**Содержание**

Введение

1 Описание предметной области

1.1 Постановка задачи

1.2 Характеристика входной и выходной информации

1.3 Построение модели информационной системы

1.4 Спецификация варианта использования «Выдача информации по конкретному номеру» для системы «Отель»

1.5 Диаграммы деятельности и состояния

2 Проектирование программного обеспечения. Логическое представление

2.1 Спецификации требований. Построение диаграммы классов

2.2 Анализ структуры базы данных информационной системы «Отель» и блок-схемы системы

3 Инструкция пользователю

4 Контрольный пример

Заключение

Список использованных источников

Приложение. А – Исходный код программы

**Введение**

В основе решения многих задач лежит обработка информации. Для упрощения процесса обработки информации создаются информационные системы.

Информационная система – это программный комплекс, функции которого состоят в поддержке надежного хранения информации в памяти компьютера, выполнении специфических для данного приложения преобразований информации и вычислений, предоставления для пользователя удобного и легко осваиваемого интерфейса. Обычно объемы информации, с которыми приходится иметь дело таким системам, достаточно велики, а сама информация имеет достаточно сложную структуру. Классическими примерами информационных систем являются банковские и бухгалтерские системы, системы авиационных или железнодорожных билетов, системы налоговой службы, статистические системы, системы резервирования мест в отеле и так далее.

В зависимости от специфики предметной области информационные системы могут очень сильно отличаться друг от друга по своим функциям, архитектуре, реализации, но все-таки можно выделить некоторые общие свойства характерные для информационных систем:

* информационные системы предназначены для сбора, хранения и обработки информации. Поэтому в основе любой из них лежит среда хранения и доступа к данным;
* информационные системы ориентируются на конечного пользователя, не обладающего высокой квалификацией в области применения вычислительной техники. Поэтому клиентские приложения должны обладать простым, удобным интерфейсом, который предоставляет возможность конечному пользователю выполнять все необходимые для работы функции, но в то же время не дает ему выполнять лишние действия.

Таким образом, при разработке информационной системы приходится решать следующие задачи:

* задачу разработки базы данных, предназначенной для хранения информации;
* задачу разработки графического интерфейса пользователя.

Целью выполнения данной курсовой работы является разработка информационной системы «Отель», обеспечивающую информационную поддержку отеля.

**1 Описание предметной области**

**1.1 Постановка задачи**

Необходимо построить информационную систему «Отель». База данных должна решать следующие задачи:

* ведение списка постояльцев;
* ведение архива выбывших постояльцев за последний год.

Необходимо также предусмотреть:

* получение списка свободных номеров (по количеству мест и классу);
* получение списка номеров (мест), освобождающихся в ближайшее время;
* выдачу информации по конкретному номеру;
* автоматизацию выдачи счетов на оплату номера и услуг.

**1.2 Характеристика входной и выходной информации**

Данные в базу заносятся на основании первичных документов. Основанием для ввода информации является удостоверение личности или паспорт. Входная информация в данной информационной системе формируется в виде списка постояльцев. Обязательными полями являются:

– фамилия;

– имя;

– отчество постояльцев;

– номера удостоверения постояльца;

– номер комнаты, в которую заселяется постоялец.

В результате эксплуатации данного программного продукта может быть получена следующая выходная информация:

* отчеты, то есть выдача информации по конкретному номеру (номеру комнаты или классу);
* чеки оплаты.

**1.3 Построение модели информационной системы**

Для всякой системы, разработанной с чистого листа, на первом этапе необходимо создание концептуальной модели, основанной на требованиях заказчика, с учетом специфики предприятия. Модель информационной системы разработана при помощи визуального языка моделирования общего назначения, который используется для спецификации, визуализации, конструирования и документирования, называемого UML.

UML (унифицированный язык моделирования) – это визуальный язык моделирования общего назначения, который используется для спецификации, визуализации конструирования и документирования программной системы.

Язык UML поддерживает объектно-ориентированный подход к созданию программного обеспечения.

Конструкции языка UML позволяют смоделировать статику (структуру) и динамику (поведение) системы. Система представляется в виде взаимодействующих программных модулей, которые реагируют на внешние события. Взятые в комплексе модели обеспечивают полное описание системы. Модели создаваемые с помощью UML делятся на три группы:

* статические;
* модели поведения;
* модели изменения состояния.

Модель будущей информационной системы представляется в виде диаграмм.

Для описания функционального назначения системы постоим диаграмму вариантов использования.

Диаграмма вариантов использования является исходным представлением или концептуальной моделью системы в процессе её проектирования и разработки.

Разработка диаграммы вариантов использования преследует цели:

* определить общие границы и контекст моделируемой области на начальных этапах проектирования системы;
* сформулировать общие требования к функциональному поведению проектируемой системы;
* разработать исходную концептуальную модель системы для её последующей детализации;
* подготовить исходную документацию для взаимодействия разработчиков системы с её заказчиками и пользователями.

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью так называемых вариантов использования. При этом актером называется любая сущность, взаимодействующая с системой извне.

В контексте системы «Отель» могут быть получены следующие данные:

* служащий отеля:

а) вводит список постояльцев,

б) выдает информацию по конкретному номеру;

в) ведет учет дополнительных услуг;

г) выдает чеки оплаты;

* постоялец:

а) заказывает места и дополнительные услуги,

б) оплачивает номер и услуги,

в) запрашивает информацию по конкретному номеру.

На основании перечисленного легко перечислить следующие категории актеров, взаимодействующих с системой «Отель»:

* «Служащий отеля»,
* «Постоялец».

