Министерство образования и науки Российской Федерации

Государственное образовательное учреждение

Высшего профессионального образования

**«ОРЕНБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

Факультет информационных технологий

Кафедра информационных систем и технологий

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовому проекту**

по дисциплине «Управление данными»

База Данных «Пассажирские железнодорожные перевозки»

ГОУ ОГУ 230201.4007.01

Руководитель проекта

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Извозчикова С.А.

"\_\_\_\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2009г.

Исполнители

студенты гр. ИСТ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Антипова Т.В.

"\_\_\_\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2009г.

Оренбург 2009

**Содержание:**

Введение

1 Аналитическая часть

1.1 Основные задачи предметной области

1.2 Технологический процесс сбора и обработки информации

2 Проектная часть

2.1 Информационное обеспечение комплекса задач

2.2 Характеристика входной информации

2.3 Характеристика выходной информации

2.4 Характеристика структуры данных

3 Описание программного обеспечения

3.1 Выбор и обоснование инструментальных средств

3.2 Описание программных модулей и их взаимодействие

3.3 Описание интерфейса пользователя

Заключение

Список использованных источников

**Введение**

Поскольку базы данных, вне всяких сомнений, занимают в настоящее время лидирующее положение в области информационных технологий, они становятся неотъемлемой частью в любой сфере деятельности, а особенно в обработке большого объёма информации.

Для того, чтобы обработка данных не стала новой проблемой был создан стандарт структурированного языка запросов SQL (Structured Query Language).

Программное обеспечение, работающее с использованием SQL, функционирует незаметно для пользователя, предоставляя ему необходимую основу для манипуляции информацией.

Сейчас особое внимание уделяется на использование сетевых информационных технологий, базирующихся на архитектуре клиент – сервер, и переходе от малопонятных внутренних процедур к содержательной работе с предметной областью, которую описывают хранимые данные. Поэтому предпочтение отдаётся СУБД, которая поддерживает многопользовательский доступ MySQLServer.

Все более актуальным становится вопрос об автоматизации продажи билетов на пассажирские поезда. Функциональность базы данных во многом зависит от предоставленной создателям информации о предметной области. Большую роль играет создание интерфейса доступного для любого пользователя, работающего с ней.

В соответствии с поставленной целью в первом разделе рассматривается общие сведения и задачи, а также функции предметной области.

Далее рассматривается технологический процесс сбора и обработки информации создаваемой базы данных. Все входные и выходные информационные потоки, которые надо использовать в работе.

В заключительном разделе описывается созданная база данных, её интерфейс, программные модули и технические требования для её использования на персональном компьютере.

**1 Аналитическая часть**

* 1. **Основные задачи предметной области**

Перевозка пассажиров на железнодорожном транспорте является актуальной проблемой на сегодняшний день, так как продажа билетов ведётся хоть и автоматизировано, но зачастую у касс скапливается большое количество народу. То есть идёт медленное обслуживание пассажиров происходит это из-за неудобного интерфейса. Где требуемые данные находятся на 3-ем, 4-ом и более уровнях доступа, что увеличивает время доступа к информации.

Перед нами стоит задача упростить доступ к данным, то есть сделать его более доступным в том плане, что информация, необходимая для продажи билетов пассажирам находилась максимум на 3-ем уровне доступа.

Наша база данных помогает облегчить поиск информации, требуемой пасса-жиру и самому оператору, по продажи билетов:

- выдать полную информацию о требуемом рейсе;

а) название рейса;

б) код рейса;

в) время отправления;

г) время прибытия до пункта назначения;

д) время в пути;

- предоставить информацию о наличие:

а) свободных мест;

б) стоимости билета по типу вагона:

1) плацкарт;

2) купе;

в) стоимость страхового сбора на время поездки;

В последнее время, люди часто интересуются о стоимости страхового полиса на время поездки. Особенно, такую информацию просят предоставить пассажиры дальних рейсов.

