было реализовать в проекте. Для реализации требований, поставленных перед автором проекта, в программе был создан ряд процедур, которые обеспечивают их корректное выполнение. Процедуры, используемые в программе, не являются встроенными в язык программирования, а реализованы автором.

Многие процедуры в программе взаимосвязаны между собой. Взаимосвязь процедур используется как для возможности дальнейшей обработки уже обработанной информации другими процедурами, так и для возможности существования некоторых процедур по средствам других. Исходя из диаграммы вариантов использования, самой главной функцией в программе является ввод информации, поиск и контроль снятия взыскания.

Для реализации данной функции было создано ряд процедур, которые представлены ниже:

* процедура сортировки информации областников или аппарата управления и автоматическое снятие взыскания

procedure TForm2.FormActivate(Sender: TObject);

var k:integer;

begin

with datamodule4.ADOQuery1 do begin

 Close;

 SQL.Clear;

 SQL.Add('select \* from Pract');

 sql.Add('where Rugo="Аппарат УВД"');

 Open;

 if RecordCount <> 0 then

 begin

 datamodule4.DataSource1.DataSet :=datamodule4.adoquery1;

 end

 else

 begin

 end;

end;

with datamodule4.ADOQuery2 do begin

 Close;

 SQL.Clear;

 SQL.Add('select \* from Pract');

 sql.Add('where Rugo<>"Аппарат УВД"');

 Open;

 if RecordCount <> 0 then

 begin

 datamodule4.DataSource2.DataSet :=datamodule4.adoquery2;

 end

 else

 begin

 end;

end;

datamodule4.ADOTable1.Open;

datamodule4.ADOTable2.Open;

while not datamodule4.ADOTable1.Eof do

begin

k:=0;

if datamodule4.ADOTable1dataspr.Value<=date() then

begin

datamodule4.ADOTable2.Insert;

datamodule4.ADOTable2['Fam']:=datamodule4.ADOTable1fam.Value;

datamodule4.ADOTable2['Imia']:=datamodule4.ADOTable1imia.Value;

datamodule4.ADOTable2['otc']:=datamodule4.ADOTable1otc.Value;

datamodule4.ADOTable2['zvanie']:=datamodule4.ADOTable1zvanie.Value;

datamodule4.ADOTable2['rugo']:=datamodule4.ADOTable1rugo.Value;

datamodule4.ADOTable2['sl']:=datamodule4.ADOTable1sl.Value;

datamodule4.ADOTable2['podr']:=datamodule4.ADOTable1podr.Value;

datamodule4.ADOTable2['dolj']:=datamodule4.ADOTable1dolj.Value;

datamodule4.ADOTable2['vzisk']:=datamodule4.ADOTable1vzisk.Value;

datamodule4.ADOTable2['chpr']:=datamodule4.ADOTable1chpr.Value;

datamodule4.ADOTable2['nompr']:=datamodule4.ADOTable1nompr.Value;

datamodule4.ADOTable2['datapr']:=datamodule4.ADOTable1datapr.Value;

datamodule4.ADOTable2['opis']:=datamodule4.ADOTable1opis.Value;

datamodule4.ADOTable2['nomspr']:=datamodule4.ADOTable1nomspr.Value;

datamodule4.ADOTable2['dataspr']:=datamodule4.ADOTable1dataspr.Value;

datamodule4.ADOTable2.Post;

datamodule4.ADOTable1.Delete;

k:=1;

end;

datamodule4.ADOTable1.Next;

if k=1 then datamodule4.ADOTable1.Prior;

end;

datamodule4.ADOTable1.Close;

datamodule4.ADOTable2.Close;

datamodule4.ADOTable2.Active:=true;

end;

* процедура поиска по фамилии (аналогично выполняются и другие запросы)

procedure TForm2.N6Click(Sender: TObject);

var

fam: string[30];

begin

form6.Caption:='Поиск по фамилии';

form6.bsSkinButton2.Visible:=true;

if pagecontrol1.ActivePage=tabsheet1 then begin

fam:=InputBox('Выборка информации из БД','Укажите фамилию, по которой необходимо выполнить поиск', '');

if fam<> '' then

begin

with datamodule4.ADOQuery3 do begin

 Close;

 SQL.Clear; //очищение предыдущего запроса

 SQL.Add('select \* from Pract'); //выбирает все данные из таблицы Sotr1

 SQL.Add('where (Fam like "'+fam+ '%") and (Rugo="Аппарат УВД")'); //выбор данных из таблицы, где

 Open;

 if RecordCount <> 0 then //чтение записей пока не дойдет до последней

 begin

 datamodule4.DataSource4.DataSet:=datamodule4.adoquery3;

 end

 else

 begin

 ShowMessage('В БД сотрудников нет записей, удовлетворяющих' +

 #13 + 'критерию запроса.'); exit;

 end;

end;

end;

end;

if pagecontrol1.ActivePage=tabsheet2 then begin

fam:=InputBox('Выборка информации из БД','Укажите фамилию, по которой необходимо выполнить поиск', '');

if fam<> '' then

begin

with datamodule4.ADOQuery3 do begin

 Close;

 SQL.Clear; //очищение предыдущего запроса

 SQL.Add('select \* from Pract'); //выбирает все данные из таблицы Sotr1

 SQL.Add('where (Fam like "'+fam+ '%") and (Rugo<>"Аппарат УВД")'); //выбор данных из таблицы, где

 Open;

 if RecordCount <> 0 then //чтение записей пока не дойдет до последней

 begin

 datamodule4.DataSource4.DataSet:=datamodule4.adoquery3;

 end

 else

 begin

 ShowMessage('В БД сотрудников нет записей, удовлетворяющих' +

 #13 + 'критерию запроса.'); exit;

 end;

end;

end;

end;

if pagecontrol1.ActivePage=tabsheet3 then begin

fam:=InputBox('Выборка информации из БД','Укажите фамилию, по которой необходимо выполнить поиск', '');

if fam<> '' then

begin

with datamodule4.ADOQuery3 do begin

 Close;

 SQL.Clear; //очищение предыдущего запроса

 SQL.Add('select \* from SnVzisk'); //выбирает все данные из таблицы Sotr1

 SQL.Add('where (Fam like "'+fam+ '%")'); //выбор данных из таблицы, где

 Open;

 if RecordCount <> 0 then //чтение записей пока не дойдет до последней

 begin

 datamodule4.DataSource4.DataSet:=datamodule4.adoquery3;

 end

 else

 begin

 ShowMessage('В БД сотрудников нет записей, удовлетворяющих' +

 #13 + 'критерию запроса.'); exit;

 end;

end;

end;

end;

form6.Show;

end;

* процедура редактирования записи

procedure TForm2.bsSkinButton3Click(Sender: TObject);

begin

if pagecontrol1.ActivePage=tabsheet1 then begin

if datamodule4.ADOQuery1.IsEmpty then exit;

form5.Show;

form5.DataSource1.DataSet:=datamodule4.ADOQuery1;

form5.DateTimePicker1.Date:=datamodule4.ADOQuery1Datapr.Value;

end;

if pagecontrol1.ActivePage=tabsheet2 then begin

if datamodule4.ADOQuery2.IsEmpty then exit;

form5.Show;

form5.DataSource1.DataSet:=datamodule4.ADOQuery2;

form5.DateTimePicker1.Date:=datamodule4.ADOQuery2Datapr.Value;

end;

if pagecontrol1.ActivePage=tabsheet3 then begin

if datamodule4.ADOtable2.IsEmpty then exit;

form5.Show;

form5.DataSource1.DataSet:=datamodule4.ADOTable2;

form5.DateTimePicker1.Date:=datamodule4.ADOTable2Datapr.Value;

end;

end;

* процедура удаления записи

procedure TForm2.bsSkinButton2Click(Sender: TObject);

begin

if pagecontrol1.ActivePage=tabsheet1 then begin

if datamodule4.ADOQuery1.IsEmpty then exit;

if datamodule4.ADOQuery1 = nil then Exit;

 if MessageBox(0,'Вы действительно хотите удалить?','Подтверждение',

 33) <> IDOKthen Exit;

 datamodule4.ADOQuery1.Delete;

 datamodule4.ADOQuery1.Requery;

 end;

if pagecontrol1.ActivePage=tabsheet2 then begin

if datamodule4.ADOQuery2.IsEmpty then exit;

if datamodule4.ADOQuery2 = nil then Exit;

 if MessageBox(0,'Вы действительно хотите удалить?','Подтверждение',

 33) <> IDOKthen Exit;

 datamodule4.ADOQuery2.Delete;

 datamodule4.ADOQuery2.Requery;

 end;

if pagecontrol1.ActivePage=tabsheet3 then begin

if datamodule4.ADOTable2.IsEmpty then exit;

if datamodule4.ADOTable2 = nil then Exit;

 if MessageBox(0,'Вы действительно хотите удалить?','Подтверждение',

 33) <> IDOKthen Exit;

 datamodule4.ADOTable2.Delete;

 datamodule4.ADOTable2.Requery;

end;

end;