Система должна удовлетворять следующим требованиям:

* ведение списка постояльцев;
* ведение архива выбывших постояльцев за последний год;
* ведение списка свободных номеров;
* ведение списка освободившихся номеров;
* автоматически рассчитывать оплату номера и услуг.

Исходя из этих требований, в системе будут следующие варианты использования:

* ввод списка постояльцев;
* выдача информации по конкретному номеру;
* учет дополнительных услуг;
* выписка чеков;
* заказ места;
* заказ дополнительных услуг;
* оплата счета;
* запрос информации по конкретному номеру.

На рисунке 1 представлена диаграмма вариантов использования.

**1.4 Спецификация варианта использования «Выдача информации по конкретному номеру» для системы «Отель»**

1.0 Наименование варианта использования: «Выдача информации по конкретному номеру».

1.1 Вариант использования инициируется актером «Сотрудник отеля» и выдает информацию о конкретном номере (или нескольким номерам) по запросу актера «Постоялец» или используется для отчета.

2.0 Потоки событий:

2.1 Функции варианта использования начинают выполняться с задания пароля актером «Сотрудник отеля». Система проверяет пароль на достоверность (если пароль неверен, то активизируется альтернативный поток 2.2.1). Далее система позволяет сотруднику запросить информацию по конкретному номеру или по отелю в целом и предлагает указать одну из следующих опций:

* запрос информации;
* просмотр;
* печать;
* выход.

Если выбрана опция «Запрос информации», то система отображает окно с полями ввода «Номер комнаты», «Класс». Запрос выполняется по одному из полей или по нескольким полям.

Если выбрана опция «Просмотр», то система отображает всю имеющуюся информацию.

Опция «Печать» позволяет вывести полученную информацию на печать (если информация не может быть распечатана, то выполняется альтернативный поток 2.2.2).

2.2. Альтернативные потоки:

2.2.1 Неверный пароль: актеру «Сотрудник отеля» предоставляется возможность повторить ввод пароля или завершить вариант использования.

2.2.2 Ошибка печати: система сообщает «Сотруднику отеля» о том, что в данный момент функция недоступна; вариант использования активизируется сначала.

3.0 Специальные требования: специальные требования не определены.

4.0 Предусловие:

4.1 Перед активизацией варианта использования может быть выполнен один из следующих потоков:

* «Запрашивает информацию по конкретному номеру»;
* «Выдача отчета».

5.0 Постусловия: постусловия не определены.

6.0 Дополнительные замечания: дополнительных замечаний нет.

**1.5 Диаграммы деятельности и состояния**

Для моделирования поведения системы в языке UML используются диаграммы деятельности и состояния.

Диаграмма деятельности отображает динамические характеристики системы. Состояние действия является специальным случаем состояния с некоторым входным действием и выходящим из него переходом. Фактически состояние действия моделирует один шаг алгоритма или потока управления.

Диаграмма деятельности представлена на рисунке 2.

В языке UML под состоянием понимается абстрактный класс, используемый для моделирования отдельной ситуации. Диаграмма состояний описывает процесс изменения состояний для одного класса. Главное предназначение этой диаграммы – описать возможные последовательности состояний и переходов, которые в совокупности характеризуют поведение элемента модели.

На рисунке 3 изображена диаграмма состояния.

**2 Проектирование программного обеспечения. Логическое представление**

**2.1 Спецификации требований. Построение диаграммы классов**

Центральное место в проектировании модели информационной системы занимает разработка логической модели статического представления моделируемой системы в виде диаграммы классов.

Диаграмма классов служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования.

Объектно-ориентированное программирование – это новый подход к созданию программ. По мере развития вычислительной техники возникали разные методики программирования. На каждом этапе создавался новый подход, который помогал программистам справляться с растущим усложнением программ. Первые программы создавались посредством ключевых переключателей на передней панели компьютера. Очевидно, что такой способ подходит только для очень небольших программ. Затем был изобретен язык ассемблера, который позволял писать длинные программы. Следующий шаг был сделан в 1950 году, когда был создан первый язык высокого уровня Фортран.

Используя язык высокого уровня, программисты могли писать программы до нескольких тысяч строк длинной. Для того времени указанный подход к программированию был наиболее перспективным. Однако язык программирования, легко понимаемый в коротких программах, когда дело касалось больших программ, становился нечитабельным (и неуправляемым). Избавление от таких неструктурированных программ пришло после изобретения в 1960 году языков структурного программирования (structured programming language). К ним относятся языки Алгол, Паскаль и С. Структурное программирование подразумевает точно обозначенные управляющие структуры, программные блоки, отсутствие (или, по крайней мере, минимальное использование) инструкций GOTO, автономные подпрограммы, в которых поддерживается рекурсия и локальные переменные. Сутью структурного программирования является возможность разбиения программы на составляющие ее элементы. Используя структурное программирование, средний программист может создавать и поддерживать программы свыше 50000 строк длиной.

Хотя структурное программирование, при его использовании для написания умеренно сложных программ, принесло выдающиеся результаты, даже оно оказывалось несостоятельным тогда, когда программа достигала определенной длины. Чтобы написать более сложную программу, необходим был новый подход к программированию. В итоге были разработаны принципы объектно-ориентированного программирования. OOP (Object Oriented Programming) аккумулирует лучшие идеи, воплощенные в структурном программировании, и сочетает их с мощными новыми концепциями, которые позволяют оптимально организовывать ваши программы. Объектно-ориентированное программирование позволяет вам разложить проблему на составные части. Каждая составляющая становится самостоятельным объектом, содержащим свои собственные коды и данные, которые относятся к этому объекту. В этом случае вся процедура в целом упрощается, и программист получает возможность оперировать с гораздо большими по объему программами.

