Министерство образования Калининградской области

государственное бюджетное учреждение Калининградской области

профессиональная образовательная организация

«Колледж информационных технологий и строительства»

(ГБУ КО ПОО «КИТиС»)

**Отчет по учебной практике**

УП.01 Разработка модулей программного обеспечения для компьютерных систем

по ПМ.01 Разработка модулей программного обеспечения для компьютерных систем

Специальность 09.02.07 «Информационные системы и программирование»

Сроки прохождения практики:

с «02» декабря 2020 г. по «22» декабря 2020 г.

Место практики ГБУ КО ПОО «КИТиС»

|  |  |
| --- | --- |
| Выполнил: | студент 3 курса,  группы ИС 18-2к  <Даниелян Гор Артурович>  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |
| Проверила: | Большакова-Стрекалова Анна Викторовна  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (оценка)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) |

Калининград

2020

Содержание

[Введение 3](#_Toc59316885)

[1 Первая неделя: решение задач 4](#_Toc59316886)

[1.1 Задача 1 (стр.50 с главы В) в 10 4](#_Toc59316887)

[1.2 Задача 2 (стр.87) в.10 7](#_Toc59316888)

[1.3 Задача 3 (стр.124) В.10 15](#_Toc59316889)

[1.4 Задача 4 (стр.145) В.10 17](#_Toc59316890)

[1.5 Задача 5 (стр.194) в 10 20](#_Toc59316891)

[2 Вторая неделя : разработка АИС 24](#_Toc59316892)

[Введение 24](#_Toc59316893)

[2. 1 Описание предметной области 25](#_Toc59316894)

[2. 2 Концептуальная модель предметной области 26](#_Toc59316895)

[2. 3 Диаграмма классов (ClassDiagram) 27](#_Toc59316898)

[2.4 Построение ER-диаграммы 29](#_Toc59316899)

[2.5 Диаграмма ER- экземпляров 30](#_Toc59316903)

[2.6 Построение диаграммы деятельности 34](#_Toc59316905)

[Заключение 39](#_Toc59316906)

[3 Третья неделя: Разработка мобильного приложения 40](#_Toc59316907)

[3.1 Подробное описание 40](#_Toc59316908)

[3.2 Ход выполнения задания 41](#_Toc59316909)

[4 Сайт и оценка эффективности 43](#_Toc59316910)

[4.1 Ход выполнения задания 43](#_Toc59316911)

[4.2 Результаты работы 45](#_Toc59316912)

[Заключение 46](#_Toc59316913)

[Список использованных источников 47](#_Toc59316914)

# Введение

Учебная практика является этапом теоретического обучения и подготовки студентов к профессиональной деятельности в различных организациях.

Цель учебной практики – приобретение основных навыков практической работы по специальности – "Информационные системы и программирования", применения современных средств вычислительной техники, систематизации и закрепление теоретических знаний, формирование информационной базы.

Целью данной работы является подробный отчет о выполнении задании, данных на практику.

В ходе учебной практики мною были выдвинуты следующие задачи:

- Решение задач по вариантам, учебник «Java - Методы программирования»

1 Первая неделя: решение задач

1.1 Задача 1 (стр.50 с главы В) в 10

Условие

Вычислить норму матрицы

Таблица   идентификаторов

|  |  |  |
| --- | --- | --- |
| Наименование | Тип Данных | Назначение |
| max | Integer | Выходная |

Блок Схема

Вычеслить max матрицы

Вычеслить min матрицы

Да дддддддд нет

return max

return min;

Код программы

import java.util.Arrays;  
import java.util.Random;  
public class Main {  
 final static Random *rand* = new Random();  
 public static void main(String[] args) {  
  
 int [][]arr = *createArray*(7,8,-100,100);  
 *toConsole*(arr);  
 System.*out*.println("Norm ||A||L ="+*maxRows*(*sumRows*(arr)));  
 System.*out*.println("Norm ||A||m ="+*maxCols*(*sumCols*(arr)));  
 }  
 public static int[][] createArray(int N,int M,int a,int b)  
 {  
 int [][]arr = new int[N][M];  
 for(int i = 0;i<N;i++)  
 {  
 for(int j = 0;j<M;j++)  
 {  
 arr[i][j] = *rand*.nextInt((b - a) + 1) + a;  
 }  
 }  
 //System.out.println(Arrays.deepToString(arr));  
 return arr;  
  
 }  
  
 public static int[] sumRows(int[][]arr)  
 {  
 int []array = new int[arr.length];  
 for(int i = 0;i<arr.length;i++)  
 {  
 for(int j = 0;j<arr[0].length;j++)  
 {  
 array[i]+=Math.*abs*(arr[i][j]);  
 }  
 }  
 System.*out*.println(Arrays.*toString*(array));  
 return array;  
  
 }  
  
 public static int[] sumCols(int [][]arr)  
 {  
 int[]array = new int[arr[0].length];  
 for(int j = 0;j<arr[0].length;j++)  
 {  
 for(int i =0;i<arr.length;i++)  
 {  
 array[j]+=Math.*abs*(arr[i][j]);  
 }  
 }  
 System.*out*.println(Arrays.*toString*(array));  
 return array;  
  
 }  
  
 public static void toConsole(int [][]arr)  
 {  
 for (int[] ints : arr) {  
 for (int anInt : ints) {  
 System.*out*.print(anInt + " ");  
 }  
 System.*out*.println();  
 }  
 }  
  
 public static int maxRows(int []arr)  
 {  
 int max = 0;  
 for(int val:arr) {  
 if(val > max)max = val;  
 }  
 return max;  
 }  
  
 public static int maxCols(int []arr)  
 {  
 int min = arr[0];  
 for(int val:arr) {  
 if(val < min) min=val;  
 }  
 return min;  
 }  
}

Выполнение

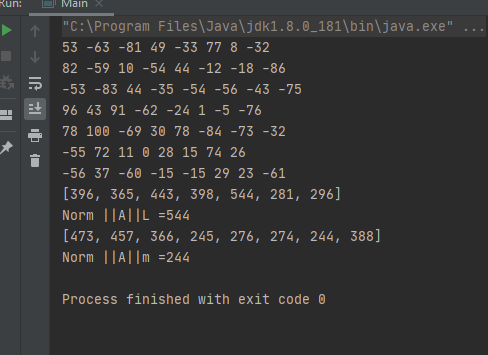


Рисунок 1 – Результат работы задачи 1 (стр.50 с главы В)

### 1.2 Задача 2 (стр.87) в.10

Условие

Создать классы, спецификации которых приведены ниже. Определить конструкторы и методы setТип(), getТип(), toString(). Определить дополнительно методы в классе, создающем массив объектов. Задать критерий выбора данных и вывести эти данные на консоль. В каждом классе, обладающем информацией, должно быть объявлено несколько конструкторов.

Train: Пункт назначения, Номер поезда, Время отправления, Число мест (общих, купе, плацкарт, люкс). Создать массив объектов. Вывести:

a) список поездов, следующих до заданного пункта назначения;

b) список поездов, следующих до заданного пункта назначения и отправляющихся после заданного часа;

c) список поездов, отправляющихся до заданного пункта назначения и имеющих общие места.

Таблица   идентификаторов

|  |  |  |
| --- | --- | --- |
| Наименование | Тип Данных | Назначение |
| destination  number  timeOfDeparture  seatCountHolder | String  String  String  Integer | Входная  Входная Входная  Выходная |

Блок схема

Вычеслить сколько мест в вагоне

нет

Да

Вычеслить направлене поезда

Свободных мест нету

Количество свободных мест в Разных классах

Null

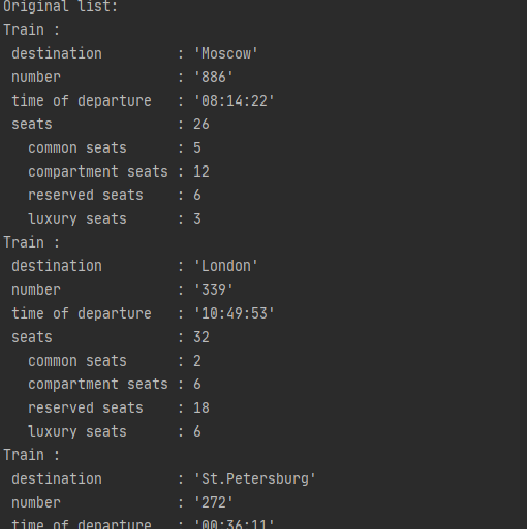
Код программы

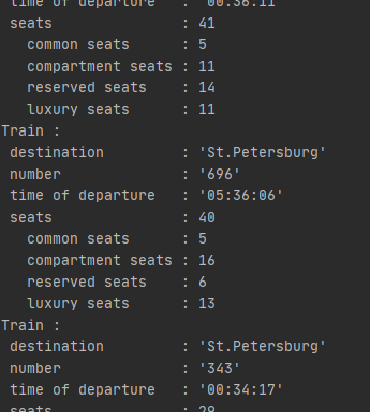
public class Main {  
 public static void main(String[] args) {  
 List<Train> trainsList = Trains.*newRandomTrainList*(10);  
  