**1.2 Технологический процесс сбора и обработки информации**

Представим себе работу вокзалов, имеющее в наличии большое количество отдалённых касс, и целый штат операторов по продаже билетов.

В основной круг задач работников касс при оформлении проездных документов входят следующие задачи:

* сбор информации о пассажире;
* выбор нужной информации о необходимом рейсе;
* обработка информации о данном рейсе;
* формирование выходных документов.

При сборе и обработке информации большое внимание должно уделяться функции ведения отчётности о наличии билетов. Отчётность – это внутренняя информация, на основе которой ведутся все расчеты и выводы в работе. В данном случае отчётными данными для операторов будут являться следующие данные:

* список пассажиров с указанием фамилии, имя и отчества, количества детей, присутствующих с родителями. Для того, чтобы знать поток пассажиров в данный пункт, объём пассажиров в вагоне;
* список проданных билетов с указанием даты для расчета стоимости проезда;

Помимо этого, в системе расчета присутствует внутренний неизменяемый список поездов. Неизменяемый – потому что на протяжении многих лет список поездов практически не меняется, так как нарушится поток поездов и будет простой у железной дороги, поэтому будем считать его постоянным.

Таким образом, использование внутренней справочной информации обуславливается самим процессом учёта билетов.

Поэтому разработка и создание удобной автоматизированной системы для учёта билетов и потока пассажиров облегчит работу не только операторов, но и работников аналитического отдела ДЕПО. Компьютер выполняет любые арифметические операции любой сложности за доли секунды с безошибочной точностью, что является залогом успеха в любом вычислении. На основании этого создана упрощённая автоматизированная система, призванная помочь работнику не углубляться в большой поток информации, а выбирать только требуемую (информацию) в данный момент времени. Отобразим на рисунке 1 информационные потоки:

Внешняя среда

4

Администратор

3

1

Клиент

(пассажир)

Оператор

2

Рисунок 1 – Схема информационных потоков

1-запрос от пассажира оператору;

2-ответ оператору;

3-данные для базы данных;

4-распоряжения по замене или добавлению данных.

**2 Проектная часть**

**2.1 Информационное обеспечение комплекса задач**

Информация в базе данных должна быть представлена простым языком. Мы решили представить базу данных в виде модели «сущность-связь», которая также известна как ER-диаграмма. Такие модели удобны для понимания.

Каждой сущности присваиваются уникальное имя. Между сущностями могут существовать различные связи. Связь изображается линией, проводимой между сущностью-родителем и сущностью-потомком, с точкой на конце линии у сущности-потомка. Атрибуты сущности изображаются в виде списка имен внутри блока сущности. Атрибуты, определяющие первичный ключ, размещаются наверху списка и отделяются от других атрибутов горизонтальной чертой.

Проанализировав предметную область, мы выделили 6 сущностей: станции рейса, рейс, билет, тип вагона, станции, цены.

Обычно разработка модели базы данных состоит из двух этапов: составление логической модели и создание на ее основе физической модели. Средство разработки ER-диаграмм полностью поддерживает такой процесс, он имеет два представления модели: логическое и физическое. Таким образом, разработчик может строить логическую модель базы данных, не задумываясь над деталями физической реализации, т.е. уделяя основное внимание требованиям к информации и бизнес-процессам, которые будет поддерживать будущая база данных.

**2.2 Характеристика входной информации**

В данной системе вся необходимая входная информация вводится пользователем системы, то непосредственно оператором по продажи билетов, вручную с клавиатуры или с помощью мыши. Вся эта информация представлена в виде следующих информационных массивов:

- данные о пассажире (фамилия, имя, отчество), непосредственно отправляющегося в поездку;

- документ, удостоверяющий личность пассажира (номер, серия);

- информация о рейсе, на котором пассажир собирается выезжать (номер рейса, станции, время).

