

ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ

Физико-технический факультет

Кафедра «Компьютерных технологий»

Лабораторная работа №2

Курс «Технологии разработки программного обеспечения»

Студента 1 ускоренного курса

Группы ИВТ 6

Очной формы обучения

Ошиткова Виталия Андреевича

Преподаватель

старший преподаватель

Колесник Анатолий Васильевич

Донецк – 2020

## СОДЕРЖАНИЕ

1. Технологический раздел.....	2
1.1 Выбор методов реализации.....	2
1.2 Выбор среды и языка программирования .....	3
1.3 Выбор технологии программирования .....	5
2. Разработка интерфейса программы.....	8
2.1 Разработка общего интерфейса .....	8
2.2 Форма «Каталог стран» .....	9
2.3 Форма «Каталог туров» .....	10
2.4 Форма «Бронирование туров» .....	12
2.5 Форма «Добавление / редактирование сведений о турах» .....	14
2.6 Форма «О клиентах».....	15
2.7 Форма «Подбор тура».....	16
2.8 Форма «Запрос на выборку в диапазоне дат» .....	17
2.9 Форма «Запрос на выборку в диапазоне цен» .....	18
2.10 Форма «Отчет о бронировании туров».....	18
2.11 Форма «Туристическая путевка» .....	19
2.12 Компоненты интерфейса.....	20
3. Технический проект .....	24
3.1 Разработка технического проекта .....	24

## **1. Технологический раздел**

### **1.1 Выбор методов реализации**

Проектирование схемы базы данных должно решать задачи минимизации дублирования данных и упрощения процедур их обработки и обновления.

В качестве методов реализации выбранной темы рассматривались несколько вариантов:

- иерархическая модель данных;
- сетевая модель данных;
- реляционная модель данных.

Иерархическая модель данных (ИМД), основана на понятии деревьев, состоящих из вершин и ребер. Вершине дерева ставится в соответствие совокупности атрибутов данных, характеризующих некоторый объект. Вершины и ребра дерева как бы образуют иерархическую древовидную структуру, состоящую из  $n$  уровней. Данный метод реализации выбранной темы не приемлем, так как операции в ИМД имеют нелогичный позаписный характер. Механизмы доступа к данным и перемещения по структуре данных в таких моделях достаточно сложны и существенным образом опираются на концепцию текущего состояния механизма доступа. Отношение "многие-ко-многим" реализуется очень сложно, дает громоздкую структуру и требует хранения избыточных данных, иерархическая упорядоченность усложняет операции удаления и добавления, доступ к любой вершине возможен только через корневую вершину, что увеличивает время доступа, сложно осуществить поиск и сортировку данных.

В сетевой модели данных больше внимания уделяется структуризации данных, чем развитию ее операционных возможностей. Наборы отношений и структуру записей необходимо задавать наперед. Изменение структуры базы

данных обычно ведет к перестройке всей базы данных. Конечно, сетевая модель имеет ряд преимуществ: гибкость, стандартизация, быстроедействие.

В данной дипломной работе использована реляционная модель данных. Реляционной называется база данных, в которой все данные, доступные пользователю, организованы в виде таблиц, все операции над данными сводятся к операциям над этими таблицами. Данная модель представления данных обладает рядом неоспоримых преимуществ:

- 1) доступ к данным свободен от двусмысленности;
- 2) автоматически поддерживается целостность данных;
- 3) перенос базы данных на другой компьютер не оказывает влияния на приложение;
- 4) удобно использовать для хранения любого количества любых данных, особенно если данные однородные;
- 5) полная независимость данных;
- 6) возможна выборка по заданным условиям;
- 7) легко осуществить поиск данных.

## **1.2 Выбор среды и языка программирования**

Бурное развитие вычислительной техники, потребность в эффективных средствах разработки программного обеспечения привели к появлению систем программирования, ориентированных на «быструю разработку». В основе систем быстрой разработки или RAD-систем (Rapid Application Development) лежит технология визуального проектирования и событийного программирования. Ее суть заключается в том, что среда разработки берет на себя большую рутинной работы, оставляя программисту работу по конструированию окон и созданию функций обработки событий повышая

этим производительность работы программиста.

Разработчики приложений баз данных выдвигают требования к инструментам, при помощи которых такие приложения можно создавать, которые в общем виде можно сформулировать как: "быстрота, простота, эффективность, надежность".

Рынок программных продуктов предлагает множество сред для автоматизации программирования, самыми популярными из которых являются MS Access, Visual C++, Borland Delphi, Borland C++ Builder. В большинстве случаев языки программирования нельзя сравнивать между собой без связи с решаемыми задачами. Кратко рассмотрим их особенности:

1) MS Access — реляционная СУБД корпорации Microsoft. Имеет широкий спектр функций, включая связанные запросы, сортировку по разным полям, связь с внешними таблицами и базами данных. Благодаря встроенному языку VBA, в самом Access можно писать приложения, работающие с базами данных. В MS Access используется язык программирования Visual Basic for Applications (VBA), существенным недостатком которого является невозможность создания выполняемых файлов (.EXE). В плане поддержки целостности данных Access отвечает только моделям БД небольшой и средней сложности. В отношении защиты информации и разграничения доступа Access не имеет надежных стандартных средств.

2) Visual C++ — хотя в названии и фигурирует заявка на «визуальность», не позволяет напрямую создавать визуально-графические компоненты. Их приходится создавать с помощью классов, набирая код на клавиатуре, что способствует совершению множества ошибок и увеличивает время, затраченное не на разработку программ, а на создание интерфейса. В тоже время Visual C++ позволяет создавать программы минимального размера по сравнению с RAD-системами.

3) Borland Delphi и Borland C++ Builder — следующий шаг в развитии RAD-систем. Выглядят совершенно одинаково. Основное отличие состоит в том, что Delphi базируется на языке Pascal, а Builder на языке C++, более мощном и богатом по своим возможностям. Borland C++ Builder достаточно устойчиво занимает свою нишу, это обусловлено меньшей требовательностью к аппаратным ресурсам при разработке приложений, большей легкостью в освоении и применении средств системы для разработки приложений различной степени сложности. Borland C++ Builder позволяет создавать приложения с помощью инструментальных программных средств, визуально подготавливать, а также непосредственно писать SQL-запросы к базам данных.

Принимая во внимание все выше изложенное и связи с тем, что язык Borland C++ Builder 5 для меня является более знакомым языком программирования, в качестве среды разработки выбираем Borland C++ Builder версии 5.

### **1.3 Выбор технологии программирования**

Традиционная технология программирования складывалась в условиях, когда основными потребителями программ были научные учреждения, вычислительные ресурсы были ограничены. Кризис программного обеспечения привел к новой технологии программирования, которая была названа структурным программированием.