Объектно-ориентированное программирование основано на объектах. Под объектом понимается фрагмент кода, который обеспечивает выполнение некоторых операций, объединяет переменные и методы управления ими. Все элементы управления интерфейса (кнопки, панели, и т.д.) являются объектами.

Можно также определить объект как совокупность свойств и методов, а также событий, на которые он может реагировать. Внешнее управление объектом осуществляется через обработчики событий. Эти обработчики обращаются к свойствам и методам объекта. Начальные значения объекта могут задаваться также в процессе проектирования установкой различных свойств. В результате выполнения методов объекта могут генерироваться новые события, воспринимаемые другими объектами программы или пользователем.

Таким образом, объектно-ориентированное программирование основано также и на событиях. Приложение отслеживает события и реагирует на необходимые.

Включать объекты в программу можно вручную, через соответствующие операторы (это делается нечасто), или путем визуального программирования, используя заготовки — компоненты.

Диаграмма классов отражает различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывает их внутреннюю структуру и типы отношений. Диаграмма классов является дальнейшим развитием концептуальной модели проектируемой системы.

|  |
| --- |
| Nomer |
| Nomer\_mesta: Integer  Klass: String  Kolichestvo\_mest: Integer  Opisanie: String  Stoimost: Integer |

|  |  |
| --- | --- |
| Postoyalec | |
| Imya\_postoyalca: String  Nomer\_mesta: Integer  Data\_zaseleniya: String  Data\_vuseleniya: String  Nomer\_ydostovereniya\_postoyalca: Integer | |
| Dopolnitelnue\_yslygi |
| Naimenovanie: String  Cena: Integer |

Рисунок 4 – Диаграмма классов

**2.2 Анализ структуры базы данных информационной системы «Отель» и блок-схемы системы**

После построения модели, необходимо определить, в какой базе данных её строить: локальной или клиент-серверной.

Автономные (локальные) базы данных хранят все данные на том ПК, на котором установлены, при этом вся сеть не используется.

В данном случае это применимо, так как информационная система разрабатывается для АРМ администратора небольшой отеля. Поэтому, основываясь на специфике информационной системы, решено использовать локальную базу данных.

Клиент-серверную модель базы данных можно построить при дальнейшем усовершенствовании этой системы. Например, при добавлении АРМ бухгалтера, АРМ администратора кафе при отелю и так далее.

Единицей хранящейся в БД информации является таблица. Каждая таблица представляет совокупность строк и столбцов. Строки соответствуют экземпляру объекта, конкретному событию или явлению, а столбцы – атрибутам (признакам, характеристикам, параметрам) этого объекта, события, явления.

Теперь определим состав таблиц и их поля и связи. В данной базе данных используется шесть таблиц.

Таблица 1 – Структура таблицы «Номера» (Nomera.db)

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование поля | Поле | Тип поля | Ширина поля |
| Номер комнаты | Nom\_kom | Short |  |
| Класс | Klass | Alpha | 20 |
| Количество мест | Kolvo\_mest | Short |  |
| Стоимость | Stoimost | $ |  |
| Описание | Opisanie | Alpha | 255 |

Таблица 2 – Структура таблицы «Постояльцы» (Post.db)

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование поля | Поле | Тип поля | Ширина поля |
| Код постояльца | K\_post | Short |  |
| Фамилия | Fam | Alpha | 20 |
| Имя | Im | Alpha | 15 |
| Отчество | Otch | Alpha | 20 |
| Название организации | Nazv\_org | Alpha | 20 |
| Номер удостоверения | Nom\_ydost | Alpha | 7 |

Таблица 3 – Структура таблицы «Занятость номеров» (Zan\_nom.db)

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование поля | Поле | Тип поля | Ширина поля |
| Номер комнаты | Nom\_kom | Short |  |
| Код постояльца | K\_post | Short |  |
| Дата заселения | Data\_zas | Date |  |
| Дата выселения | Data\_vus | Date |  |
| Код занятости | K\_zan | Short |  |

Таблица 4 – Структура таблицы «Архив выбывших постояльцев» (Arhiv.db)

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование поля | Поле | Тип поля | Ширина поля |
| Код постояльца | K\_post | Short |  |
| Фамилия | Fam | Alpha | 20 |
| Имя | Im | Alpha | 15 |
| Отчество | Otch | Alpha | 20 |
| Номер удостоверения | Nom\_ydost | Alpha | 7 |
| Номер комнаты | Nom\_kom | Short |  |
| Срок проживания | Srok\_proj | Short |  |

Таблица 5 – Структура таблицы «Дополнительные услуги» (Dop\_ys.db)

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование поля | Поле | Тип поля | Ширина поля |
| Код услуги | K\_ysl | Short |  |
| Наименование услуги | Naim | Alpha | 20 |
| Цена | Cena | Long integer |  |

Таблица 6 – Структура таблицы «Оказание дополнительных услуг» (Okaz\_dy.db)

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование поля | Поле | Тип поля | Ширина поля |
| Код услуги | K\_ysl | Short |  |
| Номер комнаты | Nom\_kom | Short |  |
| Дата оказания | Data\_okaz | Date |  |
| Код занятости | K\_zan | Short |  |

В каждой таблице определен первичный ключ – поле, однозначно идентифицирующее запись и облегчающее установление связи между таблицами. Первичными ключами являются все первые поля в каждой таблице.

Также были определены индексы. Индексы отличаются от первичных ключей тем, что не требуют непременной уникальности значений входящих в их состав полей. Они устанавливаются по полям, которые часто используются при поиске и сортировке данных: индексы помогут системе значительно быстрее найти нужные данные или отсортировать их в нужной последовательности.