* процедура добавления записи

procedure TForm3.Button1Click(Sender: TObject);

var s,b:string;

a:integer;

begin

s:=datetostr(datetimepicker1.Date);

a:=strtoint(copy(s,7,4));

a:=a+1;

b:=copy(s,1,6);

s:=b+inttostr(a);

datamodule4.ADOQuery1.Insert;

datamodule4.ADOQuery1['Fam']:=edit1.Text;

datamodule4.ADOQuery1['Imia']:=edit2.Text;

datamodule4.ADOQuery1['Otc']:=edit3.Text;

datamodule4.ADOQuery1['Zvanie']:=edit4.Text;

datamodule4.ADOQuery1['Rugo']:=combobox2.Text;

datamodule4.ADOQuery1['Sl']:=edit6.Text;

datamodule4.ADOQuery1['Podr']:=edit7.Text;

datamodule4.ADOQuery1['Dolj']:=edit8.Text;

datamodule4.ADOQuery1['Vzisk']:=combobox1.Text;

datamodule4.ADOQuery1['Chpr']:=combobox3.Text;

datamodule4.ADOQuery1['Nompr']:=edit11.Text;

datamodule4.ADOQuery1['Datapr']:=datetimepicker1.Date;

datamodule4.ADOQuery1['Opis']:=edit13.Text;

datamodule4.ADOQuery1['Dataspr']:=strtodate(s);

datamodule4.ADOQuery1.Post;

form3.Hide;

end;

Полный листинг программы представлен в приложении А.

4. Описание программного средства

4.1 Общие сведения

Программное средство «Автоматизация учета правонарушений в УВД Миноблисполкома» облегчает работу по ведению учета сотрудников УВД, нарушивших дисциплинарные нормы. Здесь планируется создавать отчеты и диаграммы по результатам дисциплины как по УВД так и по области в целом, вести статистику изменения показателей по сравнению с прошлым годом, все это преследует цель постоянного контроля сотрудников и принятия мер для прекращения нарушения дисциплины в УВД Миноблисполкома.

При желании пользователя информация по поиску данных может быть легко сформирован в отчет и при необходимости выведен на печать.

На компьютере, использующем разрабатываемую программу, должна быть установлена версия ОС Windows (98, 2000, XP Home Edition, XP Professional). Данное требование объясняется тем, что приложения созданные с помощью Delphi 7 рекомендуется использовать на компьютере, на котором установлена одна из перечисленных ОС.

Для установки не обходимо с диска скопировать папку DisPract, а для запуска программы необходимо запустить файл DisPract.exe. После запуска программы пользователь может совершать необходимые операции над базой путем выбора соответствующих пунктов меню или кнопок расположенных на форме.

Для получения информации о программе, а также помощи необходимо выбрать соответствующие пункты меню или при нажатии на кнопку. Пункт помощь предназначен для предоставления пользователю краткой информации об использовании программы, а также для описания всех нюансов программы. Пункт о программе или об авторе предназначен для предоставления пользователю информации о назначение программы, разработчике, среде разработки, а так же адресе электронной почты разработчика.

Для закрытия программы необходимо воспользоваться пунктом меню выход или соответствующий кнопкой “Выход”, расположенной в любом окне программы.

4.2 Функциональное назначение

В приложении “DisPract” были реализованы следующие функции:

* возможность добавления, редактирование и удаления записи;
* выбор должности, звания, областей, службы и подразделения из блока;
* запросы по блокам по фамилии, области, службе, подразделению, по фамилии; по должностям; по приказу;
* отчеты по запросам с возможностью предварительного просмотра и выводом на печать;
* автоматическое снятие взыскания через год;
* досрочное снятие взыскания.

Программный продукт хранит информацию о сотрудниках в таблицах Microsoft Office Access, это позволяет легко обмениваться данными между различными пользователями, для это необходимо скопировать в папке DisPract файл pr.mdb.

5. Программа и методика испытаний

5.1 Цель испытаний

Отладка программы – это деятельность, направленная на обнаружение и исправление ошибок в ПС с использованием процессов выполнения его программ. Его цель – проверка синтаксической и логической правильности программы, а также определение того, что программа функционирует на всем диапазоне допустимых данных.

В процессе отладки программы выделяются этапы:

– трансляция исходного текста программы – выполняется перевод программы, понятной человеку, на язык, понятный компьютеру. В ней последовательно выполняются лексический, синтаксический, семантический анализ, генерация промежуточного кода, оптимизация промежуточного кода, генерация внутреннего представления;

– компоновка программы – создает готовую для работы программу, которая называется также исполняемой программой или загрузочным модулем. При этом решаются две основные задачи: если необходимо использование специальных функций, например математических, то их программные модули выбираются из библиотеки подпрограмм соответствующей системы программирования и вставляются в объектный модуль; объектный модуль преобразуется в соответствии с реальными адресами основной памяти, куда будет размещаться программа для выполнения;

– выполнение программы с целью определения логических ошибок – осуществляется после успешной компоновки или по ходу интерпретации каждого оператора. При этом выявляются такие ошибки, как, например, деление на ноль или вычисление логарифма отрицательного числа. Здесь же апробируется корректность описания используемых переменных и законченность циклов;

– тестирование программы – имеет целью определение работоспособности программы на всем требуемом диапазоне исходных данных. Программистом составляется представительная выборка исходной информации, которая позволит определить корректность программы при любых входных параметрах.

Основная цель тестирования обнаружения ошибок – выявление всех отклонений результатов функционирования реальной программы от заданных значений. Задача состоит в обнаружении максимального числа ошибок, в качестве которых принимается любое отклонение от заданных.

Поле тестирования для обнаружения ошибок применяется тестирование для их диагностики локализации. Основная задача – точно установить место искажения программы или данных, явившегося причиной отклонения результатов от заданных.

После локализации и устранения, обнаруженных ошибок применяется контрольное тестирование, задача которого состоит в подтверждении правильности выполнения корректировки программы.

5.2 Методы отладки программы

Статическое тестирование является наиболее формализованным и автоматизируемым методом проверки программ. В качестве эталонов применяются правила структурного построения программных модулей и обработки данных. Проверка степени этих правил проводится без использования объектного кода программы путем формального анализа текста программы на языке программирования. Операторы и операнды текста программ при этом анализируется в символьном виде, поэтому такой метод называют символьным тестированием.

Динамическое тестирование, особенностью этого метода является проверка исполнения тестов программ в объектном коде. При этом контролируется каждая комбинация исходных заданных данных, выходных данных и соответствующая ей комбинация результатов функционирования программы. Это позволяет выполнить отклонение результатов от заданных с фиксированием конкретных значений исходных и результирующих данных, при которых это отклонение обнаружено.

Стохастическое тестирование основано на генерации тестовых наборов, а именно множества случайных величин. В результате возможно более широкое варьирование исходных данных.

Тестирование в реальном времени. В процессе такого тестирования проверяются исполнение программ, и обработка исходных данных с учетом времени их поступления, длительности и приоритетности обработки, динамики использования памяти и взаимодействие с другими программами и так далее. При обнаружении отклонений результатов исполнения программ от предполагавшихся значений для локализации ошибок фиксируется время и переходит к детерминированному тестированию.

5.3 Тестирование программы

Тестирование программы включало в себя несколько компонентов:

– проверка правильности взаимодействия элементов интерфейса;

– тестирование методов обработки данных;

– тестирование оптимизационных алгоритмов.

Так как в процессе тестирования важно убедиться в правильности работы программы, как в стандартных ситуациях, так и в тех, которые, по мнению разработчика либо маловероятны, либо невозможны в принципе, то к этому процессу отнеслись весьма серьезно.

Поэтому был смоделирован ряд ситуаций, которые опытный пользователь вряд ли допустит, но вполне можно ожидать от новичка. Эти случаи, в основном, касались задания исходных данных.