Главное требование, которому должна удовлетворять программа — работать в полном соответствии с поставленной задачей и адекватно реагировать на любые действия пользователя. Кроме этого, программа должна быть выпущена точно к заявленному сроку и допускать оперативное внесение необходимых изменений и дополнений. Объем занимаемой памяти

и эффективность алгоритмов при этом, к сожалению, отходят на второй план. Иными словами, современные критерии качества программы — это, прежде всего, надежность, а также возможность точно планировать производство программы и ее сопровождение. Для достижения этих целей программы должна иметь простую структуру, быть хорошо читаемой и легко модифицируемой. Структурное программирование — это технология создания программ, позволяющая путем соблюдения определенных правил уменьшить время разработки и количество ошибок, а также облегчить возможность модификации программы. Структурный подход охватывает все стадии разработки проекта: спецификацию, проектирование, собственно программирование и тестирование.

Система Builder производства корпорации Borland полностью удовлетворяет этим требованиям. Интегрированная среда Borland C++ Builder 5 обеспечивает скорость визуальной разработки, продуктивность повторно используемых компонент в сочетании с мощностью языковых средств C++, усовершенствованными инструментами и разномасштабными средствами доступа к базам данных.

К несомненным достоинствам Borland Builder можно отнести следующие особенности:

1) Интегрированная среда разработки объединяет редактор форм, инспектор объектов, палитру компонент и полностью интегрированные редактор кода и отладчик — инструменты быстрой разработки программных приложений, обеспечивающие полный контроль над кодом и ресурсами.

2) Профессиональные средства языка C++ интегрированы в визуальную среду разработки. Borland Builder предоставляет быстродействующий компилятор с языка Borland C++, эффективный инкрементальный загрузчик и гибкие средства отладки как на уровне исходных инструкций, так и на уровне ассемблерных команд — в расчете удовлетворить высокие требования программистов-профессионалов.

3) Конструирование по способу "drag-and-drop " позволяет создавать

приложение простым перетаскиванием захваченных мышью визуальных компонент из Палитры на форму приложения. Инспектор объектов предоставляет возможность оперировать со свойствами и событиями компонент, автоматически создавая заготовки функций обработки событий, которые наполняются кодом и редактируются в процессе разработки.

4) Мастер инсталляции руководит созданием унифицированных дистрибутивных пакетов для разработанных приложений.

5) Открытые инструменты API могут быть непосредственно интегрированы в визуальную среду системы. Есть возможность подключения привычного текстового редактора или создания собственного мастера для автоматизации выполнения повторяющихся процедур.

Все приведенные выше достоинства среды C++ Builder говорят в пользу его использования при написании пакета программ.

В качестве используемых пользователем операционных систем были выбраны системы семейства Windows, поскольку именно они чаще всего используются для работы.



## **2. Разработка интерфейса программы**

### **2.1 Разработка общего интерфейса**

Borland C++ Builder 5 является визуальной средой программирования. Вся работа производится в Интегрированной Среде Программирования (IDE — Integrated DevElopment). Особенностью IDE является то, что проектируемый графический интерфейс сразу отображается на экране и разработчик имеет возможность сразу видеть результат своей разработки. Во время разработки будем руководствоваться следующими основными принципами построения интерфейса:

а) соответствия ожиданиям пользователя. Если интерфейс соответствует этим ожиданиям, им очень удобно пользоваться;

б) простота и ясность интерфейса. Простой и ясный интерфейс не отвлекает пользователя от решения ключевых задач;

в) интуитивно понятный и знакомый пользователю интерфейс. Пользователь должен догадаться о выполнении какой-либо задачи без необходимости специального обучения. Если с самого начала пользователь не сможет разобраться в программе, он почти наверняка откажется от нее;

г) возможность настройки. Такие простые возможности, как выбор цвета могут оказать существенное влияние на восприятие интерфейса пользователем;

д) справочная служба. Для предотвращения попадания пользователя в затруднительную ситуацию нужно предусмотреть один из способов:

- организовать службу поддержки;
- предложить достаточно полную документацию.

IDE предоставляет в распоряжение разработчика формы, на которых размещаются компоненты. Обычно это оконная видимая форма, на которую переносятся пиктограммы компонентов.

При разработке программы данного дипломного проекта использовались следующие формы.

Основные:

- 1) форма «Каталог стран»;
- 2) форма «Каталог туров»;
- 3) форма «Бронирование туров»;

Вспомогательные:

- 1) форма «Добавление / редактирование сведений о турах»;
- 2) форма «Поиск»
- 3) форма «О клиентах»;
- 4) форма «Подбор тура»;
- 5) форма «Запрос на отбор данных по дате»;
- 6) форма «Запрос на отбор данных по цене»;
- 7) форма «Отчет о бронирование туров»;
- 8) форма «Туристическая путевка».

## **2.2 Форма «Каталог стран»**

На форме «Каталог стран» расположено:

- таблица стран, с кратким описанием и фотографией страны;
- добавление / редактирование / удаление страны и сведений о ней;
- быстрый поиск названия страны в списке стран;

на вкладке «Поиск»:

- быстрый поиск туров по предложенным критериям;

на вкладке «Операции»:

- кнопка «Уровень продаж за период»;
- кнопка «Уровень продаж по странам»;
- кнопка «Прибыль за период»;
- кнопка «Количество продаж за период»;
- кнопка «Маршруты со скидкой»;
- кнопка «Популярные маршруты».

Просмотр сведений о стране осуществляется на вкладке «О стране».

Для того чтобы быстро найти интересующую страну, нужно задать ее название в поле быстрого поиска.

На форме организована возможность быстрого поиска подходящего тура. Для этого необходимо в правой нижней части формы на вкладке «Поиск» ввести параметры для поиска и нажать кнопку «Найти». В соответствии с введенными параметрами будут отображены все записи о турах находящиеся в наборе данных и выведены на экран монитора в виде списка.

Кнопка «Справка» предлагает пользователю ознакомиться со справкой по работе с программой.

Кнопка «Продолжить» открывает форму «Каталог туров» со списком туров соответствующих выбранной стране.

По нажатию кнопки «Выход» происходит окончание работы программы.

## **2.3 Форма «Каталог туров»**

Форма «Каталог туров» отражает перечень всех туров для выбранной

страны.

На форме «Каталог туров» расположено:

- главное меню;
- поиск по критериям;
- упорядочивание данных по виду тура;
- фильтрация данных по отелю;
- отбор туров по количеству мест с указанием даты;
- кнопка «Заказать»;
- кнопка «Динамика продаж»;
- кнопка «Другой выбор».

Главное меню формы «Каталог туров» включает в себя следующие пункты:

1) Редактирование:

- добавить / редактировать сведения о туре. При выборе этого пункта, открывается вспомогательная форма «Добавление / редактирование сведений о туре»;
- удалить тур — удаление выбранного тура с предварительным сообщением на удаление;

2) Функции:

- подобрать тур. При выборе этого пункта, открывается вспомогательная форма «Подбор тура», где необходимо указать параметры для отбора тура;
- туры в указанный диапазон цен. Выбрав данный пункт, открывается вспомогательная форма «Запрос на отбор данных по цене», где пользователю для отбора туров предлагается указать начальную и конечную стоимость тура;

- туры в указанный диапазон дат. При выборе этого пункта открывается вспомогательная форма «Запрос на отбор данных по дате», где пользователю для отбора туров предлагается указать начальную и конечную даты;

- сведения о туристах. При выборе этого пункта открывается форма «О туристах», где представлены сведения о туристах забронировавших тур;

- постоянные клиенты;

- горящие туры;

- не востребоваанные туры.