 System.*out*.println("Original list:");  
 trainsList.forEach(System.*out*::println);  
  
 List<Train> trainsWithGivenDestination = Trains.*findTrainsWithSameDestination*(trainsList, "Moscow");  
 System.*out*.println("Trains for Moscow:");  
 trainsWithGivenDestination.forEach(System.*out*::println);  
  
 List<Train> trainsWithGivenDestinationAndTime = Trains  
 .*findTrainsWithSameDestinationAndAfterTime*(trainsList, "Paris", "09:00:00");  
 System.*out*.println("Trains for Paris and 09:00:");  
 trainsWithGivenDestinationAndTime.forEach(System.*out*::println);  
  
 List<Train> trainsWithGivenDestinationAndSeats = Trains  
 .*findTrainsWithSameDestinationAndHaveCommonSeats*(trainsList, "Ekaterinburg");  
 System.*out*.println("Trains for Ekaterinburg and seats:");  
 trainsWithGivenDestinationAndSeats.forEach(System.*out*::println);  
 }  
}

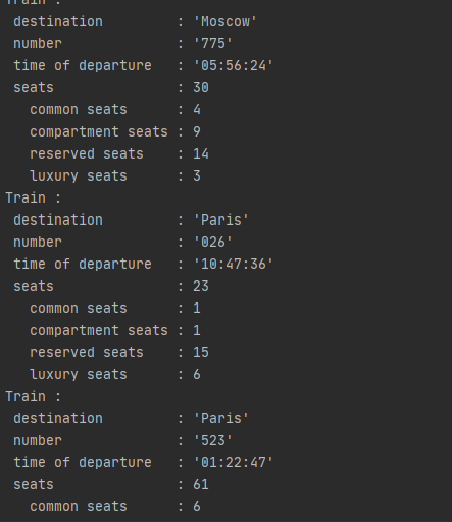
public class Train {  
 private String destination;  
 private String number;  
 private String timeOfDeparture;  
  
 private EnumMap<CarType, Integer> seatCountHolder;  
  
 public enum CarType {  
 *COMMON*, *COMPARTMENT*, *RESERVED\_SEAT*, *LUXURY*;  
 }  
  
 public Train(String destination, String number, String timeOfDeparture,  
 final int commonSeatsCount, final int compartmentSeatsCount,  
 final int reservedSeatCount, final int luxurySeatCount ) {  
 this.destination = destination;  
 this.number = number;  
 this.timeOfDeparture = timeOfDeparture;  
  
 this.seatCountHolder = new EnumMap<CarType, Integer>(CarType.class);  
  
 seatCountHolder.put(CarType.*COMMON*, commonSeatsCount);  
 seatCountHolder.put(CarType.*COMPARTMENT*, compartmentSeatsCount);  
 seatCountHolder.put(CarType.*RESERVED\_SEAT*, reservedSeatCount);  
 seatCountHolder.put(CarType.*LUXURY*, luxurySeatCount);  
 }  
  
 public String getDestination() {  
 return destination;  
 }  
  
 public void setDestination(String destination) {  
 this.destination = destination;  
 }  
  
 public String getNumber() {  
 return number;  
 }  
  
 public void setNumber(String number) {  
 this.number = number;  
 }  
  
 public String getTimeOfDeparture() {  
 return timeOfDeparture;  
 }  
  
 public void setTimeOfDeparture(String timeOfDeparture) {  
 this.timeOfDeparture = timeOfDeparture;  
 }  
  
 public int getSeatCountByCarType(final CarType type) {  
 return seatCountHolder.get(type);  
 }  
  
 public void setSeatCountByCarType(final CarType type, final int newCount) {  
 seatCountHolder.put(type, newCount);  
 }  
  
 public int getTotalSeatCount() {  
 int count = 0;  
  
 for (CarType type : CarType.*values*())  
 count = count + seatCountHolder.get(type);  
  
 return count;  
 }  
  
 @Override  
 public String toString() {  
 return "Train :\n" +  
 " destination : '" + destination + "\'\n" +  
 " number : '" + number + "\'\n" +  
 " time of departure : '" + timeOfDeparture + "\'\n" +  
 " seats : " + getTotalSeatCount() + "\n" +  
 " common seats : " + seatCountHolder.get(CarType.*COMMON*) + "\n" +  
 " compartment seats : " + seatCountHolder.get(CarType.*COMPARTMENT*) + "\n" +  
 " reserved seats : " + seatCountHolder.get(CarType.*RESERVED\_SEAT*) + "\n" +  
 " luxury seats : " + seatCountHolder.get(CarType.*LUXURY*);  
 }  
}

public class Trains {  
 private static final String[] *CITIES* = {"Moscow", "St.Petersburg", "Ekaterinburg", "Paris", "London"};  
  
 private static final ThreadLocalRandom *RNG* = ThreadLocalRandom.*current*();  
  
  
 public static List<Train> findTrainsWithSameDestination(final Collection<Train> trains, final String destination) {  
 List<Train> result = new ArrayList<>(trains);  
 return result.stream()  
 .filter(train -> Objects.*equals*(train.getDestination(), destination))  
 .collect(Collectors.*toList*());  
 }  
  
 public static List<Train> findTrainsWithSameDestinationAndAfterTime(final Collection<Train> trains,  
 final String destination,  
 final String givenTime) {  
 List<Train> result = new ArrayList<>(trains);  
 return result.stream()  
 .filter(train -> Objects.*equals*(train.getDestination(), destination))  
 .filter(train -> DatatypeConverter.*parseTime*(train.getTimeOfDeparture()).after(  
 DatatypeConverter.*parseTime*(givenTime)))  
 .collect(Collectors.*toList*());  
 }  
  
 public static List<Train> findTrainsWithSameDestinationAndHaveCommonSeats(final Collection<Train> trains,  
 final String destination) {  
 List<Train> result = new ArrayList<>(trains);  
 return result.stream()  
 .filter(train -> Objects.*equals*(train.getDestination(), destination))  
 .filter(train -> train.getSeatCountByCarType(Train.CarType.*COMMON*) > 0)  
 .collect(Collectors.*toList*());  
 }  
  
 public static List<Train> newRandomTrainList(final int trainCount) {  
 List<Train> trains = new ArrayList<>(trainCount);  
  
 for (int i = 0; i < trainCount; i++)  
 trains.add(*newRandomTrain*());  
  
 return trains;  
 }  
  
 public static Train newRandomTrain() {  
 final String city = *randomCity*();  
 final String number = *randomNumber*();  
 final String time = *randomTime*();  
  
 Train train = new Train(city, number, time, 0, 0, 0, 0);  
 for (Train.CarType type : Train.CarType.*values*())  
 train.setSeatCountByCarType(type, *RNG*.nextInt(21));  
 return train;  
 }  
  
 private static String randomCity() {  
 return *CITIES*[*RNG*.nextInt(*CITIES*.length)];  
 }  
  
 private static String randomTime() {  
 return String.*format*("%02d:%02d:%02d", *RNG*.nextInt(13), *RNG*.nextInt(60), *RNG*.nextInt(60));  
 }  
  
 private static String randomNumber() {  
 return String.*format*("%03d", *RNG*.nextInt(1000));  
 }  
}

Выполнение







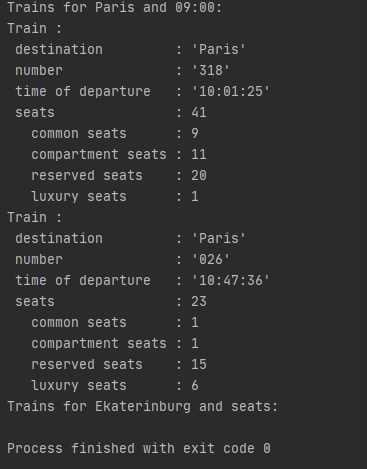


Рисунок 2 – Результат работы Задачи 2 (стр.87)

### 1.3 Задача 3 (стр.124) В.10

Условие

Создать объект класса Щенок, используя классы Животное, Собака. Методы: вывести на консоль имя, подать голос, прыгать, бегать, кусать.