**2.3 Характеристика выходной информации**

Разработка данной программы ориентирована на формирование всего лишь одного результирующего выходного документа – проездного билета. В билете будут указаны:

1. номер билета (уникальный);
2. номер рейса;
3. время отправления;
4. время прибытия;
5. время стоянки;
6. дата отправления;
7. дата прибытия;
8. номер вагона;
9. тип вагона;
10. количество человек;
11. фамилия, имя, отчество пассажира;
12. вид предъявленного документа;
13. номер и серия предъявленного документа;
14. стоимость проезда (рассчитывается на основании расстояния, стоимости типа вагона);
15. стоимость страхового сбора на время движения поезда;

Билет действителен при наличии документа, удостоверяющего личность, только на время и дату отправления поезда, а также на время его движения.

* 1. **Характеристика структуры данных**

При создании программного продукта была использована база данных, состоящая из 5 таблиц.

1. в таблице 1 «Reis» ведется учет всех возможных рейсов с указанием их номера, который является уникальным и служит ключевым индексом для связи с другими таблицами, название рейса (начальная и конечная станции) и путь, то есть все станции, которые проходит поезд в порядке следования.
2. в таблице 2 «Allpoints», содержатся станции, из названий которых формируется маршрут рейса. Для уникальности используется поле “No”.
3. Таблица 3 «Tickets» состоит из 10 полей, поле «No\_ticket» является уникальным и кодируется датой и временем регистрации пассажира. В поле «Start point» заносится пункт отправления пассажира, который может и не совпадать со станцией начала пути. Аналогичным образом в поле «Endpoint» запоминается пункт прибытия пассажира. Поле «EndDate» вычисляемое, и зависит от времени, проведенном в пути.
4. В таблице 4 «Stationofreis», содержатся 7 полей, где указывается информация о времени между двумя смежными станциями, стоимость участка пути, время прибытия, отправления и стоянки.
5. В таблице 5 «TypeVag» отображаются, какие вагоны относятся к плацкар-тному классу вагонов, какие к купейному.
6. В таблице 6 «Price» является временной и используется для сохранения данных о билете заказа для последующего формирования билета.

Таблица 1 **-** Reis

Таблица 2 **–** Allpoints

Таблица 3 **-** Tickets

Stationeries

Таблица 5 **–** TypeVag

Таблица 6 – Price

**3 Описание программного обеспечения**

**3.1 Выбор и обоснование инструментальных средств**

Данная система создана с использованием технологии «файл-сервер». Разработанная база данных является локальной. Приложение написано с помощью средств разработки Delphi 7 и осуществляет многопользовательский режим доступа.

Delphi 7 – мощная система визуального объектно-ориентированного программмирования. С ее помощью даже начинающие программисты могут создавать оконные интерфейсы, удовлетворяющие стандартам Windows, причем очень быстро. Спектр отраслей, в которых можно применять Delphi, достаточно широк: инженерные, офисные, торговые и др. К тому же Delphi является одним из самых распространенных средств создания приложений баз данных. Простота языка и эффективность (небольшие размеры и высокая производительность) создаваемых с его помощью программ сделали Delphi незаменимым средством разработки приложений для доступа к базам данных. Delphi используется для генерирования и вывода на печать сложных отчетов баз данных, а также предоставляет возможность связываться из своего приложения с такими продуктами Microsoft, как Word, Excel и другие, используя их возможности в создании мощных систем помощи.

Занимает 16 Мб дискового пространства. Оперативная память должна быть 64 Мб минимум.

**3.2 Описание программных модулей и их взаимодействие**

Данный программный продукт состоит из 4 основных вычисляющих модулей:

а) поиск всех возможных рейсов, последовательно проходящих через заданные начальную и конечную точки маршрута.

Для каждого из существующих рейсов указываются станции, через которые поезд проходит последовательно, к примеру, Москва – Санкт-Петербург – Казань – Воронеж.

Следует отметить, что поезд может идти только в одном направлении (из Москвы в Воронеж).

Задаются пункт отправления и пункт прибытия, указанные пассажиром. Рейс считается подходящим, если в его маршруте содержатся пункты отправления и прибытия, причем пункт прибытия следует после пункта отправления. Например, если пассажир следует из Санкт-Петербурга в Воронеж, то указанный выше рейс считается подходящим. Если же пункт его отправления Казань, а пункт назначения Санкт-Петербург, то рейс №100 не будет соответствовать заданному условию.