3) Справка включает в себя:

- справку по работе с базой данных;

- справку о программе;

- об авторе.

Кнопка «Другой выбор» открывает форму «Каталог стран».

Кнопка «Заказать» открывает форму «Бронирование туров», где пользователю предлагается ввести сведения о клиентах желающих забронировать выбранный тур.

## **2.4 Форма «Бронирование туров»**

Общий вид формы «Бронирование туров» представлен на рисунке 1.

**ЗАКАЗ ТУРА**

Страна: Австрия | Тур: Мюнхен-Зальцбург-Вена | Вид тура: Автобусный тур | Размещение: Best Western Hotel 4\* | Дата звонка: 26.06.2009 | Длительность тура: 10 | Цена: 51 629,00р.

**Данные о клиентах**

Добавить | Изменить | Сохранить | Отменить | Удалить

№ ЗАКАЗА: | Возраст: |  
 Фамилия: | Пол: |  
 Имя: | день: | месяц: | год: |  
 Отчество: |

Загранпаспорт  
 №: | серия: | действителен до: 31.03.2009

Контактная информация  
 Тел.: |

№ заказа	Фамилия	Имя	Отчество	Возраст	Пол	День	Месяц	Год
52-113	Павлова	Василия	Сергеевна	взр	ж	17	июл	1969
52-114	Светикова	Галина	Матвеевна	реб	ж	10	июл	1983
52-125	Прокопцов	Степан	Доромедович	взр	м	25	ноб	1959
52-01	Селепина	Наталья	Сергеевна	взр	ж	21	июл	1979

**Бронь билета**

Тип транспорта: Автобус | Номер: 5-789  
 Пункт отправления: СПб, Пулково 2 | Пункт назначения: Мюнхен  
 Дата отправления: 19.04.2009 | Дата прибытия: 19.04.2009  
 Время отправления: 0:00:00 | Время прибытия: 0:00:00

Калькулятор  
 1: | руб.  
 2: | руб.  
 3: | руб.  
 4: | руб.  
 Рассчитать | Сбросить  
 Общая стоимость тура: | руб.

Оформить туристическую путевку  
 Список клиентов с учетом брони билетов  
 Отчет  
 Сохранить в файл  
 Справка

Новый билет | Бронировать | Изменить бронь | Удалить бронь | Отмена

Рисунок 1

Кнопки «Добавить», «Изменить», «Сохранить», «Удалить» — соответственно, добавляют, редактируют, сохраняют и удаляют сведения о клиентах.

В правой части формы находится калькулятор для расчета общей стоимости тура для группы клиентов.

Кнопка «Каталог туров» открывает форму «Каталог туров», которая предоставляет возможность выбора нового тура.

При нажатии кнопки «Оформить туристическую путевку» выводится отчет, где отражаются данные о выбранной стране, о выбранном туре, сведения о клиенте, о брони билета.

При нажатии кнопки «Отчет» выводится отчет, который предоставляет информацию обо всех забронированных маршрутах со сведениями о клиентах. Данный отчет предоставляется туристическому агенту для дальнейшей работы с клиентами.

При нажатии кнопки «Справка» — выводится справочная информация

по работе с программой.

Кнопки «Новый билет», «Бронировать», «Удалить бронь», «Изменить бронь» служат, соответственно, для введения данных о новом билете, занесения данных о забронированном билете в базу данных, удаления брони билета, редактирования данных брони билета.

## 2.5 Форма «Добавление / редактирование сведений о турах»

Форма «Добавление / редактирование сведений о турах» представлена на рисунке 2.

Рисунок 2

Форма разделена на 2 части. Левая часть — добавление / редактирование сведений о туре, правая часть — работа с данными таблицы «Отели».

При заполнении поля «Код отеля» правой части формы достаточно ввести код выбранного отеля, и в поле «Наименование отеля» название отеля отобразится автоматически (технология LookUp подробно рассмотрена в пункте 4.2).

Кнопка «Новая запись» служит для добавления нового тура и сведений о нем.

Кнопка «Отменить» служит для отмены всех действий и возвращает на форму «Каталог туров».

Кнопка «Сохранить и выйти» служит для занесения сведений о туре в базу данных и открытия формы «Каталог туров».

Для управления набором данных таблицы «Отели» правой части формы используется навигатор, который внешним видом похож на мультимедийный проигрыватель. Навигатор содержит кнопки, обеспечивающие выполнение различных операций с набором данных.

В правой части формы организована возможность быстрого поиска наименования отеля.

## **2.6 Форма «О клиентах»**

Форма «О клиентах» представлена на рисунке 3.



Номер заказа	Фамилия	Имя	Отчество	Пол	Год	Месяц	Год
52113	Павлова	Варвара	Сергеевна	Ж	17	май	1960
52114	Сидорова	Ольга	Михайловна	Ж	21	июл	1970
52125	Приказов	Олег	Дмитриевич	М	25	июл	1959
52123	Харченко	Антон	Геннадиевич	М	21	июл	1970
52115	Иванов	Макс	Сергеевич	М	9	фев	1996
52116	Маслова	Анна	Геннадиевна	Ж	20	июл	1962

Загранпаспорт  
 № паспорта: 45706 серия: 4000-21456 действителен до: 30.07.2010

Контактная информация  
 Тел.: 4547656

Тип транспорта: Автобус  
 Пункт отправления: СПб Пулково 2  
 Дата отправления: 10.07.2009  
 Время отправления: 7:20:00

Номер: 0451 5621  
 Пункт назначения: Вена Аэро  
 Дата прибытия: 10.07.2009  
 Время прибытия: 9:20:00

Справка ОК

Рисунок 3

Форма «О клиентах» отражает сведения обо всех клиентах, которые забронировали тур.

Кнопка «Справка» служит для вызова справочной информации по работе с программой.

Кнопка «ОК» служит для закрытия формы «О клиентах» и открытия формы «Каталог туров».

## 2.7 Форма «Подбор тура»

Форма «Подбор тура» представлена на рисунке 4.

Рисунок 4

В форме «Подбор тура» предлагается ввести параметры для отбора сведений о турах.

Кнопка «Найти» служит для нахождения сведений о турах согласно введенным параметрам.

Кнопка «Отменить» служит для отмены всех действий.

## 2.8 Форма «Запрос на выборку в диапазоне дат»

Форма «Запрос на выборку по дате» представлена на рисунке 5.

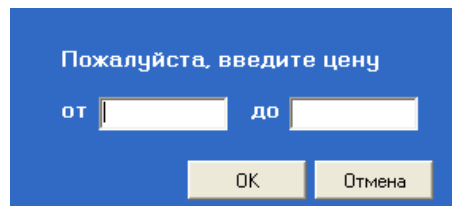
Рисунок 5

В данной форме необходимо указать начальную и конечную даты и нажать кнопку «ОК».

Кнопка «Отмена» отменяет все действия.

## 2.9 Форма «Запрос на выборку в диапазоне цен»

Форма «Запрос на выборку по цене» представлена на рисунке 6.