Особенностью тестирование любой расчетной программы является наличие тестов для проверки корректности реализации расчетных алгоритмов, а также таких свойств как точность, количество шагов и время расчетов.

5.4 Выявление ошибок

Выявление ошибок иначе называют тестированием. Некоторые ошибки проявляются после первого же запуска программы на выполнение, а некоторые проявляются в чисто случайные моменты работы программы. Эти ошибки считаются самыми сложными.

Также существует такой ряд ошибок, которые связаны с неправильным распределением ресурсов компьютера. Они в основном скрыты но при увеличении загрузки системных ресурсов могут появится.

Процесс нахождения и исправления ошибок может длиться бесконечно, поэтому в основном применяют бета-тестирование. Ошибки, выявленные на этапе бета-тестирования, фиксируются и устраняются в коммерческой версии продукта. Еще одним положительным моментом бета-тестирования является то, что оно позволяет оценить программу с точки зрения удобства ее использования, собрать критические замечания пользователей и учесть их в конечном варианте программы.

После того, как ошибка зафиксирована, необходимо выяснить причину ее возникновения и найти в исходном тексте программы то место, в котором она возникает, т.е. нужно ее локализировать.

В общем случае причина ошибки может быть связана с недопониманием задачи, недостатками проектирования, ошибками, допущенными при кодировании и многими другими несоответствиями.

Самыми распространенными ошибками являются ошибки кодирования, а именно: использование переменных без предварительной инициализации, выход за границы диапазона при работе со строками либо массивами, неверный расчет числа итераций цикла, переполнение разрядной сетки при выполнении арифметических операций, накопление погрешности вычислений в операциях с вещественными числами и др.

При исправлении ошибок самое главное не наделать новых и не спешить, а обдуманно проанализировать ошибку, чем в дальнейшем к ней возвращаться.

Отладка программы осуществлялась встроенными средствами среды Delphi. С помощью встроенного компилятора были обнаружены синтаксические ошибки, которые тут же исправлялись с помощью встроенного редактора текста программы.

В ходе разработки данного программного продукта возникали вопросы и ошибки. В процессе тестирования и отладки все ошибки были устранены, что может свидетельствовать о работоспособности программы.

После чего было проведено регрессионное тестирование которое проводиться после основного, когда исправлены все ошибки программы и внесены в нее изменений.

В результате испытаний на контрольных примерах было доказано, что данная программа работает согласно заданного алгоритма.

6. Область применения программы

6.1 Назначение программы

# Данная программа разрабатывалась по заказу УВД Миноблисполкома для ведения учета сотрудников, нарушивших дисциплинарные нормы.

# На данный момент пробная версия программы проходит апробацию в УВД Миноблисполкома.

Данное программное средство облегчает работу по ведению учета сотрудников УВД, нарушивших дисциплинарные нормы. Здесь планировалось создавать отчеты и диаграммы по результатам дисциплины как по УВД так и по области в целом, вести статистику изменения показателей по сравнению с прошлым годом, все это преследует цель постоянного контроля сотрудников и принятия мер для прекращения нарушения дисциплины в УВД Миноблисполкома. Все эти требования, на мой взгляд были достигнуты в ходе разработки приложения.

6.2 Условия применения

Правильная работоспособность программы гарантируется только в тех случаях, если:

* программа используется на компьютере, минимальные аппаратные и программные требования которого удовлетворяют системным требованиям ранее оговоренными автором проекта;
* пользователь, который использует программу, намерено не пытается изменить или удалить файлы относящиеся к приложению;
* некорректно выходит из программного средства, при чем в памяти компьютера процесс продолжает выполнятся.

При соблюдении этих требований работа с программой не доставит никаких неудобств и будет обеспечена правильная работоспособность.

6.3 Справочная система

Справочная система имеет три раздела:

* Работа с данными
* Поиск информации
* Выходная информация

Первый раздел “Работа с данными” содержит информацию при работе с записями: добавление, удаление, редактирование и досрочное снятие взыскания.

Второй раздел “Поиск информации” подробно описывает как выполнить необходимый поиск по заданному критерию, в справке используется также наглядное описание, что даже неопытному пользователю не составит труда выполнить запрос.

В разделе “Выходная информация” находится информация по формированию отчета.

Для использования справки выберите пункт меню на главной форме “Справка”, а затем “Помощь”.

При открытии справки загружается главная страница, где показан основной нюанс программы.

На рисунке 6.1 представлено содержание справочной системы приложения.

Рисунок 6.1 – Структура справочной системы

ЗАКЛЮЧЕНИЕ

Результатом данной дипломной работы было создание автоматизированного учета правонарушений в УВД Миноблисполкома.

Данный продукт содержит информацию о сотрудниках. С помощью него можно без проблем найти того или иного сотрудника по заданным критериям, а также вывести эту информацию на печать.

Программа найдет широкое применение не только среди работников отдела кадров, но и среди многих сотрудников УВД.

Разрабатываемая автоматизированная система будет отличаться от других аналогичных программных средств тем, что она проста в обращении, имеет удобный, а самое главное простой в использовании интерфейс.

Все основные требования, предъявляемые к программному продукту, в полной мере реализованы в данной разработке.

ЛИТЕРАТУРА

1. Вьюкова Н.И., Галатенко В.А., Ходулев А.Б. Систематический подход к программированию.–М.: Наука., 1988. – 208 с.
2. Калверт Ч. Delphi Энциклопедия пользователя: Пер. с англ./Чарлз Калверт.- ДиаСофт, К.:, 2000.
3. Коннолли Т., Бегг К., Страчан А. Базы данных: Проектирование, реализация и сопровождение. Теория и практика. Издание второе, исправленное, дополненное. – Издательство “Питер”, Санкт-Петербург – 2002г.
4. Культин Н.Б. Delphi в задачах и примерах. – БХВ-Петербург, СПб., 2004. – 288 с.: ил.
5. Офицеров Д.В. и др. Программирование на персональных ЭВМ: Практикум: Учеб. пособие. – Мн.: Выш.шк., 1993. –256 с.
6. Пономаренко Сергей. Базы данных в Delphi 7. Самоучитель. – Издательство «Питер», Санкт-Петербург – 2003 г.
7. Форонов В.В. Программирование баз данных в Delphi 7. Учебный курс.-СПб.: Питер, 2006.-459 с.: ил.

ПРИЛОЖЕНИЕ А

(обязательное)

Текст программы

unit Unit1; //загрузочная форма

interface

uses

 Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

 Dialogs, ComCtrls, ExtCtrls, jpeg;

type

 TForm1 = class(TForm)

 Image1: TImage;

 Timer1: TTimer;

 ProgressBar1: TProgressBar;

 procedure Timer1Timer(Sender: TObject);

 procedure FormActivate(Sender: TObject);

 private

 { Private declarations }

 public

 { Public declarations }

 end;

var

 Form1: TForm1;

implementation

uses Unit2;

{$R \*.dfm}

procedure TForm1.Timer1Timer(Sender: TObject);

var

i,k:Integer;

begin

k:=250;

timer1.Enabled:=true;

 for i:=0 to 60 do

 begin

 ProgressBar1.Position:=ProgressBar1.Position+2;

 Sleep(30);

 if i mod 10=0 then begin

 k:=k-30;

 form1.AlphaBlendValue:=k;

 end;

 end;

ProgressBar1.Position:=0;

form2.Show;

form1.Hide;

timer1.Enabled:=false;

form2.PageControl1.ActivePage:=form2.TabSheet1;

end;

procedure TForm1.FormActivate(Sender: TObject);

begin

timer1.Enabled:=true;

end;

end.

unit Unit2;//главная форма отображения информации

interface

uses

 Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

 Dialogs, StdCtrls, Buttons, ComCtrls, Grids, DBGrids, Menus, bsSkinCtrls,

 bsSkinData, BusinessSkinForm,ShellAPI;

type

 TForm2 = class(TForm)

 PageControl1: TPageControl;

 TabSheet1: TTabSheet;

 TabSheet2: TTabSheet;

 DBGrid1: TDBGrid;

 DBGrid2: TDBGrid;

 TabSheet3: TTabSheet;

 DBGrid3: TDBGrid;

 bsBusinessSkinForm1: TbsBusinessSkinForm;

 bsSkinData1: TbsSkinData;

 bsCompressedStoredSkin1: TbsCompressedStoredSkin;

 bsSkinButton1: TbsSkinButton;

 bsSkinButton2: TbsSkinButton;

 bsSkinButton3: TbsSkinButton;

 bsSkinButton4: TbsSkinButton;

 bsSkinMainMenu1: TbsSkinMainMenu;

 bsSkinMainMenuBar1: TbsSkinMainMenuBar;