Таблица   идентификаторов

|  |  |  |
| --- | --- | --- |
| Наименование | Тип Данных | Назначение |
| Dog puppy=new Dog("Sirko"); | String | Входная |

Блок схема

Задать имя собаки

Обучать собаку к различным командам

Да

Собака обучена

подать голос, прыгать, бегать, кусать

Код программы

public class Blinov4A10\_Puppy {  
 public static void main(String[] args) {  
  
 Dog puppy=new Dog("Sirko");  
 puppy.showName();  
 puppy.makeVoice();  
 puppy.makeBite();  
 puppy.makeJump();  
 puppy.makeRun();  
}  
}  
  
class Animal {  
 private final String name;  
  
 public Animal(String name) {  
 this.name=name;  
 }  
  
 public void showName() {  
 System.*out*.println(name);  
 }  
}  
  
class Dog extends Animal {  
  
 public Dog(String name) {  
 super(name);  
 }  
  
 public void makeJump() {  
 System.*out*.println("I'm jumping");  
 }  
  
 public void makeVoice() {  
 System.*out*.println("Bow-wow !");  
 }  
  
 public void makeRun() {  
 System.*out*.println("Running around...");  
 }  
  
 public void makeBite() {  
 System.*out*.println("Clunk...");  
 }

Выполнение

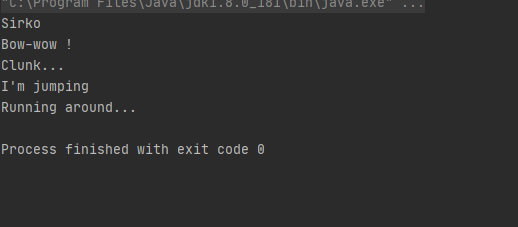


Рисунок 3 – Результат работы задачи 3 (стр.124)

### 1.4 Задача 4 (стр.145) В.10

Условие

Создать класс Mobile с внутренним классом, с помощью объектов которого можно хранить информацию о моделях телефонов и их свойствах.

Таблица   идентификаторов

|  |  |  |
| --- | --- | --- |
| Наименование | Тип Данных | Назначение |
| brand  model | String  String | Входная  Выходная |

Блок схема

Создать метод model

Создать метод brand

Занести данные

данные есть null

Показ данных на консоли

Ошибка введите данные

Код программы

public class Mobile {  
  
 public static <ModelProperties> void main(String[] args) {  
  
 String brand;  
 String model;  
 ModelProperties modelProperties;  
  
 brand:"Samsung Galaxy".toString();  
 System.*out*.println("brand:Samsung Galaxy");  
 model:"A71".toString();  
 System.*out*.println("model A71");  
  
 brand:"Xiaomi".toString();  
 System.*out*.println("brand:Xiaomi");  
  
 model:"model:Redmi note 8 pro".toString();  
  
 System.*out*.println("model Redmi note 8 pro");  
  
 brand:"Iphome".toString();  
  
 model:"X".toString();  
  
 System.*out*.println("brand: Iphone");  
 System.*out*.println("model: X");  
  
  
 }  
  
 private class ModelProperties {  
  
 private String model;  
 private long accumCapacity;  
 private double screenSize;  
  
 public void setModel(String X) {  
 this.model = model;  
  
  
  
 }  
  
 }}

Выполнение

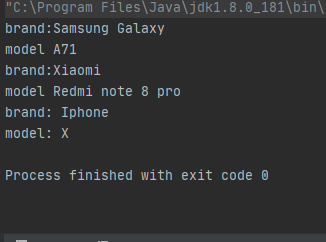


Рисунок 4 – Результат работы задачи 4 (стр.145)

### 1.5 Задача 5 (стр.194) в 10

Условие

В тексте найти и напечатать n символов (и их количество), встречающихся наиболее часто.

Таблица Индетификаторов

|  |  |  |
| --- | --- | --- |
| Наименование | Тип Данных | Назначение |
| k=1  str = sc.nextLine(); | Integer  String | Входная  Выходная |

Блок-Схема

Создать метод Scanner

Примнить строку

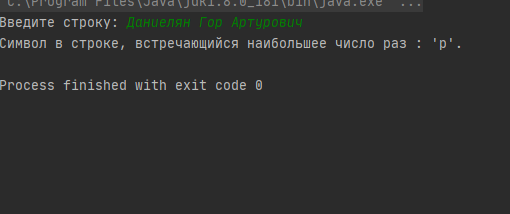
Расчитать буквы

for (int i = 0; i < str.length(); i++) { Character c = str.charAt(i);

Код программы

import java.util.HashMap;  
import java.util.Iterator;  
import java.util.Map;  
import java.util.Map.Entry;  
import java.util.Scanner;  
import java.util.Set;  
// Практика день 5 Даниелян Гор Артурович ИС 18-2К стр 194 вариант 10  
public class Main {  
  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 System.*out*.print("Введите строку: ");  
 String str = sc.nextLine();  
 sc.close();  
  
 Map<Character, Integer> map = new HashMap<Character, Integer>();  
 for (int i = 0; i < str.length(); i++) {  
 Character c = str.charAt(i);  
 int k = 1;  
 if (map.containsKey(c)) {  
 k = map.get(c) + 1;  
 }  
 map.put(c, k);  
 }  
  
 Set<Entry<Character, Integer>> setvalue = map.entrySet();  
 Iterator<Entry<Character, Integer>> i = setvalue.iterator();  
 Entry<Character, Integer> max = null;  
 while (i.hasNext()) {  
 Entry<Character, Integer> me = i.next();  
 if (max == null) {  
 max = me;  
 } else if (me.getValue() > max.getValue()) {  
 max = me;  
 }  
 }  
  
 System.*out*.println("Cимвол в строке, встречающийся наибольшее число раз : \'" + max.getKey() + "\'.");  
 } }

Выполнение



## 2 Вторая неделя : разработка АИС

## Введение

Под автоматизированной информационной системой (АИС) мы понимаем, совокупность программно-аппаратных средств, предназначенных для автоматизации деятельности, связанной с хранением, передачей и обработкой информации.

АИС, основанная на базе данных, служит для сбора, накопления, хранения информации, а также её эффективного использования для различных целей. Информация представляется в виде данных, хранимых в памяти ЭВМ. При проектировании АИС, с одной стороны, решается вопрос о том, какие сведения и для каких целей будут содержаться в системе, с другой – как соответствующие данные будут организованы в памяти ЭВМ, и как они будут обрабатываться при эксплуатации АИС.

АИС может быть определена как комплекс автоматизированных информационных технологий, предназначенных для информационного обслуживания – организованного непрерывного технологического процесса подготовки и выдачи потребителям научной, управленческой и др. информации, используемой для принятия решений, в соответствии с нуждами для поддержания эффективной деятельности.

Основной причиной создания и развития АИС является необходимость ведения учёта информации о состоянии и динамике объекта, новому аэропорту города Калининград которому посвящена система. На основании информационной картины, создаваемой системой, руководители различного звена могут принимать решения об управляющих воздействиях с целью решения текущих проблем.

Целью учебной практики служит создать программный продукт обеспечивающий автоматизацию работы, для удобного и быстрого доступа к информации, характеризующей работу аэропорта города Калининграда.

2. 1 Описание предметной области

Аэропорт – сложная система, основной целью которой является качественное обслуживание клиентов, которое заключается в транспортировке пассажиров и грузов, предоставление услуг, направленных на улучшение удобства перелёта.

Вместе с этим необходимо обеспечить выполнение ряда других задач, таких как выбор, и зарплата персонала, поддержание оборудования в хорошем состоянии, ремонт, обеспечение авиационной безопасности, составление и изменение расписания полётов, продажа и бронирование билетов и др.

Работа современного аэропорта невозможна без автоматизации производственных и бизнес-процессов. Так как аэропорт – большая и сложная система, будем рассматривать упрощённую модель, в которой сущностями будут самолёты, пилоты, пассажиры и багаж, а связями – рейсы и принадлежность багажа.

### 2. 2 Концептуальная модель предметной области

# Концептуальная модель отражает состав и взаимодействие объектов базы данных. Для этой цели разработано несколько систем соглашений о представлении информации, содержащейся в базе данных.

# Одним средством концептуального моделирования данных является универсальный язык моделирования UML. UML предоставляет средства для создания визуальных моделей, которые единообразно понимаются всеми разработчиками, вовлеченными в проект, и являются средством коммуникации в рамках проекта. Диаграмма в UML - это графическое представление набора элементов. Диаграммы рисуют для визуализации системы с разных точек зрения.

Для наглядного отображения оптимально подходит диаграмма вариантов использования. Этот вид диаграмм позволяет создать список операций, которые выполняет система, используется при описании бизнес процессов автоматизируемой предметной области, определении требований к будущей программной системе. Отражает объекты, как системы, так и предметной области и задачи, ими выполняемые.

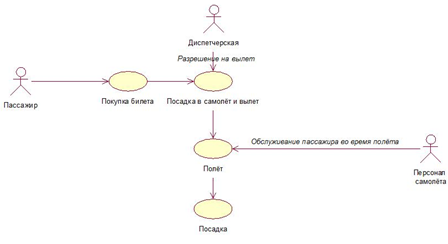


Рисунок 1 – Диаграмма вариантов использования

На рисунке 1 показана диаграмма вариантов использования для моей информационной системы. Она отображает основную концепцию разрабатываемой информационной системы. В данном случае:

Актёр – пассажир – человек который покупал билет самолета и , которому необходим перелёт.