На рисунке 5 приводится алгоритм основной работы информационной системы «Отель».

На рисунках 6 и 7 отображены соответственно процессы фильтрации таблиц и поиска записей в таблицах.

Начало

Просмотр списка свободных номеров

Выбор номера

Ввод данных постояльца

Ввод использованных постояльцем дополнительных услуг

Расчет оплаты за номер и дополнительные услуги

Оплата

Выдача чека

Заполнение архива выбывших постояльцев

Конец

[Есть свободные номера]

[Свободных номеров нет]

Рисунок 5 – Блок-схема процесса заселения-выселения постояльца

Начало

Выбор поля, по которому производится фильтрация

Ввод критерия

Вывод отфильтрованной таблицы на экран

Конец

+

\_

Рисунок 6 – Блок-схема процесса фильтрации таблиц

Начало

Выбор поля, по которому производится поиск

Ввод критерия

Переход курсора на необходимую строку таблицы

Конец

+

\_

Рисунок 7 – Блок-схема процесса поиска записей в таблицах

Реализация данной задачи проводится в системе программирования Delphi 7.0.

Среда Delphi — это среда быстрой разработки, обеспечивающей высокоэффективную работу программиста, в которой в качестве языка программирования используется язык Object Pascal. В Delphi есть все необходимое для проектирования, запуска и тестирования программ, и где главной целью является облегчение процесса создания программ.

В основе систем быстрой разработки лежит технология визуального проектирования и событийного программирования.

Delphi 7.0. располагает широкими возможностями по созданию приложений баз данных, необходимым набором драйверов для доступа к самым известным форматам баз данных, удобными и развитыми средствами для доступа к информации, расположенной как на локальном диске, так и на удаленном сервере, а также большим коллекцией визуальных компонент для построения отображаемых на экране окон, что необходимо для создания удобного интерфейса между пользователем и исполняемым кодом.

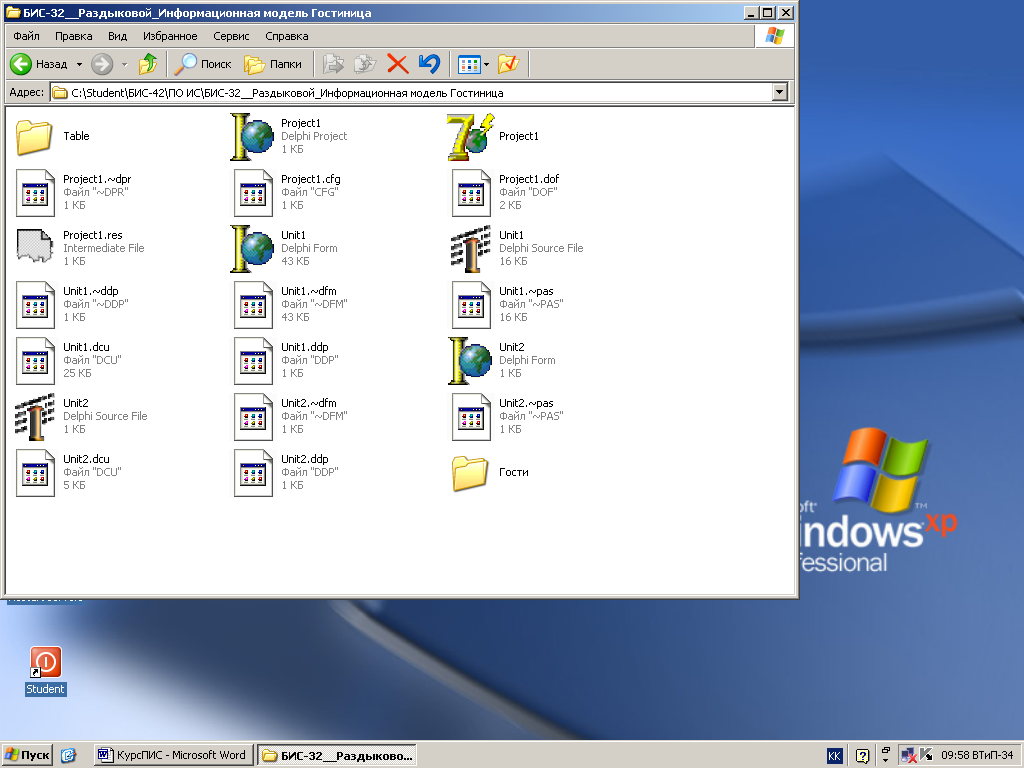
Листинг информационной системы представлен в приложении А.

**3 Инструкция пользователю**

Создайте дерево каталогов c:\STUDENT\БИС-42.

Скопируйте папку Курсовая в папку БИС-42. Откройте её.

Запустите программу, нажав на значок. В появившемся окне нажмите кнопку «Войти» (рисунок 8).



В появившейся форме «Пароль» (рисунок 9) необходимо ввести код «123» и нажать кнопку «Вход».

Теперь форма «Информационная система «Отель» стала доступной для пользователя (рисунок 10).

Путем выбора меню «Файл», который содержит команды «Открыть» и «Выход», можно активизировать любую из трех вкладок (рисунок 11):

* «Номера»;
* «Постояльцы»;
* «Дополнительные услуги»,

а также выйти из программы.

При выборе меню «Редактирование таблицы» можно выбрать одну из трех таблиц («Занятость номеров», «Постояльцы» или «Оказание дополнительных услуг») для редактирования (рисунок 12).