 N12: TMenuItem;

 N17: TMenuItem;

 N18: TMenuItem;

 N19: TMenuItem;

 N20: TMenuItem;

 N21: TMenuItem;

 N22: TMenuItem;

 N23: TMenuItem;

 N24: TMenuItem;

 N25: TMenuItem;

 N26: TMenuItem;

 N27: TMenuItem;

 N28: TMenuItem;

 N29: TMenuItem;

 N31: TMenuItem;

 N32: TMenuItem;

 procedure BitBtn2Click(Sender: TObject);

 procedure BitBtn1Click(Sender: TObject);

 procedure FormActivate(Sender: TObject);

 procedure BitBtn3Click(Sender: TObject);

 procedure BitBtn4Click(Sender: TObject);

 procedure N3Click(Sender: TObject);

 procedure N6Click(Sender: TObject);

 procedure N7Click(Sender: TObject);

 procedure N8Click(Sender: TObject);

 procedure N9Click(Sender: TObject);

 procedure N10Click(Sender: TObject);

 procedure N11Click(Sender: TObject);

 procedure N13Click(Sender: TObject);

 procedure bsSkinButton1Click(Sender: TObject);

 procedure bsSkinButton2Click(Sender: TObject);

 procedure bsSkinButton3Click(Sender: TObject);

 procedure bsSkinButton4Click(Sender: TObject);

 procedure N20Click(Sender: TObject);

 procedure N22Click(Sender: TObject);

 procedure N23Click(Sender: TObject);

 procedure N24Click(Sender: TObject);

 procedure N25Click(Sender: TObject);

 procedure N26Click(Sender: TObject);

 procedure N32Click(Sender: TObject);

 procedure N27Click(Sender: TObject);

 procedure N17Click(Sender: TObject);

 procedure N31Click(Sender: TObject);

 procedure N18Click(Sender: TObject);

 procedure N29Click(Sender: TObject);

 private

 { Private declarations }

 public

 { Public declarations }

 end;

var

 Form2: TForm2;

implementation

uses Unit1, Unit3, Unit4, Unit5, Unit6, Unit7, Unit8, Unit9, Unit10,

 Unit11, Unit12, Unit13, Unit14, Unit15;

{$R \*.dfm}

procedure TForm2.BitBtn2Click(Sender: TObject);

begin

Form1.close;

form2.Close;

form3.Close;

form5.Close;

form6.Close;

form7.Close;

form8.Close;

form9.Close;

form10.Close;

form11.close;

form12.close;

form13.close;

end;

procedure TForm2.BitBtn1Click(Sender: TObject);

begin

form3.Show;

end;

procedure TForm2.FormActivate(Sender: TObject);

var k:integer;

begin

with datamodule4.ADOQuery1 do begin

 Close;

 SQL.Clear;

 SQL.Add('select \* from Pract');

 sql.Add('where Rugo="Аппарат УВД"');

 Open;

 if RecordCount <> 0 then

 begin

 datamodule4.DataSource1.DataSet :=datamodule4.adoquery1;

 end

 else

 begin

 end;

end;

with datamodule4.ADOQuery2 do begin

 Close;

 SQL.Clear;

 SQL.Add('select \* from Pract');

 sql.Add('where Rugo<>"Аппарат УВД"');

 Open;

 if RecordCount <> 0 then

 begin

 datamodule4.DataSource2.DataSet :=datamodule4.adoquery2;

 end

 else

 begin

 end;

end;

datamodule4.ADOTable1.Open;

datamodule4.ADOTable2.Open;

while not datamodule4.ADOTable1.Eof do

begin

k:=0;

if datamodule4.ADOTable1dataspr.Value<=date() then

begin

datamodule4.ADOTable2.Insert;

datamodule4.ADOTable2['Fam']:=datamodule4.ADOTable1fam.Value;

datamodule4.ADOTable2['Imia']:=datamodule4.ADOTable1imia.Value;

datamodule4.ADOTable2['otc']:=datamodule4.ADOTable1otc.Value;

datamodule4.ADOTable2['zvanie']:=datamodule4.ADOTable1zvanie.Value;

datamodule4.ADOTable2['rugo']:=datamodule4.ADOTable1rugo.Value;

datamodule4.ADOTable2['sl']:=datamodule4.ADOTable1sl.Value;

datamodule4.ADOTable2['podr']:=datamodule4.ADOTable1podr.Value;

datamodule4.ADOTable2['dolj']:=datamodule4.ADOTable1dolj.Value;

datamodule4.ADOTable2['vzisk']:=datamodule4.ADOTable1vzisk.Value;

datamodule4.ADOTable2['chpr']:=datamodule4.ADOTable1chpr.Value;

datamodule4.ADOTable2['nompr']:=datamodule4.ADOTable1nompr.Value;

datamodule4.ADOTable2['datapr']:=datamodule4.ADOTable1datapr.Value;

datamodule4.ADOTable2['opis']:=datamodule4.ADOTable1opis.Value;

datamodule4.ADOTable2['nomspr']:=datamodule4.ADOTable1nomspr.Value;

datamodule4.ADOTable2['dataspr']:=datamodule4.ADOTable1dataspr.Value;

datamodule4.ADOTable2.Post;

datamodule4.ADOTable1.Delete;

k:=1;

end;

datamodule4.ADOTable1.Next;

if k=1 then datamodule4.ADOTable1.Prior;

end;

datamodule4.ADOTable1.Close;

datamodule4.ADOTable2.Close;

datamodule4.ADOTable2.Active:=true;

end;

procedure TForm2.BitBtn3Click(Sender: TObject);

begin

if pagecontrol1.ActivePage=tabsheet1 then begin

if datamodule4.ADOQuery1.IsEmpty then exit;

form5.Show;

form5.DataSource1.DataSet:=datamodule4.ADOQuery1;

form5.DateTimePicker1.Date:=datamodule4.ADOQuery1Datapr.Value;

end;

if pagecontrol1.ActivePage=tabsheet2 then begin

if datamodule4.ADOQuery2.IsEmpty then exit;

form5.Show;

form5.DataSource1.DataSet:=datamodule4.ADOQuery2;

form5.DateTimePicker1.Date:=datamodule4.ADOQuery2Datapr.Value;

end;

if pagecontrol1.ActivePage=tabsheet3 then begin

if datamodule4.ADOtable2.IsEmpty then exit;

form5.Show;

form5.DataSource1.DataSet:=datamodule4.ADOTable2;

form5.DateTimePicker1.Date:=datamodule4.ADOTable2Datapr.Value;

end;

end;

procedure TForm2.BitBtn4Click(Sender: TObject);

begin

if pagecontrol1.ActivePage=tabsheet1 then begin

if datamodule4.ADOQuery1.IsEmpty then exit;

if datamodule4.ADOQuery1 = nil then Exit;

 if MessageBox(0,'Вы действительно хотите удалить?','Подтверждение',

 33) <> IDOKthen Exit;

 datamodule4.ADOQuery1.Delete;

 datamodule4.ADOQuery1.Requery;

 end;

if pagecontrol1.ActivePage=tabsheet2 then begin

if datamodule4.ADOQuery2.IsEmpty then exit;

if datamodule4.ADOQuery2 = nil then Exit;

 if MessageBox(0,'Вы действительно хотите удалить?','Подтверждение',

 33) <> IDOKthen Exit;

 datamodule4.ADOQuery2.Delete;

 datamodule4.ADOQuery2.Requery;

 end;

if pagecontrol1.ActivePage=tabsheet3 then begin

if datamodule4.ADOTable2.IsEmpty then exit;

if datamodule4.ADOTable2 = nil then Exit;

 if MessageBox(0,'Вы действительно хотите удалить?','Подтверждение',

 33) <> IDOKthen Exit;

 datamodule4.ADOTable2.Delete;

 datamodule4.ADOTable2.Requery;

end;

end;

procedure TForm2.N3Click(Sender: TObject);

begin

form1.Close;

form2.Close;

form3.Close;

form5.Close;

form6.Close;

form7.Close;

form8.Close;

form9.Close;

form10.Close;

form11.Close;

form12.Close;

form13.Close;

end;

procedure TForm2.N7Click(Sender: TObject);

begin

form6.Caption:='Поиск по званию';

form8.Show;

end;

procedure TForm2.N8Click(Sender: TObject);

begin

form6.Caption:='Поиск по области';