Пожалуйста, введите цену

от  до

ОК Отмена

Рисунок 6

В данную форму необходимо ввести начальное и конечное значение цены и нажать кнопку «ОК».

Кнопка «Отмена» отменяет все действия.

## 2.10 Форма «Отчет о бронировании туров»

Форма «Отчет о бронировании туров» представлена на рисунке 7.

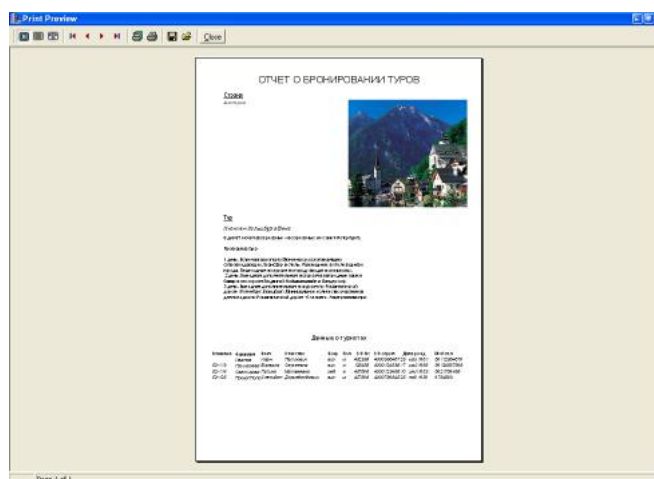


Рисунок 7

Для того чтобы сформировать «Отчет о бронировании туров» необходимо нажать кнопку «Отчет» на форме «Бронирование туров». Отчет содержит сведения о наименовании страны, наименовании тура, описании тура, данные о клиентах забронировавших тур.

## 2.11 Форма «Туристическая путевка»

Форма «Туристическая путевка» представлена на рисунке 8.

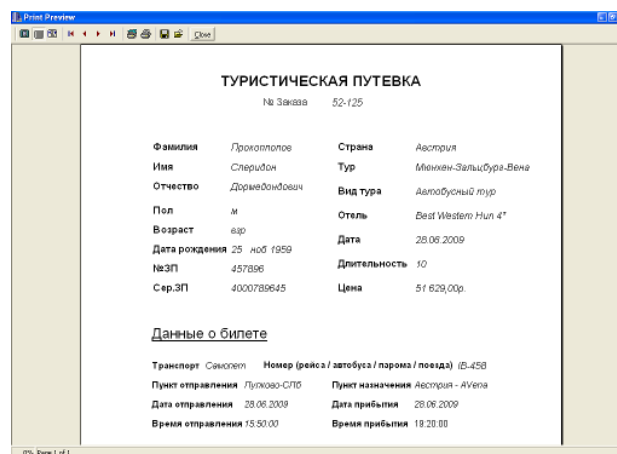


Рисунок 8

Для того чтобы оформить туристическую путевку для клиента, необходимо нажать кнопку «Оформить туристическую путевку» на форме «Бронирование туров».

Форма «Туристическая путевка» представляет собой отчет, в котором отображаются сведения о клиенте, сведения о выбранном туре и сведения о забронированном билете. Данный отчет необходим каждому клиенту, а так же агенту туристической формы для дальнейшей работы с клиентом.

## **2.12 Компоненты интерфейса**

Основой для создания пользовательского интерфейса является форма. В процессе разработки на нее добавляют соответствующие компоненты, задавая их расположение и размер. Компоненты представляют собой код, скомпилированный для выполнения определенных операций и избавляющий разработчика от необходимости создавать его заново.

Компоненты Borland Builder делятся на два вида: визуальные – видимые во время разработки и работы программы, и невидимые – видимые только во время разработки.

В программе для создания интерфейса были использованы следующие компоненты:

### **Визуальные**

- TEdit (окно редактирования) — служит для отображения, редактирования и ввода небольших однострочных текстов.

- TLabel (метка) — служит для отображения текста, не изменяемого пользователем.

- TPanel (панель) — контейнер для группирования органов управления

и других контейнеров. Также используется для построения полос состояния палитры инструментов и т.д.

- TComboBox — отображает список строк в развернутом виде или в виде выпадающего списка, позволяет пользователю выбрать из списка необходимую строку или задать в качестве выбора собственный текст.

- TUpDown — компонент представляет собой счетчики. С помощью UpDown создаются спаренные кнопки, с помощью которых прокручивают некоторый компонент, который связывается с данными.

- TRadioButton — радиокнопка, компонент используется, для выбора одной из взаимоисключающих альтернатив.

- TBitBtn — это управляющая кнопка, на поверхности которой можно располагать изображение.

- TPageControl — компонент позволяет построить набор страниц, которые друг друга перекрывают и которые можно перелистывать. Но главная ценность в том, что на TPageControl можно помещать другие компоненты, тем самым, расширяя возможности формы.

- TDateTimePicker — данный компонент обеспечивает возможность ввода даты или времени в редактируемое прямоугольное поле.

- TBDDGrid — компонент создает в форме таблицу для отображения данных из таблицы базы данных.

- TDBEdit — представляет собой окно редактирования, которое может отображать и редактировать поле набора данных.

- TDBImage — компонент предназначен для вывода изображений, содержащихся в графических полях базы данных.

- TDBMemo — позволяет отображать и редактировать данные поля, в частности, типа MEMO и BLOB.

– TDBText — позволяет отображать данные некоторого поля текущей записи набора данных, но не дает возможности их редактировать. Поле может быть различного типа: символьное, числовое, булево. Преобразование значения поля в строку текста, отображаемую в TDBText, производится автоматически.

– TQuickRep — представляет собой основу отчета на которой размещаются другие компоненты. Отчет состоит из отдельных полос — составных частей отчета, которые определяют содержание и вид созданного документа.

– TQRLabel — надпись, содержащая текст (аналог надписи Label), может размещаться на любой полосе.

– TQRDBText — значение поля записи, обычно размещается в полосе данных.

– TQRDBRichEdit — обеспечивает быстрый доступ к полям базы данных.

– TQRDBImage — размещается в полосе данных и отображает рисунок из поля таблицы базы данных.

– TDBNavigator — используется для управления набором данных. Навигатор содержит кнопки, обеспечивающие выполнение различных операций с набором данных путем автоматического вызова соответствующих методов.

– TDBLookupComboBox — компонент, предназначенный для выбора значения из выпадающего списка. Компонент связывается с полем «своего» набора данных через инспектр объектов.

#### Невизуальные

– DataSource — невизуальный компонент источника данных — обеспечивает интерфейс между компонентом набора данных и средствами

визуализации.

- TTable — обеспечивает прямой доступ к каждой записи и полю в одной указанной таблице базы данных. Компонент может также работать с подмножеством записей внутри таблицы базы данных. Во время проектирования вы можете создавать, удалять, модифицировать, или переименовывать таблицу базы данных, связанную с Table.

- Query — представляет собой набор данных, записи которого формируются в результате выполнения SQL-запроса и основаны на реляционном способе доступа к данным. Запрос включает в себя команды на языке SQL и выполняются при открытии набора данных с помощью вызова его методов.

- TOpenPictureDialog — вызывает стандартные диалоги Windows открытия и сохранения файлов изображений.