При этом открывается подвкладка «Правка» на одной из следующих вкладок:

* «Номера» для редактирования таблицы «Занятость номеров» (рисунок 13);
* «Постояльцы» для редактирования таблицы «Постояльцы» (рисунок 14);
* «Дополнительные услуги для редактирования таблицы «Оказание дополнительных услуг» (рисунок 15).

Для ввода новых данных необходимо сначала нажать кнопку «Очистить», затем ввести данные в каждую из ячеек и нажать кнопку «Добавить».

Для редактирования существующих записей нужно выбрать требуемую запись, внести изменения и нажать кнопку «Сохранить».

Для удаления какой-либо записи необходимо выбрать требуемую запись и нажать кнопку «Удалить».

Внимание! При удалении будьте внимательны, так как удаленные записи нельзя автоматически вернуть.

Также на подвкладке «Правка» на вкладке «Постояльцы»:

* можно определить срок проживания каждого постояльца. Для этого выберите необходимую запись и нажмите кнопку «Срок проживания»;
* можно автоматически рассчитать стоимость проживания с учетом оказанных услуг (кнопка «Счет»);
* при нажатии кнопки «Выселение» постоялец автоматически удаляется из таблиц «Постоялец» и «Занятость номеров» и перемещается в таблицу «Архив выбывших постояльцев», а комната, закрепленная за ним, становится свободной.

На подвкладке «Сведения» вкладки «Номера» (рисунок 16):

* + можно определить срок проживания постояльца;
  + получить список номеров (мест), освобождающихся на определенную дату, которую необходимо ввести в специально отведенную для этого ячейку «Дата выселения»;
  + отфильтровать таблицу «Номера».

Таблицу «Номера» можно отфильтровать по:

* + классу – для этого в ячейку «Условие фильтрации» вводим название требуемого класса номера, например, Люкс;
  + номеру комнаты – для этого в ячейку «Условие фильтрации» вводим номер комнаты, например, 103;
  + количеству мест в номере – для этого в ячейку «Условие фильтрации» вводим требуемое количество мест, например, 2;
  + цене номера – для этого в ячейку «Условие фильтрации» вводим цену, например, 1200;
  + свободным номерам.

После выбора критерия фильтрации необходимо нажать кнопку «Фильтровать».

Список свободных номеров отображается в окошке под кнопкой «Фильтровать».

При выборе критерия фильтра «Нет» необходимо также нажать кнопку «Фильтровать». Таблица станет отображать все записи.

Аналогично можно отфильтровать таблицу «Постояльцы» (рисунок 17) и произвести поиск по таблицам «Номера» (рисунок 18), «Архив выбывших постояльцев» (рисунок 19) и «Оказание дополнительных услуг» (рисунок 20).

**4 Контрольный пример**

Таблицы создавались в Database Desktop, входящей в комплект поставки Delphi, с использованием типа таблиц Paradox 7. Таблицы Paradox 7 по сравнению с другими поддерживают самый богатый набор разных типов полей, что позволяет автоматически следить за правильностью вводимых данных (пункт Validity Checks в списке Table properties). Также можно выбирать данные из другой таблицы (Lookup Table), строить вторичные индексы, в том числе составные (Secondary Indexes), следить за ссылочной целостностью БД (Referential Integrity), защищать таблицу от несанкционированного доступа (Password Security), выбирать языковой драйвер (Table Language).

Контрольный пример осуществлялся при занесении в каждую таблицу по 15-20 записей.

Таблицы «Номера» и «Дополнительные услуги» заполняются непосредственно в Database Desktop, так как данные в этих таблицах не должны быть доступны для служащих отеля.

В таблицу «Архив выбывших постояльцев» данные поступают автоматически при нажатии кнопки «Выселение». При этом данные соответственно удаляются из таблиц «Занятость номеров» и «Оказание дополнительных услуг».

При использовании фильтров и при поиске записей информация не искажалась, выдавалась точно по определенным критериям.

Таким образом, при использовании демонстрационной БД наглядно показано, что информационная система «Отель» обеспечивает:

* быстрый доступ к среде хранения данных;
* качественную обработку информации (поиск, сортировку, фильтрацию);
* удобство пользователя, так как приложение обладает простым интерфейсом, который предоставляет возможность конечному пользователю выполнять все необходимые для работы функции, но в то же время не дает ему выполнять лишние действия.

**Заключение**

В ходе выполнения данной курсовой работы была разработана информационная система «Отель», которая обеспечивает информационную поддержку деятельности отеля.

Преимущество использования автоматизированной системы состоит в том, затрачивается меньше времени на обработку данных, чем при ручной обработке, меньше места для хранения информации, так как в бумажном виде та же самая информация занимает большее пространство. Автоматизированная система отличается быстродействием, оперативностью обработки и наглядным и удобным интерфейсом.

Приложение выполняет следующий перечень требований:

* представление данных в удобном для пользователя виде;
* добавление и удаление данных;
* фильтрация и сортировка данных;
* навигация по набору данных.

Разработанная система позволяет выполнять следующие задачи:

* ведение списка постояльцев;
* ведение архива выбывших постояльцев за последний год.