form9.show;

end;

procedure TForm2.N9Click(Sender: TObject);

begin

form6.Caption:='Поиск по службе';

form10.show;

end;

procedure TForm2.N10Click(Sender: TObject);

begin

form6.Caption:='Поиск по подразделению';

form11.show;

end;

procedure TForm2.N11Click(Sender: TObject);

begin

form6.Caption:='Поиск по должности';

form12.show;

end;

procedure TForm2.N13Click(Sender: TObject);

begin

form6.Caption:='Поиск по месту приказа';

form13.show;

end;

procedure TForm2.bsSkinButton1Click(Sender: TObject);

begin

Form1.close;

form2.Close;

form3.Close;

form5.Close;

form6.Close;

form7.Close;

form8.Close;

form9.Close;

form10.Close;

form11.close;

form12.close;

form13.close;

form14.Close;

form15.Close;

end;

procedure TForm2.bsSkinButton2Click(Sender: TObject);

begin

if pagecontrol1.ActivePage=tabsheet1 then begin

if datamodule4.ADOQuery1.IsEmpty then exit;

if datamodule4.ADOQuery1 = nil then Exit;

 if MessageBox(0,'Вы действительно хотите удалить?','Подтверждение',

 33) <> IDOKthen Exit;

 datamodule4.ADOQuery1.Delete;

 datamodule4.ADOQuery1.Requery;

 end;

if pagecontrol1.ActivePage=tabsheet2 then begin

if datamodule4.ADOQuery2.IsEmpty then exit;

if datamodule4.ADOQuery2 = nil then Exit;

 if MessageBox(0,'Вы действительно хотите удалить?','Подтверждение',

 33) <> IDOKthen Exit;

 datamodule4.ADOQuery2.Delete;

 datamodule4.ADOQuery2.Requery;

 end;

if pagecontrol1.ActivePage=tabsheet3 then begin

if datamodule4.ADOTable2.IsEmpty then exit;

if datamodule4.ADOTable2 = nil then Exit;

 if MessageBox(0,'Вы действительно хотите удалить?','Подтверждение',

 33) <> IDOKthen Exit;

 datamodule4.ADOTable2.Delete;

 datamodule4.ADOTable2.Requery;

end;

end;

procedure TForm2.bsSkinButton3Click(Sender: TObject);

begin

if pagecontrol1.ActivePage=tabsheet1 then begin

if datamodule4.ADOQuery1.IsEmpty then exit;

form5.Show;

form5.DataSource1.DataSet:=datamodule4.ADOQuery1;

form5.DateTimePicker1.Date:=datamodule4.ADOQuery1Datapr.Value;

end;

if pagecontrol1.ActivePage=tabsheet2 then begin

if datamodule4.ADOQuery2.IsEmpty then exit;

form5.Show;

form5.DataSource1.DataSet:=datamodule4.ADOQuery2;

form5.DateTimePicker1.Date:=datamodule4.ADOQuery2Datapr.Value;

end;

if pagecontrol1.ActivePage=tabsheet3 then begin

if datamodule4.ADOtable2.IsEmpty then exit;

form5.Show;

form5.DataSource1.DataSet:=datamodule4.ADOTable2;

form5.DateTimePicker1.Date:=datamodule4.ADOTable2Datapr.Value;

end;

end;

procedure TForm2.bsSkinButton4Click(Sender: TObject);

begin

form3.Show;

end;

procedure TForm2.N20Click(Sender: TObject);

begin

form1.Close;

form2.Close;

form3.Close;

form5.Close;

form6.Close;

form7.Close;

form8.Close;

form9.Close;

form10.Close;

form11.Close;

form12.Close;

form13.Close;

form14.Close;

form15.Close;

end;

procedure TForm2.N22Click(Sender: TObject);

begin

form6.Caption:='Поиск по фамилии';

form6.bsSkinButton2.Visible:=true;

form14.show;

end;

procedure TForm2.N23Click(Sender: TObject);

begin

form6.Caption:='Поиск по званию';

form8.Show;

end;

procedure TForm2.N24Click(Sender: TObject);

begin

form6.Caption:='Поиск по области';

form9.show;

end;

procedure TForm2.N25Click(Sender: TObject);

begin

form6.Caption:='Поиск по службе';

form10.show;

end;

procedure TForm2.N26Click(Sender: TObject);

begin

form6.Caption:='Поиск по подразделению';

form11.show;

end;

procedure TForm2.N32Click(Sender: TObject);

begin

form6.Caption:='Поиск по должности';

form12.show;

end;

procedure TForm2.N27Click(Sender: TObject);

begin

form6.Caption:='Поиск по месту приказа';

form13.show;

end;

procedure TForm2.N17Click(Sender: TObject);

begin

if pagecontrol1.ActivePage=tabsheet1 then begin

form7.DataSource1.DataSet:=datamodule4.ADOQuery1;

form7.Show;

end;

if pagecontrol1.ActivePage=tabsheet2 then begin

form7.DataSource1.DataSet:=datamodule4.ADOQuery2;

form7.Show;

end;

if pagecontrol1.ActivePage=tabsheet3 then begin

form7.DataSource1.DataSet:=datamodule4.ADOTable2;

form7.Show;

end;

end;

procedure TForm2.N31Click(Sender: TObject);

begin

form15.show;

end;

procedure TForm2.N18Click(Sender: TObject);

begin

if pagecontrol1.ActivePage=tabsheet1 then datamodule4.RvProject1.Execute;

if pagecontrol1.ActivePage=tabsheet2 then datamodule4.RvProject2.Execute;

end;

procedure TForm2.N29Click(Sender: TObject);

begin

shellExeCute(handle,nil,'Справка.chm',nil,nil,sw\_restore);

end;

end.

unit Unit3; //форма добавления записи

interface

uses

 Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

 Dialogs, Menus, StdCtrls, Buttons, ComCtrls, bsSkinCtrls, bsSkinData,

 BusinessSkinForm, bsdbctrls, Mask, bsSkinBoxCtrls;

type

 TForm3 = class(TForm)

 Label2: TLabel;

 Label3: TLabel;

 Label7: TLabel;

 Label8: TLabel;

 Label9: TLabel;

 Label10: TLabel;

 Label11: TLabel;

 Edit11: TEdit;

 Edit13: TEdit;

 Edit4: TEdit;

 PopupMenu1: TPopupMenu;

 N1,N2, N3, N4, N5, N6, N7, N8, N9, N10, N11, N12, N13, N14, N15, N16, N17, N18, N19,N20: TMenuItem;

 ComboBox1: TComboBox;

 ComboBox2: TComboBox;

 PopupMenu2: TPopupMenu;

 N21,N22,N23,N24,N25, N26, N27, N28, N29, N30,N31,N32,N33,N34,N35,N36,N37,N38,N39,N40,N41,N42,N43,N44,N45,N46,N47,N48, N49,N50,N51,N52,N53,N54,N55,N56,N57,N58,N59,N60,N61: TMenuItem;

 DateTimePicker1: TDateTimePicker;

 ComboBox3: TComboBox;

 bsBusinessSkinForm1: TbsBusinessSkinForm;

 bsSkinData1: TbsSkinData;

 bsCompressedStoredSkin1: TbsCompressedStoredSkin;

 bsSkinButton1: TbsSkinButton;

 Label1: TLabel;

 Edit1: TEdit;

 Edit2: TEdit;

 Edit3: TEdit;

 Label4: TLabel;

 Edit6: TEdit;

 Label5: TLabel;

 Edit7: TEdit;

 Label6: TLabel;

 Edit8: TEdit;

 bsSkinButton2: TbsSkinButton;

 Label12: TLabel;

 Label13: TLabel;

 Label14: TLabel;

 procedure N2Click(Sender: TObject);

 procedure N3Click(Sender: TObject);

 procedure N4Click(Sender: TObject);

 procedure N5Click(Sender: TObject);

 procedure N6Click(Sender: TObject);

 procedure N7Click(Sender: TObject);

 procedure N8Click(Sender: TObject);

 procedure N10Click(Sender: TObject);

 procedure BitBtn1Click(Sender: TObject);

 procedure Button1Click(Sender: TObject);

 procedure FormActivate(Sender: TObject);

 procedure bsSkinButton1Click(Sender: TObject);

 procedure bsSkinButton2Click(Sender: TObject);

 private

 { Private declarations }

 public

 { Public declarations }

 end;

var

 Form3: TForm3;

implementation

uses Unit4;