- TImageList — список, использующийся для хранения набора изображений одинаковых размеров, на которые можно ссылаться по индексам.

- MainMenu — позволяет конструировать и создавать на форме полосу главного меню, а также сопутствующие выпадающие меню.

- TQRBand — это контейнеры для помещения в них различных данных; заголовка, верхнего и нижнего колонтитулов страницы, верхнего и нижнего колонтитулов группы данных, компонентов, через которые и выводятся данные, и т.д.



### 3. Технический проект

#### 3.1 Разработка технического проекта

В соответствии с внешними спецификациями, был определен набор функций, необходимых для решения поставленной задачи.

Приложения C++ Builder 5 могут осуществлять доступ к локальным и удаленным базам данных с использованием механизма BDE. BDE представляет собой совокупность динамических библиотек и драйверов, обеспечивающих доступ к данным. Приложение через BDE передает запрос к базе данных, а обратно получает требуемые данные.

Для работы с таблицами баз данных при проектировании приложения используем программу Database Desktop, которая позволяет:

- создавать таблицы;
- изменять структуры;
- редактировать записи.

В начале создания новой таблицы в окне Create Table программы Database Desktop выбирается ее формат. Затем, определяется структура таблицы, в которой выполняются следующие действия:

- описание полей;
- создание ключа;
- задание индексов;
- определение ограничений на значение полей;
- определение условий (ограничений) ссылочной целостности;
- задание паролей;
- задание языкового драйвера;

– задание таблицы для выбора значений.

После определения структуры таблицы сохраняем ее в файл с расширением db. Таблицы базы данных располагаются на диске и являются физическими объектами. Для операции с данными, содержащими в таблицах, используются наборы данных — которые представляют собой совокупность записей, взятых из одной или нескольких таблиц базы данных.

Базовые возможности доступа к базе данных обеспечивает класс TDataSet, представляющий наборы данных в виде совокупности строк и столбцов (записей и полей). Этот класс содержит основные средства навигации (перемещения) и редактирования наборов данных.

При конструировании формы невизуальные компоненты, используемые для доступа к данным через механизм BDE, такие как DataSource, Table, размещаются в форме, но при выполнении приложения эти компоненты не видны. Поэтому их можно размещать в любом удобном месте формы, которая для них является контейнером — модулем. Для размещения невизуальных компонентов, через которые осуществляется доступ к данным, предназначен объект — модуль данных. Модуль данных позволяет:

- отделить управление базой данных от обработки данных;
- создать модуль, совместно используемый несколькими приложениями;

Основным назначением модуля данных является централизованное хранение компонентов доступа к данным. Использование модуля данных несколькими приложениями позволяет ускорить разработку приложений, т. к. готовый модуль данных впоследствии можно включать в новые приложения. Кроме того, управление базой данных через модуль дает возможность определить для всех пользователей одинаковые режимы и правила работы с базой, а так же делает более простым изменение этих режимов и правил.

При разработке программы дипломного проекта рассматривался простой модуль данных, который представлен объектом DataModule — DataModulTurAg на котором размещены компоненты доступа к данным —

TVibStrani, DataSVibStrani,

TTuri, DataSTuri,

TZakaz, DataSZakaz,

TLookUpOtely, DataSLookUpOtely,

THotelLookUp, DataSHotelLookUp,

TBilet, DataSBilet.

При обращении к содержащимся в модуле данных компонентам для них указывается составное имя, в которое, кроме имени компонента, входит имя модуля данных.

```
DataModulTurAg->TTuri->Refresh();
```

Кроме стандартных событийных функций используемых компонентов для решения поставленной задачи в программе используются следующий основной набор функций:

1) добавление данных. Добавление данных происходит при использовании метода Insert() — вставляет новую пустую запись в таблицу, и делает ее активной. После вставки пустой записи в ее поля можно вводить данные.

```
void __fastcall TForm2::InsertTurClick(TObject *Sender)
```

```
{
```

```
DataModulTurAg->TTuri->Insert();
```

```
FormDobav->Tag=0;
```

```
FormDobav->ShowModal();
```

```

if (FormDobav->Tag)

{

    DataModulTurAg->TTuri->Refresh();

}

}

```

2) редактирование данных. Редактирование происходит при помощи метода Edit() — разрешает редактирование данных в таблице.

```

void __fastcall TFormBrony::BitEditClick(TObject *Sender)

{

    DataModulTurAg->TZakaz->Edit();

    DataModulTurAg->TZakaz->FieldByName("Zakaz")->AsString=EZakaz->Text;

}

```

3) удаление данных. Удаление данных происходит при помощи метода Delete() — удаляет активную запись и устанавливает указатель записи на следующую запись таблицы.

```

DataModulTurAg->TVibStrani->Delete();

```

4) сохранение данных. Сохранение данных рассмотрено на примере добавления / редактирования сведений о туре. В случае добавления / редактирования сведений о туре открывается форма «Добавление / редактирование сведений о туре», где пользователю предлагается ввести новые данные, либо отредактировать имеющиеся в таблице базы данных.

```

void __fastcall TFormDobav::BSaveAndExitClick(TObject *Sender)

{

```

```

DataModulTurAg->TTuri->Edit();

DataModulTurAg->TTuri->FieldByName("VidTura")-
>AsString=CBVidTura->Text;

DataModulTurAg->TTuri->FieldByName("Hotel")->AsString=

DataModulTurAg->THotelLookUp->FieldByName("NameHotel")-
>AsString;

DataModulTurAg->TTuri->Post();

}

```

5) организация быстрого поиска. Поиск в таблице осуществляется при помощи функции `Locate()`. Эта функция позволяет находить запись в таблице по совокупности значений ее полей и найденную запись делает активной, тем самым открывая доступ к ней. В примере, приведенном ниже, организация поиска представлена на примере поиска по наименованию страны

```

void __fastcall TForm3::BitBNaytiClick(TObject *Sender)

{ //Организация поиска методом Locate поиск по названию страны

TLocateOptions Options;

Options.Clear();

if (!DataModulTurAg->TVibStrani->Locate("Strana",Edit2-
>Text,Options))

ShowMessage ("Такой страны не найдено" + Edit2->Text);

}

```

б) вычисление текущей даты

```

void __fastcall TForm3::FormShow(TObject *Sender)

{

```

```

int n;

AnsiString stDay[7] = {"воскресенье", "понедельник", "вторник", "среда",
"четверг", "пятница", "суббота"};

AnsiString stMonth[12] = {"января", "февраля", "марта", "апреля",
"мая", "июня", "июля", "августа", "сентября",
"октября", "ноября", "декабря"};

TDateTime Todey;

Word Year, Month, Day;

Todey = Now();

DecodeDate(Todey, Year, Month, Day);

Label8->Caption = "Сегодня: " + IntToStr(Day) + " " + stMonth[Month-1]
+ " " +
IntToStr(Year) + " года, " + stDay[DayOfWeek(Todey) - 1];

```

7) вычисление min стоимости тура из списка туров для указанной страны.