**Список используемых источников**

1. Хомоненко А. Д., Гофман В. Э. Работа с базами данных в Delphi. – СПб.: БХВ – Петербург, 2005. – 640 с.
2. Маняшек Г., Лешек А. Анализ требований и проектирование систем с использованием UML.: Пер. с англ. – М.: Издательский дом «Вильямс», 2000. – 570 с.
3. Мюллер Р. Базы данных и UML. Проектирование. – М.: Издательский дом «Вильямс», 2000. – 590 с.
4. Кватрани Т. Rational Rose 2000 и UML. Визуальное моделирование. – М.: ДМК Пресс, 2001. – 613 с.
5. Глушаков С. В., Ломотько Д. В. Базы данных. – СПб.: Питер, 2000. – 387 с.
6. Фаронов В. В., Шумаков П. В. Delphi 5. Руководство разработчика баз данных. – М.: Нолидж, 2001. – 636 с.
7. Ларман к. Применение UML и шаблонов проектирования. – М.: Издательский дом «Вильямс», 2002. – 410 с.
8. Буч Г., Рамбо Д., Джекобсон А. Язык UML. Руководство пользователя. – СПб.: ДМК Пресс, 2004. – 580 с.
9. Вондров А. М. Проектирование программного обеспечения ЭИС. – М.: Финансы и статистика, 2000. – 467 с.
10. Богс М., Богс У. UML и Rational Rose. – М.: Лори, 2001. – 618 с.

**Приложение А**

**(обязательное)**

Исходный код программы

Форма №1 «Информационная система «Отель»

unit Unit1;

interface

uses

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

Dialogs, ComCtrls, Menus, ExtCtrls, Grids, DBGrids, DB, DBTables, DBCtrls,

StdCtrls, Mask;

type

TForm1 = class(TForm)

Panel1: TPanel;

MainMenu1: TMainMenu;

Panel2: TPanel;

Panel3: TPanel;

PageControl1: TPageControl;

TabSheet1: TTabSheet;

TabSheet2: TTabSheet;

TabSheet3: TTabSheet;

PageControl2: TPageControl;

TabSheet4: TTabSheet;

TabSheet5: TTabSheet;

TabSheet6: TTabSheet;

PageControl3: TPageControl;

TabSheet7: TTabSheet;

TabSheet8: TTabSheet;

TabSheet9: TTabSheet;

PageControl4: TPageControl;

TabSheet10: TTabSheet;

TabSheet11: TTabSheet;

N1: TMenuItem;

N2: TMenuItem;

N6: TMenuItem;

N11: TMenuItem;

N12: TMenuItem;

N13: TMenuItem;

DataSource1: TDataSource;

DBNavigator1: TDBNavigator;

Table1: TTable;

DBGrid1: TDBGrid;

Label1: TLabel;

DataSource2: TDataSource;

DBGrid2: TDBGrid;

DBNavigator2: TDBNavigator;

Table2: TTable;

DataSource3: TDataSource;

DBGrid3: TDBGrid;

DBNavigator3: TDBNavigator;

DBGrid4: TDBGrid;

DBNavigator4: TDBNavigator;

DataSource4: TDataSource;

Table3: TTable;

Table4: TTable;

Table5: TTable;

Table6: TTable;

DataSource5: TDataSource;

DBGrid5: TDBGrid;

DBNavigator5: TDBNavigator;

DBGrid6: TDBGrid;

DBNavigator6: TDBNavigator;

Label2: TLabel;

Label3: TLabel;

Label4: TLabel;

Label5: TLabel;

Label6: TLabel;

RadioGroup1: TRadioGroup;

Label12: TLabel;

Label13: TLabel;

Label14: TLabel;

Label15: TLabel;

Button2: TButton;

Button3: TButton;

Button4: TButton;

Button5: TButton;

RadioGroup2: TRadioGroup;

Label23: TLabel;

Label24: TLabel;

Label26: TLabel;

Label27: TLabel;

Label28: TLabel;

Label29: TLabel;

Label30: TLabel;

Label31: TLabel;

Label32: TLabel;

Button6: TButton;

Button7: TButton;

Button8: TButton;

Button9: TButton;

Label33: TLabel;

Label34: TLabel;

Label35: TLabel;

Label41: TLabel;

Label42: TLabel;

Label43: TLabel;

Button11: TButton;

Button12: TButton;

Button13: TButton;

Label44: TLabel;

Label45: TLabel;

Label46: TLabel;

Button14: TButton;

Table1Nom\_kom: TSmallintField;

Table1Klass: TStringField;

Table1Kolvo\_mest: TSmallintField;

Table1Stoimost: TCurrencyField;

Table1Opisanie: TStringField;

Table2Nom\_kom: TSmallintField;

Table2K\_post: TSmallintField;

Table2Data\_zas: TDateField;

Table2Data\_vus: TDateField;

Table2Bron: TBooleanField;

Table2K\_zan: TSmallintField;

Table3K\_post: TSmallintField;

Table3Fam: TStringField;

Table3Im: TStringField;

Table3Otch: TStringField;

Table3Nazv\_org: TStringField;

Table3Nom\_ydost: TStringField;

Table4K\_post: TSmallintField;

Table4Fam: TStringField;

Table4Im: TStringField;

Table4Otch: TStringField;

Table4Nom\_ydost: TStringField;

Table4Nom\_kom: TSmallintField;

Table4Srok\_proj: TIntegerField;

Table5K\_ysl: TSmallintField;

Table5Naim: TStringField;

Table6K\_ysl: TSmallintField;

Table6Nom\_kom: TSmallintField;

Table6Data\_okaz: TDateField;

Table6K\_zan: TSmallintField;

Button15: TButton;

DBEdit1: TDBEdit;

DBEdit2: TDBEdit;

DBEdit3: TDBEdit;

DBEdit4: TDBEdit;

DBEdit5: TDBEdit;

Button16: TButton;

Label47: TLabel;

DBEdit6: TDBEdit;

Edit4: TEdit;

Button17: TButton;

RadioGroup4: TRadioGroup;

Edit5: TEdit;

StringGrid1: TStringGrid;

Edit6: TEdit;