{$R \*.dfm}

procedure TForm3.N2Click(Sender: TObject);

begin

edit4.Text:='рядовой милиции';

end;

procedure TForm3.N3Click(Sender: TObject);

begin

edit4.Text:='младший сержант милиции';

end;

procedure TForm3.N4Click(Sender: TObject);

begin

edit4.Text:='сержант ';

end;

procedure TForm3.N5Click(Sender: TObject);

begin

edit4.Text:='старший сержат';

end;

procedure TForm3.N6Click(Sender: TObject);

begin

edit4.Text:='старшина милиции';

end;

procedure TForm3.N7Click(Sender: TObject);

begin

edit4.Text:='прапорщик';

end;

procedure TForm3.N8Click(Sender: TObject);

begin

edit4.Text:='старший прапорщик';

end;

procedure TForm3.N10Click(Sender: TObject);

begin

edit4.Text:='младший лейтенант милиции';

end;

……….

procedure TForm3.BitBtn1Click(Sender: TObject);

begin

form3.Hide;

end;

procedure TForm3.Button1Click(Sender: TObject);

var s,b:string;

a:integer;

begin

s:=datetostr(datetimepicker1.Date);

a:=strtoint(copy(s,7,4));

a:=a+1;

b:=copy(s,1,6);

s:=b+inttostr(a);

datamodule4.ADOQuery1.Insert;

datamodule4.ADOQuery1['Fam']:=edit1.Text;

datamodule4.ADOQuery1['Imia']:=edit2.Text;

datamodule4.ADOQuery1['Otc']:=edit3.Text;

datamodule4.ADOQuery1['Zvanie']:=edit4.Text;

datamodule4.ADOQuery1['Rugo']:=combobox2.Text;

datamodule4.ADOQuery1['Sl']:=edit6.Text;

datamodule4.ADOQuery1['Podr']:=edit7.Text;

datamodule4.ADOQuery1['Dolj']:=edit8.Text;

datamodule4.ADOQuery1['Vzisk']:=combobox1.Text;

datamodule4.ADOQuery1['Chpr']:=combobox3.Text;

datamodule4.ADOQuery1['Nompr']:=edit11.Text;

datamodule4.ADOQuery1['Datapr']:=datetimepicker1.Date;

datamodule4.ADOQuery1['Opis']:=edit13.Text;

datamodule4.ADOQuery1['Dataspr']:=strtodate(s);

datamodule4.ADOQuery1.Post;

form3.Hide;

end;

procedure TForm3.FormActivate(Sender: TObject);

begin

edit1.Text:='';

edit2.Text:='';

edit3.Text:='';

edit4.Text:='';

combobox1.Text:='';

edit6.Text:='';

combobox2.Text:='';

edit7.Text:='';

edit8.Text:='';

combobox3.Text:='';

edit11.Text:='';

datetimepicker1.Date:=date();

edit13.Text:='';

end;

procedure TForm3.bsSkinButton1Click(Sender: TObject);

var s,b:string;

a:integer;

begin

if (edit1.Text='') or (edit2.Text='') or (edit3.Text='') or (combobox2.Text='') or (combobox3.Text='') then

begin

ShowMessage('Введите все обязательные' +#13 + 'поля');

exit;

end

else begin

s:=datetostr(datetimepicker1.Date);

a:=strtoint(copy(s,7,4));

a:=a+1;

b:=copy(s,1,6);

s:=b+inttostr(a);

datamodule4.ADOQuery1.Insert;

datamodule4.ADOQuery1['Fam']:=edit1.Text;

datamodule4.ADOQuery1['Imia']:=edit2.Text;

datamodule4.ADOQuery1['Otc']:=edit3.Text;

datamodule4.ADOQuery1['Zvanie']:=edit4.Text;

datamodule4.ADOQuery1['Rugo']:=combobox2.Text;

datamodule4.ADOQuery1['Sl']:=edit6.Text;

datamodule4.ADOQuery1['Podr']:=edit7.Text;

datamodule4.ADOQuery1['Dolj']:=edit8.Text;

datamodule4.ADOQuery1['Vzisk']:=combobox1.Text;

datamodule4.ADOQuery1['Chpr']:=combobox3.Text;

datamodule4.ADOQuery1['Nompr']:=edit11.Text;

datamodule4.ADOQuery1['Datapr']:=datetimepicker1.Date;

datamodule4.ADOQuery1['Opis']:=edit13.Text;

datamodule4.ADOQuery1['Dataspr']:=strtodate(s);

datamodule4.ADOQuery1.Post;

form3.Hide;

end;

end;

procedure TForm3.bsSkinButton2Click(Sender: TObject);

begin

form3.Hide;

end;

end.

unit Unit4; //модуль компонент

interface

uses

 SysUtils, Classes, DB, ADODB, RpBase, RpSystem, RpRave, RpDefine, RpCon,

 RpConDS;

type

 TDataModule4 = class(TDataModule)

 ADOQuery1: TADOQuery;

 ADOQuery2: TADOQuery;

 DataSource1: TDataSource;

 DataSource2: TDataSource;

 ADOConnection1: TADOConnection;

 ADOTable1: TADOTable;

 DataSource3: TDataSource;

 ADOQuery1Fam: TWideStringField;

 ADOQuery1Imia: TWideStringField;

 ADOQuery1Otc: TWideStringField;

 ADOQuery1Zvanie: TWideStringField;

 ADOQuery1Rugo: TWideStringField;

 ADOQuery1Sl: TWideStringField;

 ADOQuery1Podr: TWideStringField;

 ADOQuery1Dolj: TWideStringField;

 ADOQuery1Vzisk: TWideStringField;

 ADOQuery1Chpr: TWideStringField;

 ADOQuery1Nompr: TWideStringField;

 ADOQuery1Datapr: TDateTimeField;

 ADOQuery1Opis: TWideStringField;

 ADOQuery1Nomspr: TWideStringField;

 ADOQuery1Dataspr: TDateTimeField;

 ADOQuery2Fam: TWideStringField;

 ADOQuery2Imia: TWideStringField;

 ADOQuery2Otc: TWideStringField;

 ADOQuery2Zvanie: TWideStringField;

 ADOQuery2Rugo: TWideStringField;

 ADOQuery2Sl: TWideStringField;

 ADOQuery2Podr: TWideStringField;

 ADOQuery2Dolj: TWideStringField;

 ADOQuery2Vzisk: TWideStringField;

 ADOQuery2Chpr: TWideStringField;

 ADOQuery2Nompr: TWideStringField;

 ADOQuery2Datapr: TDateTimeField;

 ADOQuery2Opis: TWideStringField;

 ADOQuery2Nomspr: TWideStringField;

 ADOQuery2Dataspr: TDateTimeField;

 ADOTable1Fam: TWideStringField;

 ADOTable1Imia: TWideStringField;

 ADOTable1Otc: TWideStringField;

 ADOTable1Zvanie: TWideStringField;

 ADOTable1Rugo: TWideStringField;

 ADOTable1Sl: TWideStringField;

 ADOTable1Podr: TWideStringField;

 ADOTable1Dolj: TWideStringField;

 ADOTable1Vzisk: TWideStringField;

 ADOTable1Chpr: TWideStringField;

 ADOTable1Nompr: TWideStringField;

 ADOTable1Datapr: TDateTimeField;

 ADOTable1Opis: TWideStringField;

 ADOTable1Nomspr: TWideStringField;

 ADOTable1Dataspr: TDateTimeField;

 ADOQuery3: TADOQuery;

 DataSource4: TDataSource;

 ADOTable2: TADOTable;

 ADOTable2Fam: TWideStringField;

 ADOTable2Imia: TWideStringField;

 ADOTable2Otc: TWideStringField;

 ADOTable2Zvanie: TWideStringField;

 ADOTable2Rugo: TWideStringField;

 ADOTable2Sl: TWideStringField;

 ADOTable2Podr: TWideStringField;

 ADOTable2Dolj: TWideStringField;

 ADOTable2Vzisk: TWideStringField;

 ADOTable2Chpr: TWideStringField;

 ADOTable2Nompr: TWideStringField;

 ADOTable2Datapr: TDateTimeField;

 ADOTable2Opis: TWideStringField;

 ADOTable2Nomspr: TWideStringField;

 ADOTable2Dataspr: TDateTimeField;

 RvDataSetConnection1: TRvDataSetConnection;

 RvProject1: TRvProject;

 RvSystem1: TRvSystem;

 RvDataSetConnection2: TRvDataSetConnection;

 RvProject2: TRvProject;