Листинг программного модуля:

if (ComboBox1.Text <> '') and (ComboBox2.Text <> '') and (RzDateTimeEdit1.Text <> '') then

 begin

 Query1.Destroy;

 DS1.Destroy;

 Query1:=CreateQuery('select \* from reis');

 Query1.Open;

 nomera:='';

 Query1.First;

 while not Query1.Eof do

 begin

 m:=-1;

 n:=-1;

 trains:= TStringList.Create;

 trains.Text:= StringReplace(Query1.FieldValues['stancii'],'-',#13#10,[rfReplaceAll]);

 for i:=0 to trains.Count-1 do

 begin

 if trains[i] = ComboBox1.Text then m:=i;

 if trains[i] = ComboBox2.Text then n:=i;

 end;

 if (m <> -1) and (n <> -1) then

 if (m < n) then

 begin

 nomera:=nomera+Query1.Fields[0].AsString+',';

 end;

 trains.Destroy;

 Query1.Next;

 end;

 if nomera = '' then

 begin

 Application.MessageBox('Таких рейсов нет','Внимание',MB\_OK+MB\_ICONERROR);

 DS1:= CreateDataSource(Query1);

// DBGrid1.DataSource:= DS1;

 end else

 begin

 Delete(nomera,length(nomera),1);

 Query1.SQL.Clear;

 SQLString:='select \* from reis where no\_reisa in ('+nomera+')';

 Query1.SQL.Add(SQLString);

 Query1.Open;

 DS1:=CreateDataSource(Query1);

 DBGrid1.DataSource:= DS1;

 end;

 end;

Результатом работы программного модуля является список подходящих рейсов.

б) визуальное отображение всех занятых и свободных мест выбранного рейса с учетом даты отправления, проверка освободившихся мест.

После выбора необходимого рейса и даты отправления пассажира таблично отображаются занятые и свободные места. Для каждого занятого места запоминаются пункт назначения и дата прибытия. Место считается освободившимся для данного пассажира, если выполняется два условия: запомненный для места пункт назначения находится до пункта отправления пассажира или совпадает с ним; запомненная дата прибытия для места совпадает с датой отправления пассажира или дата отправления является более поздней.

Листинг программного модуля:

SQLString:= 'select \* from typevag where no\_reisa="'+DBGrid1.Fields[0].Text+'"';

 Query2:= CreateQuery(SQLString);

 Query2.Open;

 plackart:= TStringList.Create;

 kupe:= TStringList.Create;

 plackart.Text:=StringReplace(Query2.Fields[1].AsString,' ',#13#10,[rfReplaceAll]);

 kupe.Text:=StringReplace(Query2.Fields[2].AsString,' ',#13#10,[rfReplaceAll]);

 StringGrid1.RowCount:=plackart.Count+kupe.Count+1;

 for i:=0 to plackart.Count-1 do

 begin

 StringGrid1.Cells[0,StrToInt(plackart[i])]:=plackart[i]+' плацкарт';

 for j:=1 to 55 do StringGrid1.Cells[j,i]:='';

 end;

 for i:=0 to kupe.Count-1 do

 begin

 StringGrid1.Cells[0,StrToInt(kupe[i])]:=kupe[i]+' купе';

 // for j:=1 to 55 do StringGrid1.Cells[j,i]:='';

 for j:=37 to 55 do StringGrid1.Cells[j,StrToInt(kupe[i])]:='X';

 end;

 plackart.Destroy;

 kupe.Destroy;

 with Query2 do

 begin

 SQLString:='select \* from tickets where no\_reisa="'+DBGrid1.Fields[0].Text+'"and startdate="'+FormatDateTime('dd.mm',RzDateTimeEdit1.Date)+'"';

 SQL.Clear;

 SQL.Add(SQLString);