Для технической реализации данной функции используем компонент Query с параметром, где вместо параметра указывается название страны. Компонент Query представляет собой набор данных, записи которого формируются в результате выполнения SQL-запроса. Текст запроса, на основании которого в набор данных отбираются записи, содержится в свойстве SQL типа TStrings. Запрос включает в себя команды на языке SQL и выполняется при открытии набора данных с помощью вызова его методов ExecSQL или Open.

SQL-текст

```
SELECT min(Cena)
```

FROM TabTurov

WHERE (Strana=:Strana)

Builder–код

```
void __fastcall TForm3::FormShow(TObject *Sender)
{
    DataModulTurAg->TVibStrani->Refresh();

    DataModulTurAg->TTuri->Refresh();

    DataModulTurAg->TVibStrani->First();

    for (int i=0;i<DataModulTurAg->TVibStrani->RecordCount;i++)
    {
        QMin->Close();

        QMin->Params->Items[0]->AsString=DataModulTurAg->TVibStrani
        ->FieldByName("Strana")->AsString;

        if (QMin->Prepared)

            QMin->Prepare();

        QMin->Open();

        DataModulTurAg->TVibStrani->Edit();

        DataModulTurAg->TVibStrani->FieldByName("Stoim")
        ->AsCurrency=QMin->FieldByName("MIN OF Cena")->AsCurrency;

        DataModulTurAg->TVibStrani->Post();

        DataModulTurAg->TVibStrani->Next();

    }
```

```
DataModulTurAg->TVibStrani->First();

}
```

8) вычисление общего числа туров для страны.

Для технической реализации данной функции используем компонент Query с параметром, где вместо параметра указывается название страны. Вычисление производим на форме «Каталог туров», а затем передаем на форму «Каталог стран».

SQL–текст

```
SELECT COUNT(*)

FROM TabTurov

WHERE Strana=:Strana
```

Builder–код

```
void __fastcall TForm2::FormShow(TObject *Sender)

{

    QTurov->Close();

    QTurov->Params->Items[0]->AsString=

DataModulTurAg->TVibStrani->FieldName("Strana")->AsString;

    QTurov->Open();

    DataModulTurAg->TVibStrani->Edit();

    DataModulTurAg->TVibStrani->FieldName("Putev")

->AsInteger=QTurov->FieldName("COUNT(*)")->AsInteger;

    DataModulTurAg->TVibStrani->Post();
```



```
DataModulTurAg->TTuri->Refresh();
```

```
DataModulTurAg->TVibStrani->Refresh();
```

```
//-----
```

```
void __fastcall TForm3::FormShow(TObject *Sender)
```

```
{
```

```
int n;
```

```
DataModulTurAg->TVibStrani->First();
```

```
DataModulTurAg->TTuri->First();
```

```
for (int i=0;i<DataModulTurAg->TVibStrani->RecordCount;i++)
```

```
{
```

```
for (int j=0;j<DataModulTurAg->TTuri->RecordCount;j++)
```

```
{
```

```
n=DataModulTurAg->TVibStrani->FieldByName("Putev")->AsInteger;
```

```
DataModulTurAg->TVibStrani->Edit();
```

```
DataModulTurAg->TVibStrani->FieldByName("Putev")->AsInteger=n;
```

```
DataModulTurAg->TVibStrani->Post();
```

```
DataModulTurAg->TVibStrani->Next();
```

```
}
```

```
}
```

```
DataModulTurAg->TVibStrani->First();
```

```
}
```

9) фильтрация данных.

```

void __fastcall TForm2::BitVypClick(TObject *Sender)
{
    AnsiString s=0,t=0;

    DataModulTurAg->TTuri->Cancel();

    s=DataModulTurAg->TVibStrani->FieldByName("Strana")->AsString;
    t=DataModulTurAg->TTuri->FieldByName("Strana")->AsString;

    DataModulTurAg->TTuri->Filtered=false;

    if (s == t)
    {
        DataModulTurAg->TTuri->Filtered=true;

        DataModulTurAg->TTuri->Filter="((KolMest>="+EKol1
        ->Text)+")and(KolMest<="+EKol2->Text)+
        ")and((DataZaezda>="+DateToStr(DTData1
        ->Date)+")and(DataZaezda<="+DateToStr(DTData2->Date)+"))";
    }

    else return;

}

```

10) подбор тура.

```

void __fastcall TForm2::NParamClick(TObject *Sender)
{
    FormPodborTura->Tag=0;

    FormPodborTura->ShowModal();
}

```

```

if (FormPodborTura->Tag)

{

DataModulTurAg->TTuri->Filtered=false;

DataModulTurAg->TTuri->Filter=

"(((DataZaezda>='"+DateToStr(FormPodborTura->DData1->Date)+

"')AND(DataZaezda<='"+DateToStr(FormPodborTura->DData2

->Date)+

"'))AND((KolDney>='"+StrToInt(FormPodborTura->Edit4

->Text)+"')AND(KolDney<='"+

StrToInt(FormPodborTura->Edit3

->Text)+"'))AND((Cena>='"+CurrToStr(FormPodborTura->Edit6

->Text)+

"')AND(Cena<='"+CurrToStr(FormPodborTura->Edit5->Text)+"'))";

DataModulTurAg->TTuri->Filtered=true;

}

QTurov->Close();

QTurov->Params->Items[0]->AsString=DataModulTurAg->TVibStran

i->FieldByName("Strana")->AsString;

QTurov->Open();

DataModulTurAg->TTuri->Refresh();

}

```

11) сведения о туристах забронировавших тур.

Для технической реализации данной функции используем компонент Query с параметром, где вместо параметра указывается название тура. Запрос формируется на форме «Каталог туров», результаты запроса передаются на форму «О клиентах».

SQL–текст

SELECT \*

FROM TabBrony

WHERE Tur=:Tur

Builder–код

```
void __fastcall TFormZapOTuristah::FormShow(TObject *Sender)
```

```
{
```

```
    QueryOTuristah->Close();
```

```
    QueryOTuristah->Params->Items[0]->AsString=DataModulTurAg->TTuri  
->FieldName("Tur")->AsString;
```

```
    if (! QueryOTuristah->Prepared)
```

```
    {
```

```
        QueryOTuristah->Prepare();
```

```
    }
```

```
    QueryOTuristah->Open();
```

```
    if (QueryOTuristah->RecordCount!=0)
```

```
        DataSOTuristah->DataSet=QueryOTuristah;
```

```
    else {ShowMessage ("По вашему запросу ничего не найдено");
```

```
    return;
```

```
}
```

```
Edit1->Text=IntToStr(DataModulTurAg->TZakaz->RecordCount);
```

```
}
```

12) вычисление свободного количества путевок.

Для того чтобы реализовать данную функцию сначала необходимо определить общее число клиентов забронировавших тур. Для этого используем компонент Query. Затем, рассчитываем свободное количество путевок путем вычитания количества путевок тура из общего числа клиентов забронировавших тур. Набор данных Query, в отличие от компонента Table, может включать в себя более чем одной таблицы базы данных.