Актёр – диспетчерская – разрешает или запрещает вылет самолёта на намеченный рейс.Это один из главных работников аэропорта потому что он можно сказать регулирует воздушное пространство и ежедневно из его работы зависеть жизнь миллион человека

Актёр – персонал – работники самолёта, обеспечивающие контроль за пассажирами и их обслуживание во время полёта.

Из диаграммы видим, что пассажир приобретает билет и занимает своё место в самолёте, когда все пассажиры на месте, диспетчерская даёт разрешение на вылет. Во время полёта пассажир взаимодействует с персоналом. Следующий и заключительный этап – посадка.

2. 3 Диаграмма классов (ClassDiagram)

Диаграмма классов отражает, в частности, различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывает их внутреннюю структуру и типы отношений.

Диаграмма классов (classdiagram) — диаграмма языка UML, на которой представлена совокупность декларативных или статических элементов модели, таких как классы с атрибутами и операциями, а также связывающие их отношения.

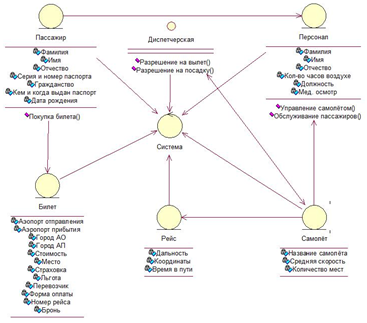


Рисунок 2 – Диаграмма классов

На рисунке 2 представлена диаграмма классов моего проекта.

Система – управляющий класс (controlclass), контролирующий работу всей системы и хранение данных. Диспетчерская – является интерфейсом (interfaceklass) и осуществляет 2 операции. Остальные объекты, представленные на диаграмме, являются классами-сущностями (entityclass): пассажир, персонал, самолёт, рейс и билет. Для каждого из них определены атрибуты, а также операции. На диаграмме видно, что все классы взаимодействуют с системой, а также некоторые друг с другом.

Диаграмма прецедентов — диаграмма, отражающая отношения между [актёрами](http://ru.wikipedia.org/wiki/%D0%90%D0%BA%D1%82%D1%91%D1%80_(UML)) (человеком или системой) и [прецедентами](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D1%86%D0%B5%D0%B4%D0%B5%D0%BD%D1%82_(UML)) (последовательностью действий) и являющаяся составной частью модели прецедентов.

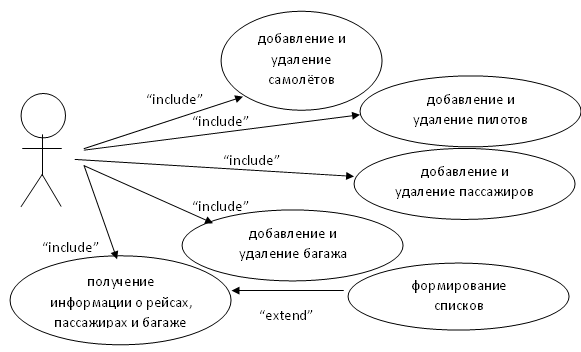


Рисунок 3 – Диаграмма прецедентов

На рисунке 3 описывается функциональное значение системы, то есть что система будет выполнять в процессе своего функционирования.

Пользователь имеет доступ к таким возможностям программы как:

- добавление и удаление самолётов;

- добавление и удаление пилотов;

- добавление и удаление пассажиров;

- добавление и удаление багажа;

- получение информации о рейсах, пассажирах и багаже.

Получение информации о рейсах, пассажирах и багаже может выполняться с помощью формирования списков.

2.4 Построение ER-диаграммы

Рассмотрим предметную область «АИС учет работы редакции аэропорта». Проанализировав ее, выделим часть, которую будем автоматизировать. Список сущностей, которые я выделил из предметной области, будет выглядеть так: самолёты, пилоты, пассажиры и багаж.

Связи между объектами можно сформировать в список следующего вида: рейс (связь: самолёт – пилот – пассажир ), принадлежит (связь: пассажир – багаж ).

Связи между сущностями отображены на ER-диаграмме предметной области "АИС учет работы редакции аэропорта "

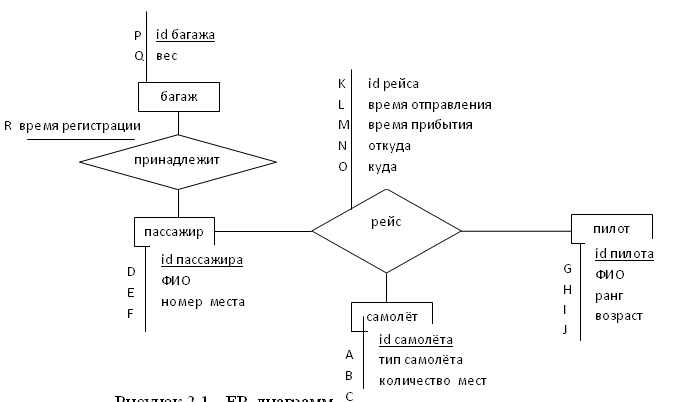


Рисунок 4 – ER диаграмма аэропорта

### 2.5 Диаграмма ER- экземпляров

Для понимания логики работы предметной области, а также установления степени связности и класса принадлежности таблиц разработана диаграмма ER-экземпляров (рис.5)

пассажир 1

багаж 1

пассажир 2

багаж 2

пассажир 3

Рисунок 5 – Диаграмма ER- экземпляров

Построение диаграммы последовательностей

Основными элементами диаграммы последовательности являются обозначения объектов (прямоугольники с названиями объектов), вертикальные "линии жизни" ("Пассажир", "Сотрудник аэропорта", "Инспектор по досмотру", "Транспортировка багажа"), отображающие течение времени, прямоугольники, отражающие деятельность объекта или исполнение им определенной функции (прямоугольники на пунктирной "линии жизни"), и стрелки, показывающие обмен сигналами или сообщениями между объектами.

Входом в диаграмму последовательностей служит выбор и покупка билета, далее Пассажир, взаимодействуя с системой и Сотрудником аэропорта для дальнейшего прохождения регистрации на рейс. Далее Сотрудник проверяет билет и если есть багаж, то отправляет его на погрузку. После данной процедуры идет выбор сиденья на воздушном судне и выдача посадочного билета.

Потом наступает очередь инспектора, который проверяет билеты и проводит осмотр пассажиров. После удачного прохода всех проверок, пассажир идет на посадку в самолет, как это показано на рисунке 6.

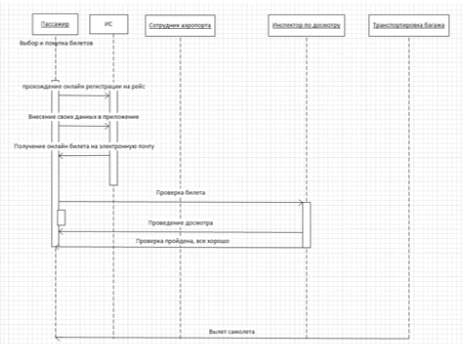


Рисунок 6 – Диаграмма последовательностей для online регистрации пассажира и проведения осмотра перед полетом

Как мы можем увидеть из диаграмм, действие сотрудников аэропорта в регистрации пассажиров не требуется и, соответственно, экономятся средства и время.

Также давайте распишем спецификацию для каждого отдельного варианта использования и сделаем диаграмму последовательностей (рис.7), (рис.8).

Вход в систему.

Наименование: Вход в систему.

Actor, взаимодействующие с прецедентом: Клиент/Пассажир.

Описание: Вход в систему служит для того, чтобы клиент мог в своем личном кабинете осуществлять регистрацию на предстоящий рейс по прибытии в аэропорт.

Приоритетный маршрут: Пользователь не может быть зарегистрирован на рейс без ввода данных билета и данных самого пассажира.



Рисунок 7 – Диаграмма последовательностей для прецедента "Вход в систему"

Регистрация.

Наименование: Регистрация.

Actor, взаимодействующие с перцедентом: Клиент/Пассажир.

Описание: Регистрация на рейс проходит в процессе внесения данных о пассажире

Предположение: Регистрация обязательна для дальнейшего прохождения контроля перед полетом.

Постусловия: После успешной регистрации, пассажир проходит на досмотр, а позже на посадку в самолет.