 Open;

 end;

 Query2.First;

 while not Query2.Eof do

 begin

 v:= Query2.Fields[8].AsInteger;

 p:= Query2.Fields[9].AsInteger;

 StringGrid1.Cells[p,v]:='\*';

 pnaz:= Query2.FieldValues['EndPoint'];

 SQLString:= 'select \* from stationofreis where No\_reisa="'+DBGrid1.Fields[0].AsString+'"';

 Query3:= CreateQuery(SQLString);

 Query3.Open;

 Query3.First;

 while Query3.Fields[1].AsString <> pnaz do

 begin

 if Query3.FieldByName('station').AsString = ComboBox1.Text then

 {if FormatDateTime() < FormatDateTime() then} goto m1;

 Query3.Next;

 end;

 Query3.Destroy;

 StringGrid1.Cells[p,v]:='';

 m1:

 Query2.Next;

 end;

 Query2.Destroy;

в) вычисление основных параметров для заданного маршрута: даты прибытия, времени в пути, стоимости билета.

Для каждого пункта маршрута в отдельной таблице указываются следующие параметры: стоимость участка пути от предыдущей до текущей станции и время его прохождения, время прибытия, стоянки и отправления. Для того, чтобы вычислить стоимость маршрута, по которому следует пассажир, нужно суммировать стоимости участков пути, составляющих данный маршрут. К цене маршрута добавляется наценка за класс вагона и обязательный страховой сбор. Чтобы вычислить время .проведенное в пути, нужно суммировать время прохождения каждого участка пути и время стоянки в каждом пункте. Дата прибытия вычисляется прибавлением времени в пути к дате отправления.

Листинг программного модуля:

 if (StringGrid1.Cells[ACol,ARow] <> '\*') and (StringGrid1.Cells[ACol,ARow] <> 'X') then

 begin

 if (ComboBox1.Text <> '') and (ComboBox2.Text <> '') and (RzDateTimeEdit1.Text <> '') then

 begin

 if MessageDlg('Приступить к гегистрации пассажира?',mtConfirmation,[mbYes,mbNo],0) = mrYes then

 begin

 Application.CreateForm(TFormTicket, FormTicket);

 with FormTicket do

 begin

 Label2.Caption:= DBGrid1.Fields[0].Text;

 Label3.Caption:= DBGrid1.Fields[2].Text;

 Label6.Caption:= FormatDateTime('dd.mm',RzDateTimeEdit1.Date);

 Label17.Caption:= IntToStr(ACol);

 Label15.Caption:= IntToStr(ARow);

 Label10.Caption:= ComboBox1.Text;

 Label11.Caption:= ComboBox2.Text;

 time:=TStringList.Create;

 time.Text:=StringReplace(StringGrid1.Cells[0,ARow],' ',#13#10,[rfreplaceall]);

 Label13.Caption:= time[1];

 time.Destroy;

 Label19.Caption:= FormatDateTime('dd.mm',RzDateTimeEdit1.Date);

 SQLString:='select \* from stationofreis where no\_reisa="'+DBGrid1.Fields[0].Text+'"';

 Query2:= CreateQuery(SQLString);

 Query2.Open;

 Query2.First;

 hour:=0;

 minute:=0;

 price:=0;

 s:='0';

 while not Query2.Eof do

 begin

 if Query2.Fields[1].AsString = ComboBox1.Text then

 begin

 Label21.Caption:=Query2.Fields[6].AsString;

 s1:= Query2.Fields[6].AsFloat;

 while not Query2.Eof do

 begin

 Query2.Next;

 s2:= Query2.Fields[4].AsFloat;

 if s1>s2 then

 begin

 Label6.Caption:= FormatDateTime('dd.mm',RzDateTimeEdit1.Date+1);

 s1:=s2;

 end;

 if Query2.Fields[1].AsString = ComboBox2.Text then

 begin

 if s = '' then

 begin

 s:= Query2.Fields[3].AsString;

 end;