Label16: TLabel;

Button18: TButton;

DBEdit8: TDBEdit;

DBEdit9: TDBEdit;

DBEdit10: TDBEdit;

DBEdit11: TDBEdit;

DBEdit12: TDBEdit;

DBEdit13: TDBEdit;

Button19: TButton;

Button20: TButton;

Edit8: TEdit;

Label17: TLabel;

Edit1: TEdit;

Label7: TLabel;

DataSource6: TDataSource;

Edit2: TEdit;

Label8: TLabel;

Edit3: TEdit;

Label9: TLabel;

Table5Cena: TIntegerField;

Label10: TLabel;

DBEdit7: TDBEdit;

DBEdit14: TDBEdit;

DBEdit15: TDBEdit;

DBEdit16: TDBEdit;

Button1: TButton;

Edit7: TEdit;

Edit9: TEdit;

Edit10: TEdit;

Edit11: TEdit;

Edit12: TEdit;

Edit13: TEdit;

Button10: TButton;

Edit14: TEdit;

Edit15: TEdit;

procedure FormActivate(Sender:TObject);

procedure N6Click(Sender: TObject);

procedure N11Click(Sender: TObject);

procedure N12Click(Sender: TObject);

procedure N13Click(Sender: TObject);

procedure Button2Click(Sender: TObject);

procedure Button6Click(Sender: TObject);

procedure Button11Click(Sender: TObject);

procedure Button4Click(Sender: TObject);

procedure Button3Click(Sender: TObject);

procedure Button8Click(Sender: TObject);

procedure Button13Click(Sender: TObject);

procedure Button15Click(Sender: TObject);

procedure Button16Click(Sender: TObject);

procedure Button17Click(Sender: TObject);

procedure Button5Click(Sender: TObject);

procedure Button18Click(Sender: TObject);

procedure Button7Click(Sender: TObject);

procedure Button19Click(Sender: TObject);

procedure Button20Click(Sender: TObject);

procedure Button12Click(Sender: TObject);

procedure Button1Click(Sender: TObject);

procedure Button14Click(Sender: TObject);

procedure Button9Click(Sender: TObject);

procedure Button10Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form1: TForm1;

implementation

uses Unit2;

{$R \*.dfm}

procedure TForm1.FormActivate(Sender:TObject);

Var k : string;

begin

k:=GetCurrentDir+'\Table\';

Table1.Active:=false;

Table2.Active:=False;

Table3.Active:=False;

Table4.Active:=False;

Table5.Active:=False;

Table6.Active:=False;

Table1.DatabaseName:=k;

Table2.DatabaseName:=k;

Table3.DatabaseName:=k;

Table4.DatabaseName:=k;

Table5.DatabaseName:=k;

Table6.DatabaseName:=k;

Table1.TableName:='Nomera';

Table2.TableName:='Zan\_nom';

Table3.TableName:='Post';

Table4.TableName:='Arhiv';

Table5.TableName:='Dop\_ysl';

Table6.TableName:='Okaz\_dy';

Table1.Active:=True;

Table2.Active:=True;

Table3.Active:=True;

Table4.Active:=True;

Table5.Active:=True;

Table6.Active:=True;

end;

procedure TForm1.N6Click(Sender: TObject);

begin

Close;

end;

procedure TForm1.N11Click(Sender: TObject);

begin

PageControl1.Visible:=True;

PageControl1.ActivePage:=TabSheet1;

end;

procedure TForm1.N12Click(Sender: TObject);

begin

PageControl1.Visible:=True;

PageControl1.ActivePage:=TabSheet2;

end;

procedure TForm1.N13Click(Sender: TObject);

begin

PageControl1.Visible:=True;

PageControl1.ActivePage:=TabSheet3;

end;

procedure TForm1.Button2Click(Sender: TObject);

begin

DBEdit1.Clear;

DBEdit2.Clear;

dbEdit3.Clear;

dbEdit4.Clear;

dbEdit5.Clear;

DBEdit6.Clear;

end;

procedure TForm1.Button6Click(Sender: TObject);

begin

DBEdit8.Clear;

DBEdit9.Clear;

DBEdit10.Clear;

DBEdit11.Clear;

DBEdit12.Clear;

DBEdit13.Clear;

end;

procedure TForm1.Button11Click(Sender: TObject);

begin

DBEdit7.Clear;

DBEdit14.Clear;

DBEdit15.Clear;

DBEdit16.Clear;

end;

procedure TForm1.Button4Click(Sender: TObject);

begin

Table2.Insert;

end;

procedure TForm1.Button3Click(Sender: TObject);

begin

if MessageDlg(‘Вы действительно хотите удалить запись?', mtConfirmation,[mbYes,mbNo],0) = mrYes then

Table2.Delete;

end;

procedure TForm1.Button8Click(Sender: TObject);

begin

if MessageDlg(‘Вы действительно хотите удалить запись?', mtConfirmation,[mbYes,mbNo],0) = mrYes then

Table3.Delete;

end;

procedure TForm1.Button13Click(Sender: TObject);

begin

if MessageDlg(‘Вы действительно хотите удалить запись?', mtConfirmation,[mbYes,mbNo],0) = mrYes then

Table6.Delete;

end;

procedure TForm1.Button15Click(Sender: TObject);

begin

Form2.Show;

Edit6.Text:=FormatDateTime('dd.mm.yyyy',now());

StringGrid1.Cells[0,0]:='Íîìåð';

StringGrid1.ColWidths[0]:=200;

Panel1.Caption:=FormatDateTime('dd.mm.yyyy',now());

end;

procedure TForm1.Button16Click(Sender: TObject);

begin

try

Table2.Edit;