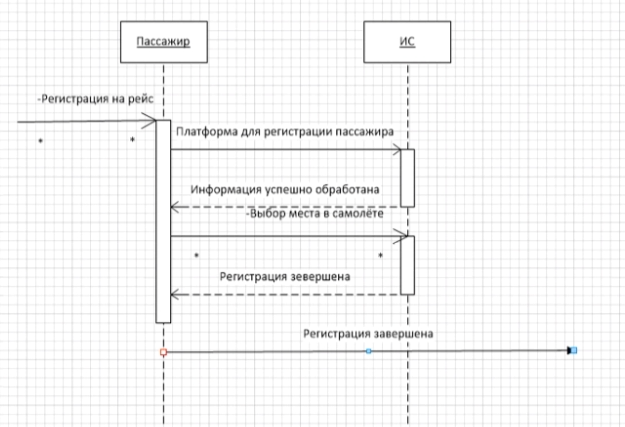


Рисунок 8 – Диаграмма последовательностей для прецедента "Регистрация

Построение диаграммы состояний

Каждая диаграмма состояний в UML описывает все возможные состояния одного экземпляра определенного класса и возможные последовательности его переходов из одного состояния в другое, то есть моделирует все изменения состояний объекта как его реакцию на внешние воздействия.

Диаграммы состояний чаще всего используются для описания поведения отдельных объектов, но также могут быть применены для спецификации функциональности других компонентов моделей, таких как варианты использования, актеры, подсистемы, операции и методы. Так была разработана диаграмма состояний (рис.9).

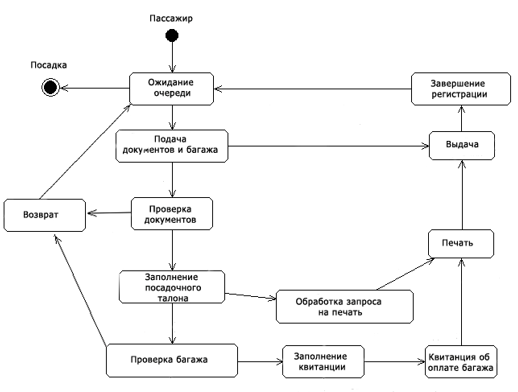


Рисунок 9 – Диаграмма состояний Пассажира

2.6 Построение диаграммы деятельности

Каждая диаграмма деятельности акцентирует внимание на последовательности выполнения определенных действий, которые в совокупности приводят к получению желаемого результата. Диаграмма деятельности – UML диаграмма, на которой показаны действия, состояния которых описано на диаграмме состояний. Так была построена диаграмма деятельности (рис.10).

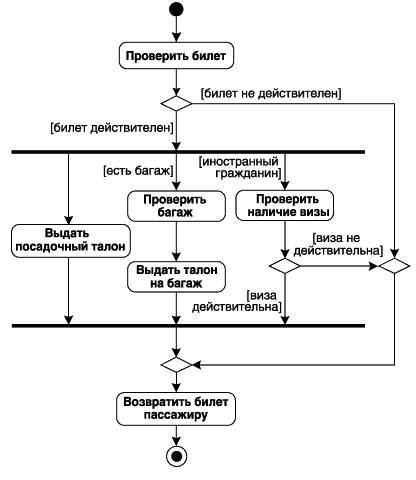


Рисунок 10 – Диаграмма регистрации пассажиров в аэропорту

Первоначально выполняется деятельность по проверке билета. В случае если билет не действителен, он возвращается пассажиру, при этом никаких дополнительных действий не выполняется.

Если же билет действителен, то пассажиру выдается посадочный талон. В дополнение к этому проверяется гражданство и наличие багажа у пассажира. Если есть багаж, то его проверка может быть выполнена параллельно, по результатам которой пассажиру выдается талон на багаж.

Если пассажир является иностранным гражданином, то дополнительно проверяется наличие у него визы. Если виза действительна, то проверка завершается успешно, и пассажир с возвращенным ему билетом может проследовать на посадку.

Если же виза окажется не действительной, то для этого пассажира посадка на данный рейс оказывается невозможной. В этом случае ему не выдается посадочный талон и талон на багаж, в случае его наличия, поскольку происходит прекращение всех выполняемых сотрудниками аэропорта действий. Была создана БД Аirport Sql в нем 12 основных таблицы

* AirCompany
* AirPort
* Buyer
* City
* Country
* Events
* Flight
* Passengers
* Payment
* Plane
* Tickets