SQL-текст

```
SELECT COUNT(*)
```

```
FROM TabBrony
```

```
WHERE Tur=:Tur
```

Builder-код

```
void __fastcall TForm2::FormShow(TObject *Sender)
```

```
{
```

```
DataModulTurAg->TTuri->Refresh();
```

```
DataModulTurAg->TTuri->First();
```

```
for (int i=0;i<DataModulTurAg->TTuri->RecordCount;i++)
```

```
{
```

```
Form2->QueryTurist->Close();
```

```
Form2->QueryTurist->Cancel();
```

```

Form2->QueryTurist->Params->Items[0]->AsString=
DataModulTurAg->TTuri->FieldByName("Tur")->AsString;
if (Form2->QueryTurist->Prepared)
Form2->QueryTurist->Prepare();
Form2->QueryTurist->Open();
DataModulTurAg->TZakaz->Edit();
DataModulTurAg->TZakaz->FieldByName("Vsego")->AsInteger=
Form2->QueryTurist->FieldByName("COUNT(*)")->AsInteger;
DataModulTurAg->TTuri->Edit();
DataModulTurAg->TTuri->FieldByName("Ost") ->AsInteger=
DataModulTurAg->TTuri->FieldByName("KolMest")->AsInteger
– DataModulTurAg->TZakaz->FieldByName("Vsego")->AsInteger;
DataModulTurAg->TTuri->Post();
DataModulTurAg->TTuri->Next();
}
DataModulTurAg->TTuri->First();
}

```

### 13) калькуляция расчетов для группы клиентов.

Если необходимо рассчитать тур для группы клиентов, предлагается калькуляция расчетов. Пользователю предлагается выбрать для каждого из группы клиентов возрастной уровень (взрослый / ребенок от 5–10 лет / ребенок от 11–16 лет). Уровень взрослый предполагает оплату стоимости тура в полном объеме, уровень ребенок от 5–10 лет предполагает оплату стоимости тура со скидкой 20%, уровень ребенок от 11–16 лет предполагает

оплату стоимости тура со скидкой 5%. Организация калькуляции расчетов представлена на рисунке 9.

Рисунок 9 — Калькуляция расчетов

```
void __fastcall TFormBrony::BitRaschetClick(TObject *Sender)
{
    Currency Summ;

    Summ=KonCena1+KonCena2+KonCena3+KonCena4;

    ECenaSumm->Text=Summ;
}

//-----

void __fastcall TFormBrony::CBox1Click(TObject *Sender)
{
    cena=0,cenaPr20=0,cenaPr5=0;
```

```

ECena2->Text=

DataModulTurAg->TTuri->FieldByName("Cena")->AsCurrency;

switch (CBox1->ItemIndex)

{

case 0:{ cena=StrToCurr(ECena2->Text);

ECena2->Text=cena;break; }

case 1:{ cena=StrToCurr(ECena2->Text);

cenaPr20=cena-((cena*20)/100);

ECena2->Text=cenaPr20; break; }

case 2:{ cena=StrToCurr(ECena2->Text);

cenaPr5=cena-((cena*5)/100);

ECena2->Text=cenaPr5;break; }

}

KonCena1=ECena2->Text;

}

//-----

void __fastcall TFormBrony::CBox2Click(TObject *Sender)

{

cena=0,cenaPr20=0,cenaPr5=0;

ECena3->Text=

DataModulTurAg->TTuri->FieldByName("Cena")->AsCurrency;

switch (CBox2->ItemIndex)

```



```

{
case 0:{ cena=StrToCurr(ECena3->Text);
      ECena3->Text=cena;break; }

case 1:{ cena=StrToCurr(ECena3->Text);
      cenaPr20=cena-((cena*20)/100);
      ECena3->Text=cenaPr20; break;}

case 2:{ cena=StrToCurr(ECena3->Text);
      cenaPr5=cena-((cena*5)/100);
      ECena3->Text=cenaPr5;break;}

}

KonCena2=ECena3->Text;

}

//-----

void __fastcall TFormBrony::CBox3Click(TObject *Sender)
{
cena=0,cenaPr20=0,cenaPr5=0;

ECena4->Text=

DataModulTurAg->TTuri->FieldByName("Cena")->AsCurrency;

switch (CBox3->ItemIndex)
{
case 0:{ cena=StrToCurr(ECena4->Text);
      ECena4->Text=cena;break; }

```

```

case 1:{ cena=StrToCurr(ECena4->Text);

cenaPr20=cena-((cena*20)/100);

ECena4->Text=cenaPr20; break ;}

case 2:{ cena=StrToCurr(ECena4->Text);

cenaPr5=cena-((cena*5)/100);

ECena4->Text=cenaPr5;break;}

}

KonCena3=ECena4->Text;

}

//-----

void __fastcall TFormBrony::CBox4Click(TObject *Sender)

{

cena=0,cenaPr20=0,cenaPr5=0;

ECena5->Text=

DataModulTurAg->TTuri->FieldByName("Cena")->AsCurrency;

switch (CBox4->ItemIndex)

{

case 0:{ cena=StrToCurr(ECena5->Text);

ECena5->Text=cena;break; }

case 1:{ cena=StrToCurr(ECena5->Text);

cenaPr20=cena-((cena*20)/100);

ECena5->Text=cenaPr20; break;}

```

```

case 2:{ cena=StrToCurr(ECena5->Text);

cenaPr5=cena-((cena*5)/100);

ECena5->Text=cenaPr5;break;}

}

KonCena4=ECena5->Text;

}

//-----

void __fastcall TFormBrony::BitBtn4Click(TObject *Sender)

{

DataModulTurAg->TZakaz->Cancel();

ECena2->Clear();

ECena3->Clear();

ECena4->Clear();

ECena5->Clear();

ECenaSumm->Clear();

}

```

14) поиск тура с главной формы «Каталог стран».

Для того, чтобы осуществить поиск тура или туров из набора данных, воспользуемся параметрическим запросом на языке SQL. Результатом неизвестных параметров в запросе являются значения критериев вводимых с формы.

Компонент Query с текстом SQL размещен на модуле DataModulTurAg, а вызов осуществляется с формы «Каталог стран» при нажатии кнопки «Найти». Результатом является перечень туров, отобранных в соответствии с

указанными критериями, на форме «Каталог туров».

На рисунке 10 представлена организация ввода параметров для выполнения поиска.

Рисунок 10 — Организация поиска

SQL–текст

SELECT \*

FROM TabTurov

WHERE (Strana=:Strana) AND

(DataZaezda BETWEEN :Data1 AND :Data2) AND

(KolDney BETWEEN :Dn1 AND :Dn2) AND

(Cena BETWEEN :Cena1 AND :Cena2)

Builder–код

```
void __fastcall TForm3::BNaytiClick(TObject *Sender)
```

```
{
```

```
DataModulTurAg->QSort2->Close();
```

```
DataModulTurAg->QSort2->Params->Items[0]->AsString=Edit3->Text;
```

```
DataModulTurAg->QSort2->Params->Items[1]->AsDateTime=Date1-
>Date;
```

```
DataModulTurAg->QSort2->Params->Items[2]->AsDateTime=Date2-
>Date;
```

```
DataModulTurAg->QSort2->Params->Items[3]->AsInteger=StrToInt(Edit4-
>Text);
```

```
DataModulTurAg->QSort2->Params->Items[4]->AsInteger=StrToInt(Edit5-
>Text);
```

```
DataModulTurAg->QSort2->Params->Items[5]-
>AsCurrency=StrToCurr(Edit6->Text);
```

```
DataModulTurAg->QSort2->Params->Items[6]-
>AsCurrency=StrToCurr(Edit11->Text);
```

```
DataModulTurAg->QSort2->Open();
```

```
if (DataModulTurAg->QSort2->RecordCount!=0)
```

```
DataModulTurAg->DataSTuri->DataSet=DataModulTurAg->QSort2;
```

```
else { ShowMessage ("По Вашему запросу ничего не найдено");
```

```
return;
```

```
}
```

```
Form2->Show();
```

```
}
```

15) программирование кнопки «Оформить туристическую путевку».