 Label23.Caption:=Query2.Fields[4].AsString;

 s:= FloatToStr(sumhhmm(StrToFloat(s),Query2.Fields[3].AsFloat));

 Label25.Caption:= s;

 priceway:= price;

 GetDir(0,Path);

 with TIniFile.Create(Path+'\zd.ini') do

 try

 price:= price+Query2.Fields[2].AsFloat+StrToFloat(ReadString('cash',Label13.Caption,'no val ues'))+StrToFloat(ReadString('cash','сбор','no values'));

 finally

 Free;

 end;

 Label27.Caption:= FloatToStr(price);

 // Edit3.Text:= StringReplace(Label27.Caption,',','.',[rfReplaceAll]);

 {!!!!!добавить указание стоимости проезда!!!!!}

 goto m1;

 end;

 price:=price+Query2.Fields[2].AsFloat;

 s:= FloatToStr(sumhhmm(StrToFloat(s),Query2.Fields[3].AsFloat));

 s:= FloatToStr(sumhhmm(StrToFloat(s),Query2.Fields[5].AsFloat));

 s1:=s2;

 end;

 end;

 Query2.Next;

 end;

 m1:

 end;

 FormTicket.ShowModal;

 end;

 end;

 end;

г) регистрация пассажира и формирование проездного документа.

После заполнения данных пассажира (фамилии, имени, отчества, номера документа, удостоверяющего личность) происходит регистрация пассажира на данный рейс и формируется проездной документ, в котором указываются рейс, вычисленные ранее стоимость билета, время пути, время отправления и прибытия, дата прибытия, данные пассажира, пункт отправления и пункт следования.

Листинг программного модуля:

 if (Edit1.Text <> '') or (MaskEdit1.Text <> '') then

 begin

 dat:= FormatDateTime('ddmmyyhhmm',Now);

 SQLString:='insert into tickets values("'+FormatDateTime('ddmmyyhhmm',Now)+'","'+Edit1.Text+'","'+MaskEdit1.Text+'","'+Label2.Caption+'","'+Label10.Caption+'","'+Label11.Caption+'","'+Label19.Caption+'","'+Label6.Caption+'","'+Label15.Caption+'","'+Label17.Caption+'","'+Label13.Caption+'","'+StringReplace(Label27.Caption,',','.',[rfReplaceAll])+'")';

 if ExecuteSQL(SQLString) then

 begin

 GetDir(0,Path);

 with TIniFile.Create(Path+'\zd.ini') do

 try

 SQLString:= 'insert into prices values("'+Label21.Caption+'","'+Label23.Caption+'","'+Label25.Caption+'","'+StringReplace(ReadString('cash',Label13.Caption,'no values'),',','.',[rfReplaceAll])+'","'+StringReplace(ReadString('cash','сбор','no values'),',','.',[rfReplaceAll])+'","'+StringReplace(FloatToStr(priceway),',','.',[rfReplaceAll])+'")';

 finally

 Free;

 end;

 ExecuteSQL('delete from prices');

 if ExecuteSQL(SQLString) then

 begin

 // ShowMessage('Успешная регистрация');

 FormTicket.Close;

 SQLString:= 'select \* from tickets, prices where tickets.no\_ticket="'+dat+'"';

 QReport.SQL.Clear;

 QReport.SQL.Add(SQLString);

 QReport.Open;

 frReport1.LoadFromFile(Path+'\ticket.frf');

 frReport1.ShowReport;

 with FormPay do

 begin

 ComboBox1.Text:= '';

 ComboBox2.Text:= '';

 RzDateTimeEdit1.Text:= '';

 // RzToolButton1Click(sender);

 with StringGrid1 do

 begin

 RowCount:= 2;

 ColCount:=55;

 for i:=1 to ColCount-1 do

 begin

 ColWidths[i]:=20;

 Cells[i,0]:=IntToStr(i);

 end;

 end;

 end;

 end;

 end;

 end else ShowMessage('Не введены обязательные данные!');

**3.3 Описание интерфейса пользователя**

При запуске программы на выполнение на экране компьютера появляется следующее меню изображённое на рисунке 3.