Вот данные из них

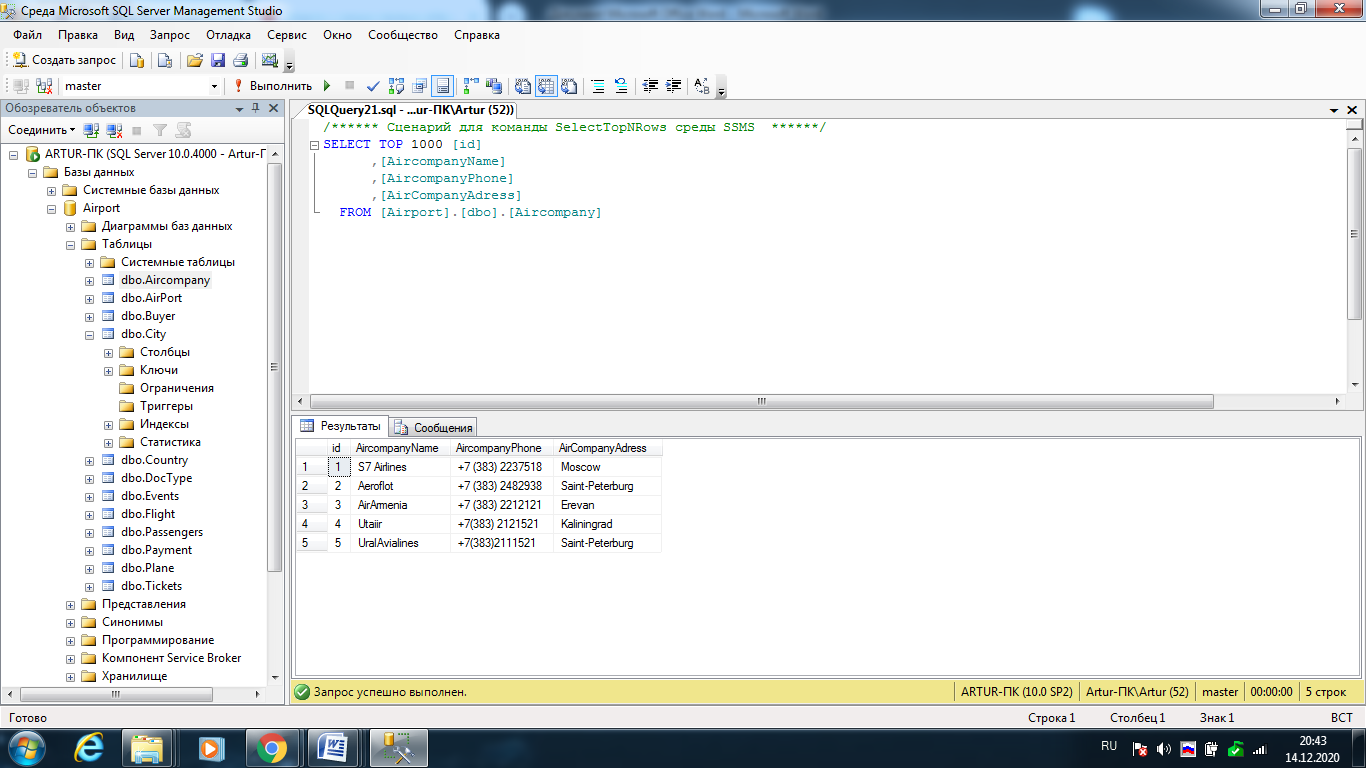


Рисунок 11 – Таблица AirCompany

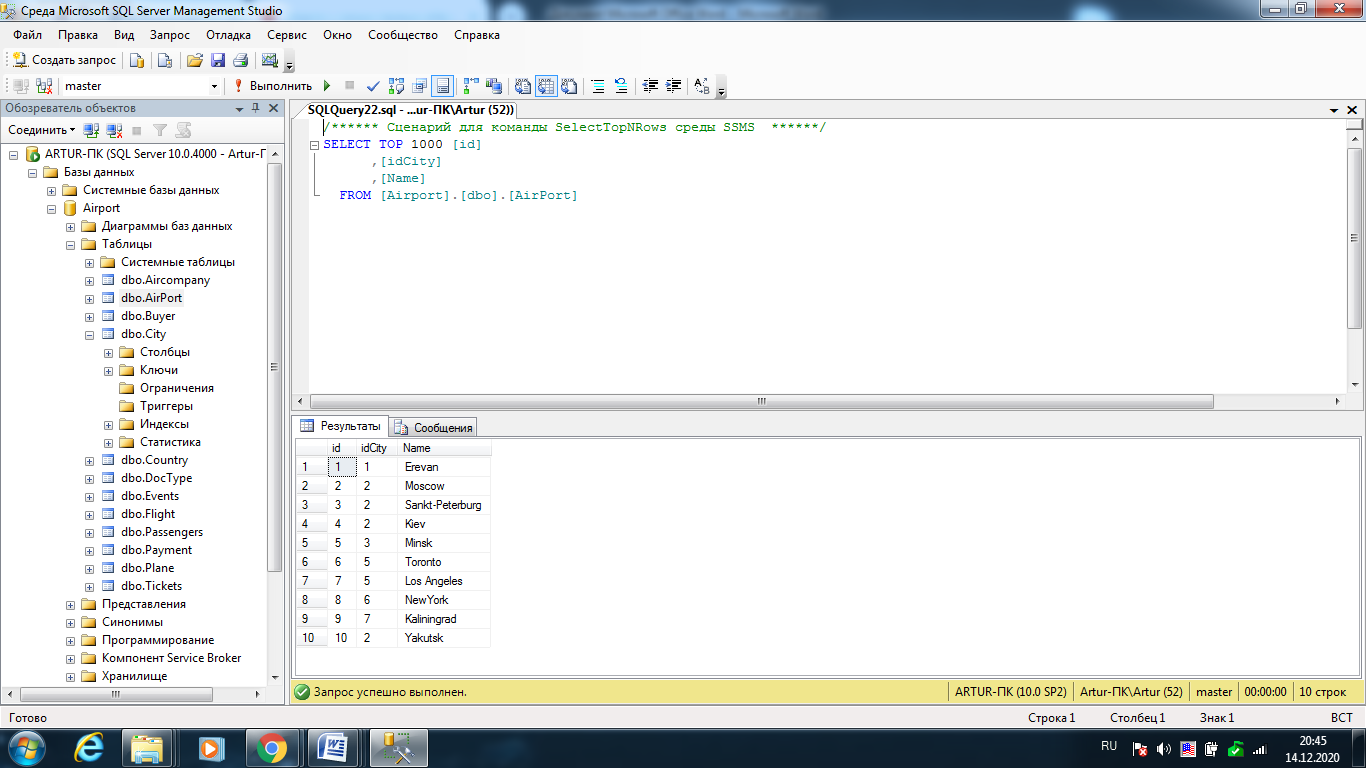


Рисунок 12 – Таблица AirPort

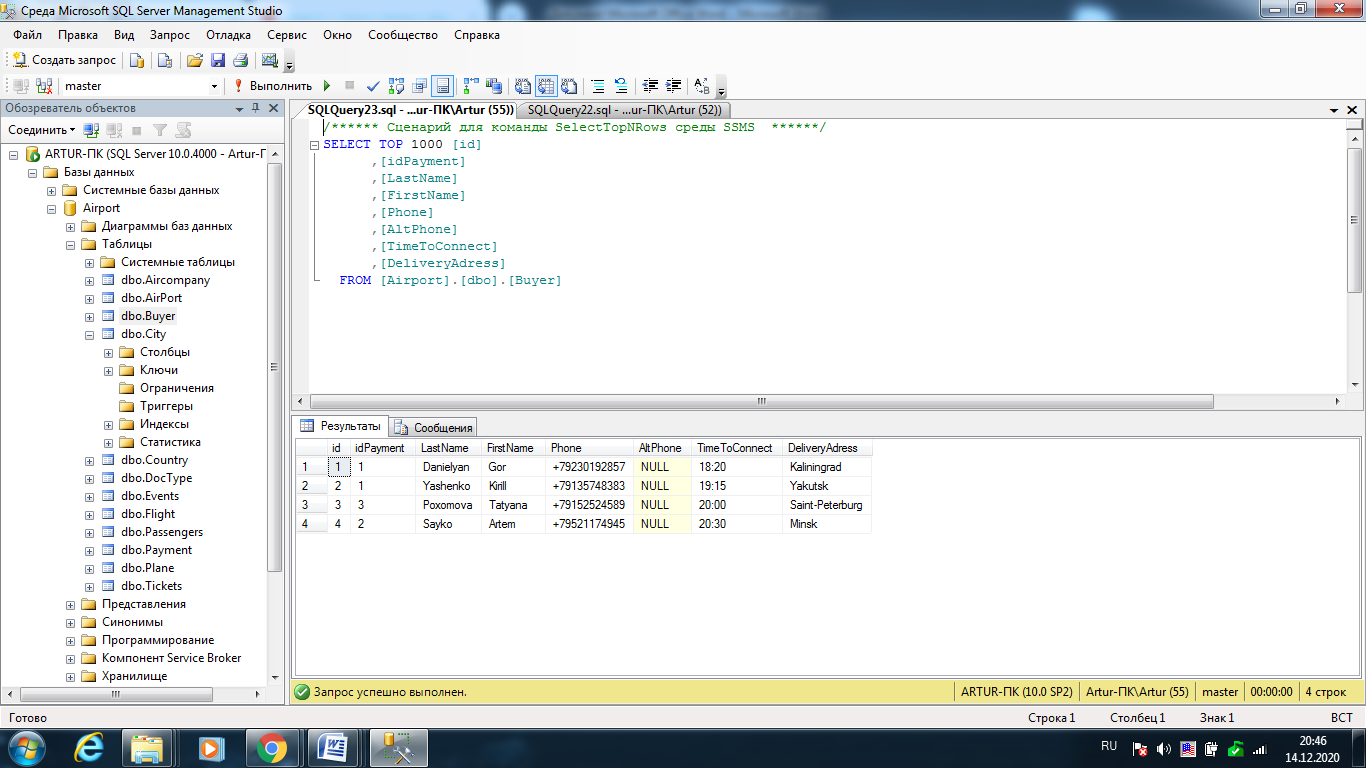