Формирование отчета для кнопки «Оформить туристическую путевку» происходит после нажатия кнопки на форме «Бронирование туров».

Builder–код

```

void __fastcall TFormBrony::BKartaTuristaClick(TObject *Sender)
{
    DataModulTurAg->QKartaTurista->Close();

    DataModulTurAg->QKartaTurista->Params->Items[0]->AsString=
        DataModulTurAg->TZakaz->FieldByName("Zakaz")->AsString;

    DataModulTurAg->QKartaTurista->Open();

    FormKartaTurista->QRKartaTurista->Preview();

    FormKartaTurista->QRKartaTurista->Print();

}

```

В процессе программирования в контейнер DataModulTurAg был добавлен компонент Query и в SQL был сформирован запрос.

SQL-текст

```

“SELECT a.Strana, a.Tur,a.VidTura,a.Hotel,a.DataZaezda,a.KolDney,
a.Cena,c.Fam,c.Imy,c.Otchestvo,c.Pol,c.Vozr,c.Den,c.Mes,c.God,
c.ZPnom,c.ZPser,b.Zakaz, b.Nomer, b.PunktOtprav, b.PunktNaznach,
b.DataOyprav,b.DataPrib, b.VremOtprav, b.VremNaznach, b.Transport
FROM TabReys b, TabTurov a

LEFT OUTER JOIN TabBrony c ON ((a.Tur=c.Tur) AND
(b.Zakaz=c.Zakaz))

WHERE (c.Zakaz=:Zakaz)”

```

Далее формируется отчет и в режиме предварительно просмотра выводится на экран. На отдельной форме разрабатывается отчет, компоненты которого подключаются к Query через инспектор объектов. Форма разработки отчета представлена на рисунке 11

**Туристическая путевка**

№ Заказа [Zakaz]

Фамилия [Fam] Страна [Strana]

Имя [Imy] Тур [Tur]

Отчество [Otchestvo] Вид тура [VidTura]

Пол [Pol] Отель [Hotel]

Возраст [Vozr] Дата [DataZaezda]

Дата рождения [DenMesGod]

№ЗП [ZPnom] Длительность [KolDney]

Сер.ЗП [ZPser] Цена [Cena]

Данные о билете

Транспорт [Transport] Номер (рейса / автобуса / парома / поезда) [Nomer]

Пункт отправления [PunktOtprav] Пункт назначения [PunktNaznach]

Дата отправления [DataOyprav] Дата прибытия [DataPriib]

Время отправления [VremOtprav] Время прибытия [VremNaznach]

Рисунок 11

16) формирование отчета «Отчет о бронировании туров».

На форме разрабатывается отчет, элементы которого, через инспектор объектов, подключаются к соответствующим компонентам таблиц базы данных (рисунок 12).

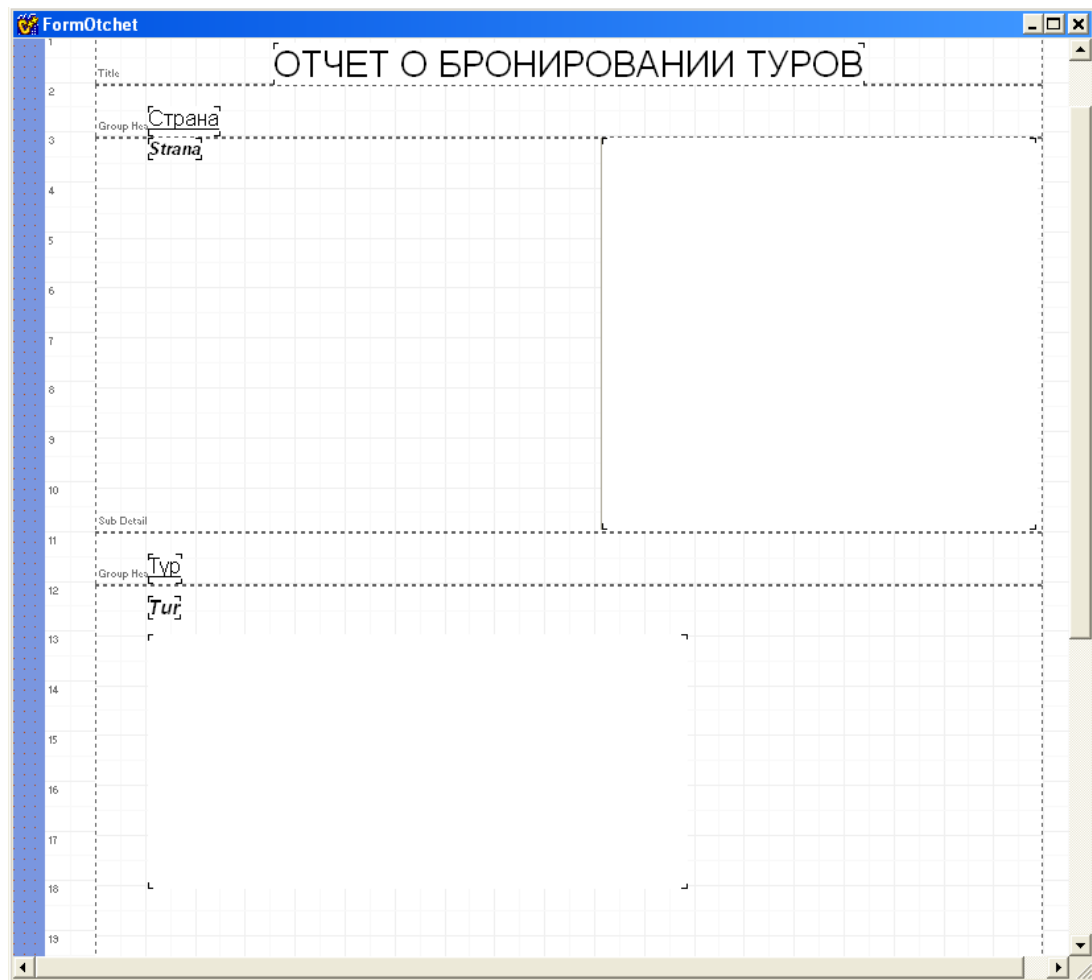


Рисунок 12 — Форма разработки «Отчет о бронировании туров»

#### Builder-код

```
void __fastcall TFormBrony::BitOtchetClick(TObject *Sender)
{
    FormOtchet->QROtchet->Preview();
    FormOtchet->QROtchet->Print();
}
```