Table2.Post;

except

MessageDlg('Введены некорректные данные',mtError,[mbOK],0);

Table2.Cancel;

end;

end;

procedure TForm1.Button17Click(Sender: TObject);

begin

If RadioGroup1.ItemIndex=0 then Table1.Filtered:=false else

begin

Case RadioGroup1.ItemIndex of

1: Table1.Filter:='Klass = '+edit4.Text;

2: Table1.Filter:='Nom\_kom = '+edit4.Text;

3: Table1.Filter:='Kol-vo\_mest = '+edit4.Text;

4: Table1.Filter:='Stoimost = '+edit4.Text;

end;

Table1.Filtered:=true;

end;

end;

procedure TForm1.Button5Click(Sender: TObject);

begin

Case RadioGroup4.ItemIndex of

0: Table1.Locate('Nom\_kom',strtoint(Edit5.Text),[]);

1: Table1.Locate('Klass',Edit5.Text,[]);

2: Table1.Locate('Kol-vo\_mest',strtoint(Edit5.Text),[]);

end;

end;

procedure TForm1.Button18Click(Sender: TObject);

Var i:integer;

begin

Table2.First;

StringGrid1.RowCount:=2;

While not Table2.Eof do

begin

i:=StringGrid1.RowCount;

If Table2Data\_vus.AsString=Edit6.Text then

begin

StringGrid1.Cells[0,i-1]:=Table2Nom\_kom.AsString;

StringGrid1.RowCount:=i+1;

end;

Table2.Next;

end;

end;

procedure TForm1.Button7Click(Sender: TObject);

begin

Table3.Insert;

end;

procedure TForm1.Button19Click(Sender: TObject);

begin

try

Table3.Edit;

table3.Post;

except

MessageDlg(‘Введены некорректные данные', mtError,[mbOK],0);

table3.Cancel;

end;

end;

procedure TForm1.Button20Click(Sender: TObject);

Var cen,st:integer;

begin

st:=0;

Table4.Insert;

Table4K\_post.AsString:=Table3K\_post.AsString;

Table4Fam.AsString:=Table3Fam.AsString;

Table4Im.AsString:=Table3Im.AsString;

Table4Otch.AsString:=Table3Otch.AsString;

Table4Nom\_ydost.AsString:=Table3Nom\_ydost.AsString;

Table2.First;

While not Table2.Eof do

begin

If Table2K\_post.AsString=Table3K\_post.AsString then

Table4Nom\_kom.AsString:=Table2Nom\_kom.AsString;

table2.Next;

end;

Table4Srok\_proj.AsString:=edit8.Text;

Table4.Post;

Table1.First;

While not table1.Eof do

begin

If Table1Nom\_kom.AsString=Table4Nom\_kom.AsString then

cen:=Table1Stoimost.AsInteger;

Table1.Next;

end;

Edit1.Text:=inttostr(cen\*Table4Srok\_proj.AsInteger);

Table2.Locate('K\_post',Table3K\_post.AsInteger,[]);

Table6.First;

While not Table6.Eof do

begin

If Table6Nom\_kom.AsString=Table2Nom\_kom.asstring then

begin

Table5.Locate('K\_ysl',Table6K\_ysl.AsInteger,[]);

cen:=Table5Cena.AsInteger;

st:=st+cen;

end;

Table6.Next;

end;

Edit2.Text:=inttostr(st);

Edit3.Text:=inttostr(strtoint(edit1.Text)+st);

Table6.First;

While not Table6.Eof do

begin

If Table6K\_zan.AsString=Table2K\_zan.AsString

then Table6.Delete;

Table6.Next;

end;

Table2.Delete;

Table3.Delete;

end;

procedure TForm1.Button12Click(Sender: TObject);

begin

Table6.Insert;

end;

procedure TForm1.Button1Click(Sender: TObject);

begin

try

Table6.Edit;

Table6.Post;

Except

Table6.Cancel;

end;

end;

procedure TForm1.Button14Click(Sender: TObject);

begin

Table6.Locate('K\_ysl;Nom\_kom;Data\_okaz',VarArrayOf([StrToInt(Edit7.Text),StrToInt(Edit9.Text),StrToDate(Edit10.Text)]),[]);

end;

procedure TForm1.Button9Click(Sender: TObject);

begin

Table4.Locate('Nom\_kom;Fam;Im',VarArrayOf([StrToInt(Edit11.Text),Edit12.Text,Edit13.Text]),[]);

end;

procedure TForm1.Button10Click(Sender: TObject);

begin

If RadioGroup2.ItemIndex=0 then Table3.Filtered:=false else

begin

Case RadioGroup2.ItemIndex of

1: Table3.Filter:='Fam = '+ edit14.Text;

2: Table3.Filter:='K\_post = '+ edit15.Text;

end;

Table3.Filtered:=true;

end;

end;

end.

Форма №2 «Пароль»

unit Unit2;

interface

uses

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

Dialogs, StdCtrls;

type

TForm2 = class(TForm)

Edit1: TEdit;

Button1: TButton;

procedure Button1Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

const s=123;

var

Form2: TForm2;

implementation

uses unit1;

{$R \*.dfm}

procedure TForm2.Button1Click(Sender: TObject);

begin

If Edit1.Text=inttostr(s)

then

begin

Form1.N1.Enabled:=true;

close;

end

else

begin

If MessageDlg('Повторить попытку?',mtError,[mbOK,mbNo],0)=mrOK

then Edit1.Text:=''

else

begin

Form1.Close;

close;

end;

end;

Form1.Button15.Visible:=false;

end;

end.