Рисунок 13 – Таблица Buyer

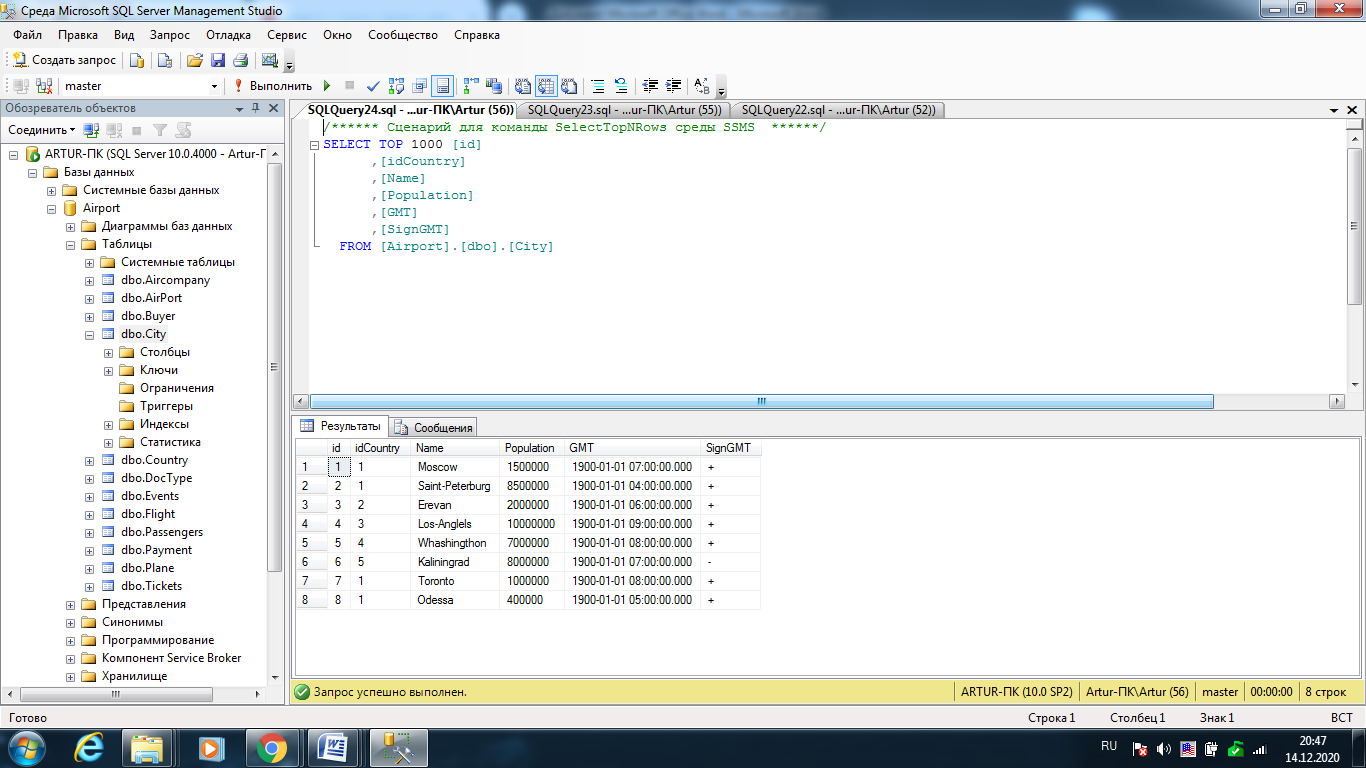


Рисунок 14 – Таблица City

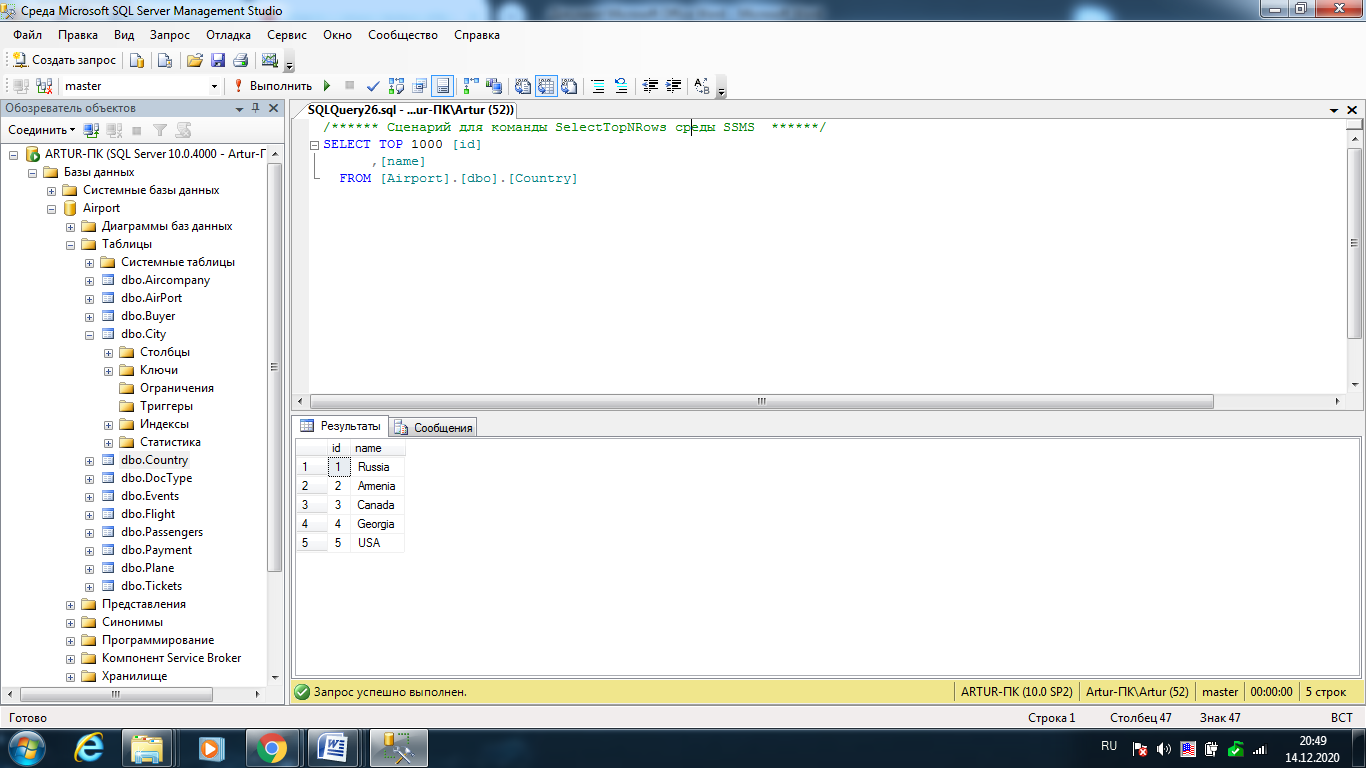


Рисунок 14 – Таблица Country

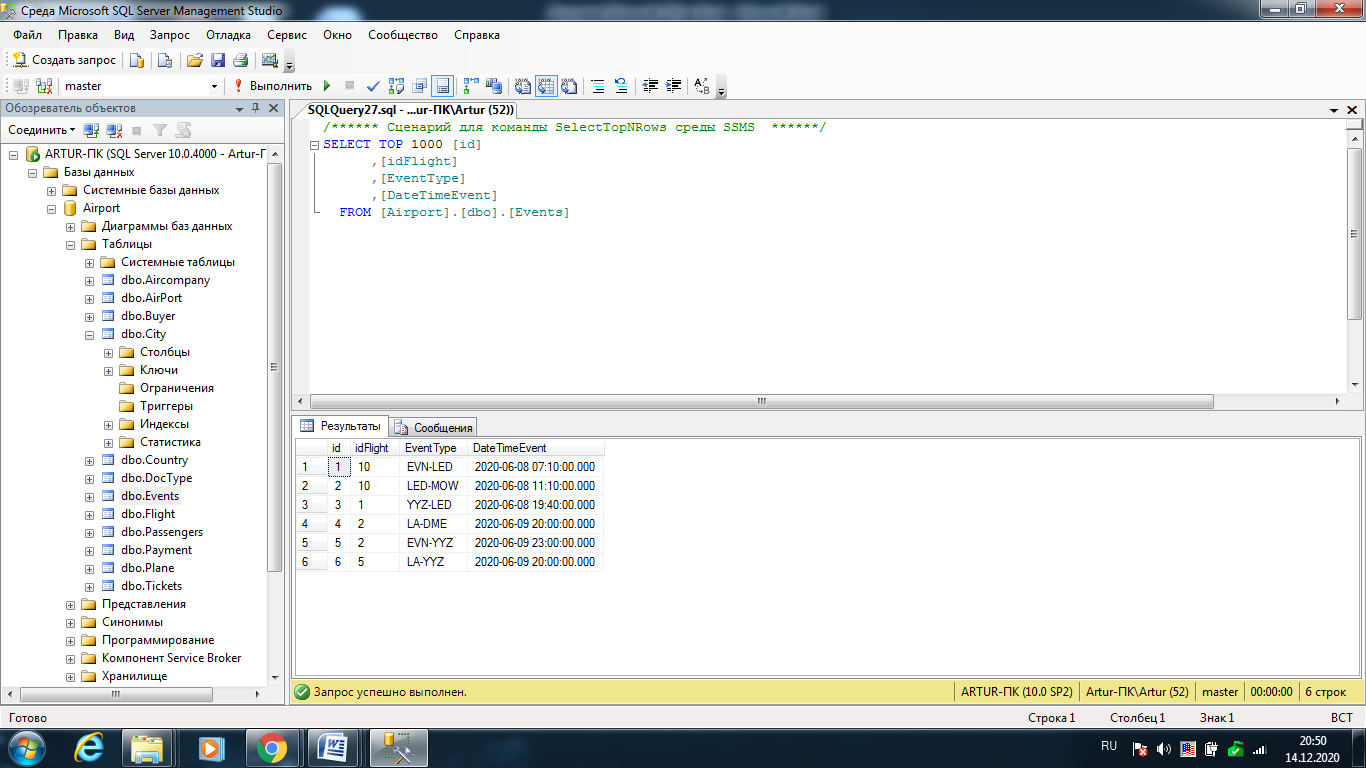


Рисунок 15 – Таблица Events