Рисунок 3 – Главное окно

Меню состоит из пунктов «Продажа билетов», «Администратор». При выборе пункта меню «Продажа билетов» появляется следующее подменю, изображённое на рисунке 4.

Рисунок 4 – Форма продажи билетов

На форме указаны «Номер рейса», «Пункт отправления», «Пункт назначения», «Дата отправления» - это поля, которые заполняются оператором; поля «Рейсы», «Состав» - отображаются сразу после запроса. Как только информация отобразилась, оператор приступает к регистрации пассажира, рисунок 5.

Рисунок 5 – Форма регистрации пассажира

После регистрации идёт оформление документа, представленного на рисунке 6.

Рисунок 6 – Проездной документ

При выборе пункта меню «Администратор», необходимо ввести пароль. Так как обычный оператор не вправе что-либо менять или исправлять в самой базе данных. Вид окошка представлен на рисунке 7.

Рисунок 7 – Ввод пароля

Появляется форма администратора с вкладками «Станции», «Рейсы», «Стоимость класса вагона», «Поезда», «Стоимость участка пути», форма изображена на рисунке 8.

Рисунок 8 – Форма администратора (вкладка «Станции»)

При выборе команды «Рейсы» на экране появляется новое окно, отображённое на рисунке 9.

В данном окне администратор может добавлять новые рейсы, внося их номер и полный путь, а название рейса формируется автоматически – из начальной заданной станции и конечной прописанной в пути.

Рисунок 9 – Рейсы (добавление)

В списке, изображенных в окне, отражаются все операции, произведенные по каждому рейсу, будь то добавление, редактирование или удаление станций и его номера. Кнопка «Добавить» предназначена для добавления из списка станций нового маршрута, содержащего полный путь рейса. Если после какой-либо корректировки данных кнопка не будет нажата, рейс не будет добавлен. При нажатии на кнопку на кнопку «Отмена» происходит возврат в предыдущее окно без сохранения. После выбора вкладки «Стоимость класса вагона» появляется новое окно, изображённое на рисунке 10.

Рисунок 10 – Вкладка «Стоимость класса вагона»

Вносятся необходимые изменения, и нажимается кнопка «Применить».

Рассмотрим вкладку «Поезда», представленную на рисунке 10. Здесь отмечаются сколько вагонов находится в составе и под каким номером какие типы вагонов. После нажатия кнопки «Применить», информация заносится в таблицу.

Рисунок 11 – Вкладка «Поезда»

На вкладке «Стоимость участка пути», изображённой на рисунке 11, администратор вносит необходимые данные: время в пути, время прибытия, стоянка, время отправления, стоимость участка. После нажатия кнопки «Применить», вне-сённые данные отображаются в верхней таблице.

Рисунок 11 – Вкладка «Стоимость пути»

**Заключение**

Разработанный нами программный продукт является удобным и надежным средством для быстрой регистрации пассажиров для проезда на железнодорожном транспорте.

Программный продукт выполняет следующие функции:

- возможность заказа билета из любой точки мира при наличии подключения к внутренней компьютерной сети;

- возможность добавления и редактирования маршрутов, цен на билеты.

- заказ билета на любую дату.

- быстрый расчет времени в пути.

Возможные улучшения:

- выбор пассажиром места в вагоне (верхняя, нижняя полка)

- выдача рекомендаций по оптимальному с точки зрения времени, проведенному в пути и стоимости билета, маршруту между двумя пунктами.

**Список литературы:**

1. ГОСТ 2.105-95 ЕСКД – общие требования к текстовым документам;
2. ГОСТ 2.304-81 ЕСКД – шрифты чертёжные;
3. ГОСТ 2.106-68 ЕСКД – текстовые документы;
4. Общие требования и правила оформления выпускных квалификационных работ, курсовых проектов, отчетов ОГУ;
5. Программирование на Delphi 6, Глушаков С.В.