 RvProject3: TRvProject;

 RvDataSetConnection3: TRvDataSetConnection;

 ADOQuery4: TADOQuery;

 private

 { Private declarations }

 public

 { Public declarations }

 end;

var

 DataModule4: TDataModule4;

implementation

{$R \*.dfm}

end.

unit Unit5; //форма редактирования записи

interface

uses

 Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

 Dialogs, DB, StdCtrls, DBCtrls, ComCtrls, Mask, Buttons, Menus,

 bsSkinData, BusinessSkinForm, bsSkinCtrls;

type

 TForm5 = class(TForm)

 Label2: TLabel;

 Label3: TLabel;

 Label7: TLabel;

 Label8: TLabel;

 Label9: TLabel;

 Label10: TLabel;

 Label11: TLabel;

 DateTimePicker1: TDateTimePicker;

 DBEdit4: TDBEdit;

 DBComboBox1: TDBComboBox;

 DBComboBox2: TDBComboBox;

 DBEdit9: TDBEdit;

 DBEdit10: TDBEdit;

 DataSource1: TDataSource;

 bsBusinessSkinForm1: TbsBusinessSkinForm;

 bsSkinData1: TbsSkinData;

 bsCompressedStoredSkin1: TbsCompressedStoredSkin;

 Label1: TLabel;

 DBEdit1: TDBEdit;

 DBEdit2: TDBEdit;

 DBEdit3: TDBEdit;

 Label4: TLabel;

 DBEdit5: TDBEdit;

 Label5: TLabel;

 DBEdit6: TDBEdit;

 Label6: TLabel;

 DBEdit7: TDBEdit;

 bsSkinButton1: TbsSkinButton;

 bsSkinButton2: TbsSkinButton;

 DBComboBox3: TDBComboBox;

 procedure Button1Click(Sender: TObject);

 procedure BitBtn1Click(Sender: TObject);

 procedure bsSkinButton1Click(Sender: TObject);

 procedure bsSkinButton2Click(Sender: TObject);

 private

 { Private declarations }

 public

 { Public declarations }

 end;

var

 Form5: TForm5;

implementation

uses Unit4;

{$R \*.dfm}

procedure TForm5.Button1Click(Sender: TObject);

begin

DataSource1.DataSet.Edit;

DataSource1.DataSet.FieldValues['Datapr']:=DateTimePicker1.Date;

DataSource1.DataSet.Post;

form5.Hide;

end;

procedure TForm5.BitBtn1Click(Sender: TObject);

begin

form5.Hide;

end;

procedure TForm5.bsSkinButton1Click(Sender: TObject);

var s,b:string;

a:integer;

begin

s:=datetostr(datetimepicker1.Date);

a:=strtoint(copy(s,7,4));

a:=a+1;

b:=copy(s,1,6);

s:=b+inttostr(a);

DataSource1.DataSet.Edit;

DataSource1.DataSet.FieldValues['Datapr']:=DateTimePicker1.Date;

DataSource1.DataSet.FieldValues['Dataspr']:=strtodate(s);

DataSource1.DataSet.Post;

form5.Hide;

end;

procedure TForm5.bsSkinButton2Click(Sender: TObject);

begin

form5.Hide;

end;

end.

unit Unit6; //форма отображения найденной информации

interface

uses

 Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

 Dialogs, StdCtrls, Buttons, Grids, DBGrids, bsSkinCtrls, bsSkinData,

 BusinessSkinForm;

type

 TForm6 = class(TForm)

 DBGrid1: TDBGrid;

 bsBusinessSkinForm1: TbsBusinessSkinForm;

 bsSkinData1: TbsSkinData;

 bsCompressedStoredSkin1: TbsCompressedStoredSkin;

 bsSkinButton1: TbsSkinButton;

 bsSkinButton2: TbsSkinButton;

 bsSkinButton3: TbsSkinButton;

 procedure BitBtn1Click(Sender: TObject);

 procedure BitBtn2Click(Sender: TObject);

 procedure bsSkinButton1Click(Sender: TObject);

 procedure bsSkinButton2Click(Sender: TObject);

 procedure bsSkinButton3Click(Sender: TObject);

 private

 { Private declarations }

 public

 { Public declarations }

 end;

var

 Form6: TForm6;

implementation

uses Unit4, Unit2, Unit1, Unit3, Unit5, Unit7;

{$R \*.dfm}

procedure TForm6.BitBtn1Click(Sender: TObject);

begin

dbgrid1.Columns[0].Visible:=true;

dbgrid1.Columns[1].Visible:=true;

dbgrid1.Columns[2].Visible:=true;

dbgrid1.Columns[3].Visible:=true;

dbgrid1.Columns[4].Visible:=true;

dbgrid1.Columns[5].Visible:=true;

dbgrid1.Columns[6].Visible:=true;

dbgrid1.Columns[7].Visible:=true;

dbgrid1.Columns[8].Visible:=true;

dbgrid1.Columns[9].Visible:=true;

dbgrid1.Columns[10].Visible:=true;

dbgrid1.Columns[11].Visible:=true;

dbgrid1.Columns[12].Visible:=true;

form2.show;

form6.Hide;

form6.bsSkinButton2.Visible:=false;

end;

procedure TForm6.BitBtn2Click(Sender: TObject);

begin

form7.DataSource1.DataSet:=datamodule4.ADOQuery3;

form7.Show;

end;

procedure TForm6.bsSkinButton1Click(Sender: TObject);

begin

dbgrid1.Columns[0].Visible:=true;

dbgrid1.Columns[1].Visible:=true;

dbgrid1.Columns[2].Visible:=true;

dbgrid1.Columns[3].Visible:=true;

dbgrid1.Columns[4].Visible:=true;

dbgrid1.Columns[5].Visible:=true;

dbgrid1.Columns[6].Visible:=true;

dbgrid1.Columns[7].Visible:=true;

dbgrid1.Columns[8].Visible:=true;

dbgrid1.Columns[9].Visible:=true;

dbgrid1.Columns[10].Visible:=true;

dbgrid1.Columns[11].Visible:=true;

dbgrid1.Columns[12].Visible:=true;

form2.show;

form6.Hide;

form6.bsSkinButton2.Visible:=false;

end;

procedure TForm6.bsSkinButton2Click(Sender: TObject);

begin

form7.DataSource1.DataSet:=datamodule4.ADOQuery3;

form7.Show;

end;

procedure TForm6.bsSkinButton3Click(Sender: TObject);

begin

datamodule4.RvProject3.Execute;

end;

end.

unit Unit7; //форма досрочного снятия сотрудника

interface

uses

 Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

 Dialogs, StdCtrls, Mask, DBCtrls, DB, ComCtrls, bsSkinCtrls, bsSkinData,

 BusinessSkinForm;

type

 TForm7 = class(TForm)

 DBEdit1: TDBEdit;

 DBEdit2: TDBEdit;

 DBEdit3: TDBEdit;

 DataSource1: TDataSource;

 Edit1: TEdit;

 DateTimePicker1: TDateTimePicker;

 Label1: TLabel;

 Label2: TLabel;

 Label3: TLabel;

 Label4: TLabel;

 Label5: TLabel;

 bsBusinessSkinForm1: TbsBusinessSkinForm;

 bsSkinData1: TbsSkinData;

 bsCompressedStoredSkin1: TbsCompressedStoredSkin;

 bsBusinessSkinForm2: TbsBusinessSkinForm;

 bsSkinData2: TbsSkinData;

 bsCompressedStoredSkin2: TbsCompressedStoredSkin;

 bsBusinessSkinForm3: TbsBusinessSkinForm;

 bsSkinData3: TbsSkinData;

 bsCompressedStoredSkin3: TbsCompressedStoredSkin;

 bsBusinessSkinForm4: TbsBusinessSkinForm;

 bsSkinData4: TbsSkinData;

 bsCompressedStoredSkin4: TbsCompressedStoredSkin;

 bsSkinButton1: TbsSkinButton;

 bsSkinButton2: TbsSkinButton;

 procedure Button2Click(Sender: TObject);

 procedure Button1Click(Sender: TObject);

 procedure bsSkinButton1Click(Sender: TObject);

 procedure bsSkinButton2Click(Sender: TObject);

 private

 { Private declarations }

 public

 { Public declarations }

 end;

var

 Form7: TForm7;

implementation

uses Unit4, Unit6, Unit5;

{$R \*.dfm}

procedure TForm7.Button2Click(Sender: TObject);

begin

form6.show;

form7.Hide;

end;

procedure TForm7.Button1Click(Sender: TObject);

begin

DataSource1.DataSet.Edit;

DataSource1.DataSet.FieldValues['Nomspr']:=edit1.Text;

DataSource1.DataSet.FieldValues['Dataspr']:=DateTimePicker1.Date;

DataSource1.DataSet.Post;

form7.Hide;

end;

procedure TForm7.bsSkinButton1Click(Sender: TObject);

begin

DataSource1.DataSet.Edit;

DataSource1.DataSet.FieldValues['Nomspr']:=edit1.Text;

DataSource1.DataSet.FieldValues['Dataspr']:=DateTimePicker1.Date;

DataSource1.DataSet.Post;

form7.Hide;

end;

procedure TForm7.bsSkinButton2Click(Sender: TObject);

begin

form7.Hide;

end;

end.

unit Unit8; //Поиск по званию (аналогично Unit9-по области; Unit10-по службе; Unit11-по подразделению; Unit12-по должности; Unit13-по месту приказа; Unit14-по фамилии)

interface

uses

 Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

 Dialogs, StdCtrls, bsSkinCtrls, bsSkinData, BusinessSkinForm;

type

 TForm8 = class(TForm)

 ComboBox1: TComboBox;

 bsBusinessSkinForm1: TbsBusinessSkinForm;

 bsSkinData1: TbsSkinData;

 bsCompressedStoredSkin1: TbsCompressedStoredSkin;

 bsSkinButton1: TbsSkinButton;

 bsSkinButton2: TbsSkinButton;

 procedure Button1Click(Sender: TObject);

 procedure FormActivate(Sender: TObject);

 procedure bsSkinButton1Click(Sender: TObject);

 procedure bsSkinButton2Click(Sender: TObject);

 private

 { Private declarations }

 public

 { Public declarations }

 end;

var

 Form8: TForm8;

implementation

uses Unit6, Unit4, Unit2, Unit3;

{$R \*.dfm}

procedure TForm8.Button1Click(Sender: TObject);

var zv: string[30];

begin

form8.Hide;

form6.Caption:='Поиск по званию';

zv:=form8.ComboBox1.Text;

if Form2.PageControl1.ActivePage=form2.tabsheet1 then begin

if zv<> '' then

begin

with datamodule4.ADOQuery3 do begin

 Close;

 SQL.Clear; //очищение предыдущего запроса

 SQL.Add('select \* from Pract'); //выбирает все данные из таблицы Sotr1

 SQL.Add('where (Zvanie like "'+zv+ '%") and (Rugo="Аппарат УВД")'); //выбор данных из таблицы, где

 Open;

 if RecordCount <> 0 then //чтение записей пока не дойдет до последней

 begin

 datamodule4.DataSource4.DataSet:=datamodule4.adoquery3;

 end

 else

 begin

 ShowMessage('В БД сотрудников нет записей, удовлетворяющих' +

 #13 + 'критерию запроса.'); exit;

 end;

end;

end;

end;

if form2.PageControl1.ActivePage=form2.tabsheet2 then begin

if zv<> '' then

begin

with datamodule4.ADOQuery3 do begin

 Close;

 SQL.Clear; //очищение предыдущего запроса

 SQL.Add('select \* from Pract'); //выбирает все данные из таблицы Sotr1

 SQL.Add('where (Zvanie like "'+zv+ '%") and (Rugo<>"Аппарат УВД")'); //выбор данных

 Open;

 if RecordCount <> 0 then //чтение записей пока не дойдет до последней

 begin

 datamodule4.DataSource4.DataSet:=datamodule4.adoquery3;

 end

 else

 begin

 ShowMessage('В БД сотрудников нет записей, удовлетворяющих' +

 #13 + 'критерию запроса.'); exit;

 end;

end;

end;

end;

if form2.pagecontrol1.ActivePage=form2.tabsheet3 then begin

if zv<> '' then

begin

with datamodule4.ADOQuery3 do begin

 Close;

 SQL.Clear; //очищение предыдущего запроса

 SQL.Add('select \* from SnVzisk'); //выбирает все данные из таблицы Sotr1

 SQL.Add('where (zvanie like "'+zv+ '%")'); //выбор данных из таблицы, где

 Open;

 if RecordCount <> 0 then //чтение записей пока не дойдет до последней

 begin

 datamodule4.DataSource4.DataSet:=datamodule4.adoquery3;

 end

 else

 begin

 ShowMessage('В БД сотрудников нет записей, удовлетворяющих' +

 #13 + 'критерию запроса.'); exit;

 end;

end;

end;

end;

form6.Show;

end;

procedure TForm8.FormActivate(Sender: TObject);

begin

combobox1.Text:='';

end;

procedure TForm8.bsSkinButton1Click(Sender: TObject);

var zv: string[30];

begin

form8.Hide;

form6.Caption:='Поиск по званию';

zv:=form8.ComboBox1.Text;

if Form2.PageControl1.ActivePage=form2.tabsheet1 then begin

if zv<> '' then

begin

form6.Show;

form6.bsSkinButton3.Visible:=true;

with datamodule4.ADOQuery4 do begin

 Close;

 SQL.Clear; //очищение предыдущего запроса

 SQL.Add('select \* from Pract'); //выбирает все данные из таблицы Sotr1

 SQL.Add('where (Zvanie like "'+zv+ '%") and (Rugo="Аппарат УВД")'); //выбор данных

 Open;

 if RecordCount <> 0 then //чтение записей пока не дойдет до последней

 begin

 datamodule4.DataSource4.DataSet:=datamodule4.adoquery4;

 end

 else

 begin

 form6.hide;

 ShowMessage('В БД сотрудников нет записей, удовлетворяющих' +

 #13 + 'критерию запроса.'); exit;

 end;

end;

end;

end;

if form2.PageControl1.ActivePage=form2.tabsheet2 then begin

if zv<> '' then

begin

form6.Show;

form6.bsSkinButton3.Visible:=true;

with datamodule4.ADOQuery4 do begin

 Close;

 SQL.Clear; //очищение предыдущего запроса

 SQL.Add('select \* from Pract'); //выбирает все данные из таблицы Sotr1

 SQL.Add('where (Zvanie like "'+zv+ '%") and (Rugo<>"Аппарат УВД")'); //выбор данных

 Open;

 if RecordCount <> 0 then //чтение записей пока не дойдет до последней

 begin

 datamodule4.DataSource4.DataSet:=datamodule4.adoquery4;

 end

 else

 begin

 form6.hide;

 ShowMessage('В БД сотрудников нет записей, удовлетворяющих' +

 #13 + 'критерию запроса.'); exit;

 end;

end;

end;

end;

if form2.pagecontrol1.ActivePage=form2.tabsheet3 then begin

if zv<> '' then

begin

form6.Show;

form6.bsSkinButton3.Visible:=false;

with datamodule4.ADOQuery3 do begin

 Close;

 SQL.Clear; //очищение предыдущего запроса

 SQL.Add('select \* from SnVzisk'); //выбирает все данные из таблицы Sotr1

 SQL.Add('where (zvanie like "'+zv+ '%")'); //выбор данных

 Open;

 if RecordCount <> 0 then //чтение записей пока не дойдет до последней

 begin

 datamodule4.DataSource4.DataSet:=datamodule4.adoquery3;

 end

 else

 begin

 form6.hide;

 ShowMessage('В БД сотрудников нет записей, удовлетворяющих' +

 #13 + 'критерию запроса.'); exit;

 end;

end;

end;

end;

end;

procedure TForm8.bsSkinButton2Click(Sender: TObject);

begin

form8.Hide;

end;

end.

unit Unit15; //форма информации о программе

interface

uses

 Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

 Dialogs, StdCtrls, bsSkinCtrls, bsSkinData, BusinessSkinForm;

type

 TForm15 = class(TForm)

 bsBusinessSkinForm1: TbsBusinessSkinForm;

 bsSkinData1: TbsSkinData;

 bsCompressedStoredSkin1: TbsCompressedStoredSkin;

 bsSkinButton1: TbsSkinButton;

 Label1: TLabel;

 Label2: TLabel;

 Label3: TLabel;

 Label4: TLabel;

 procedure bsSkinButton1Click(Sender: TObject);

 private

 { Private declarations }

 public

 { Public declarations }

 end;

var

 Form15: TForm15;

implementation

{$R \*.dfm}

procedure TForm15.bsSkinButton1Click(Sender: TObject);

begin

form15.Hide;

end;

end.