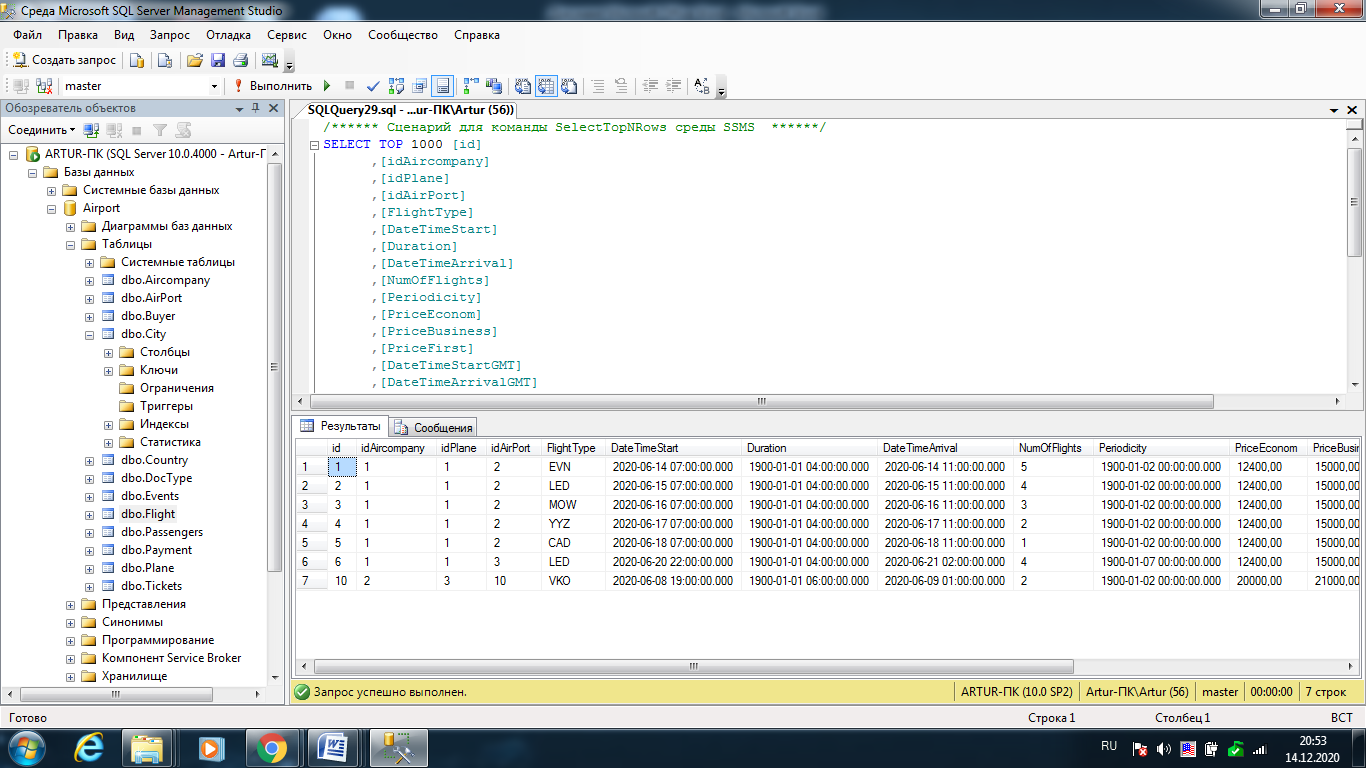


Рисунок 16 – Таблица Fights

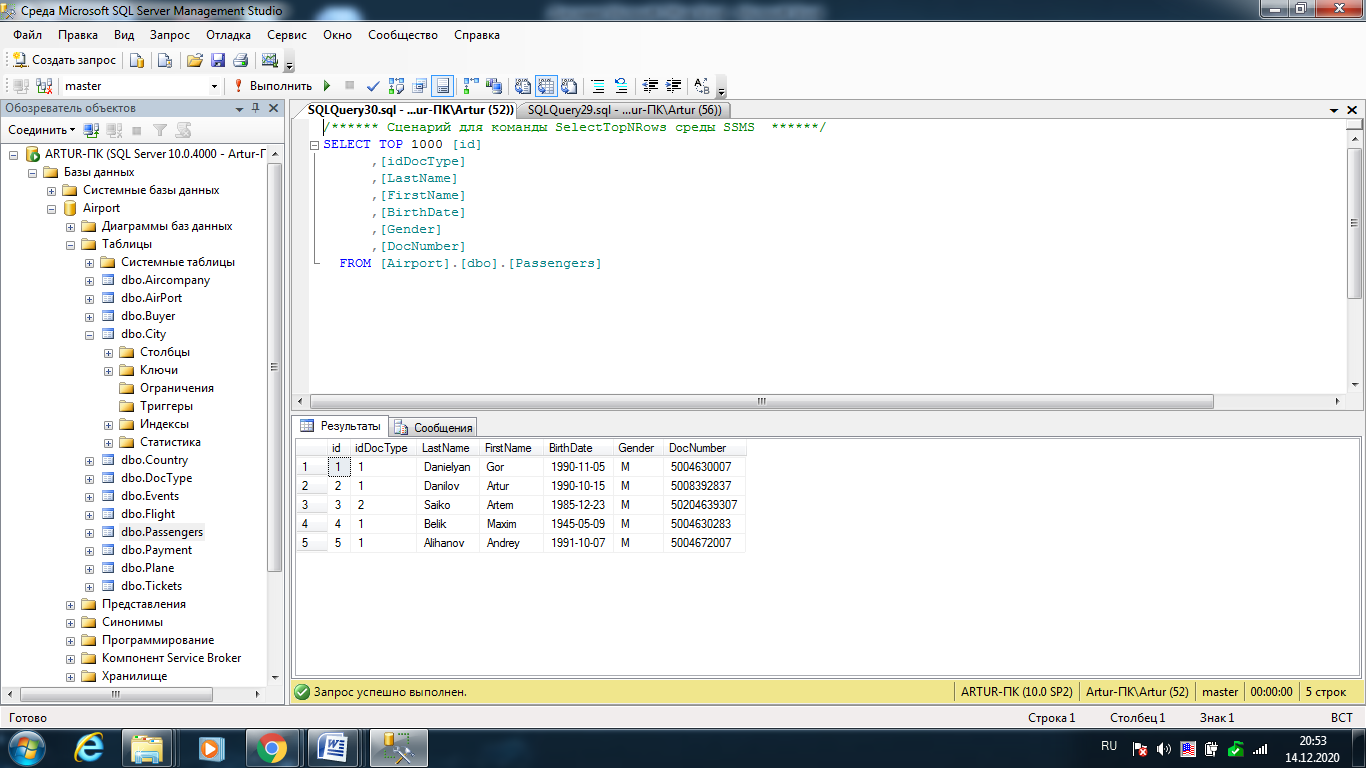


Рисунок 17 – Таблица Passengers

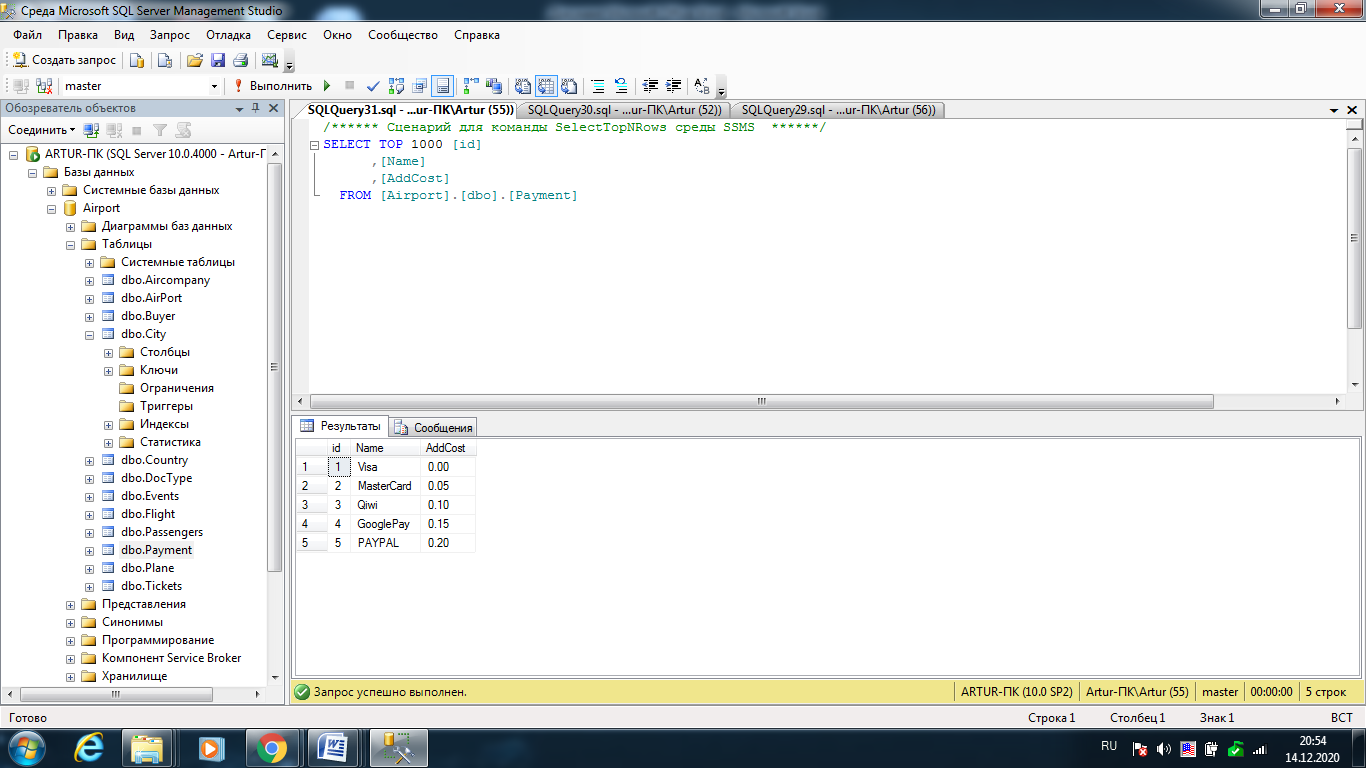


Рисунок 18 – Таблица Payment

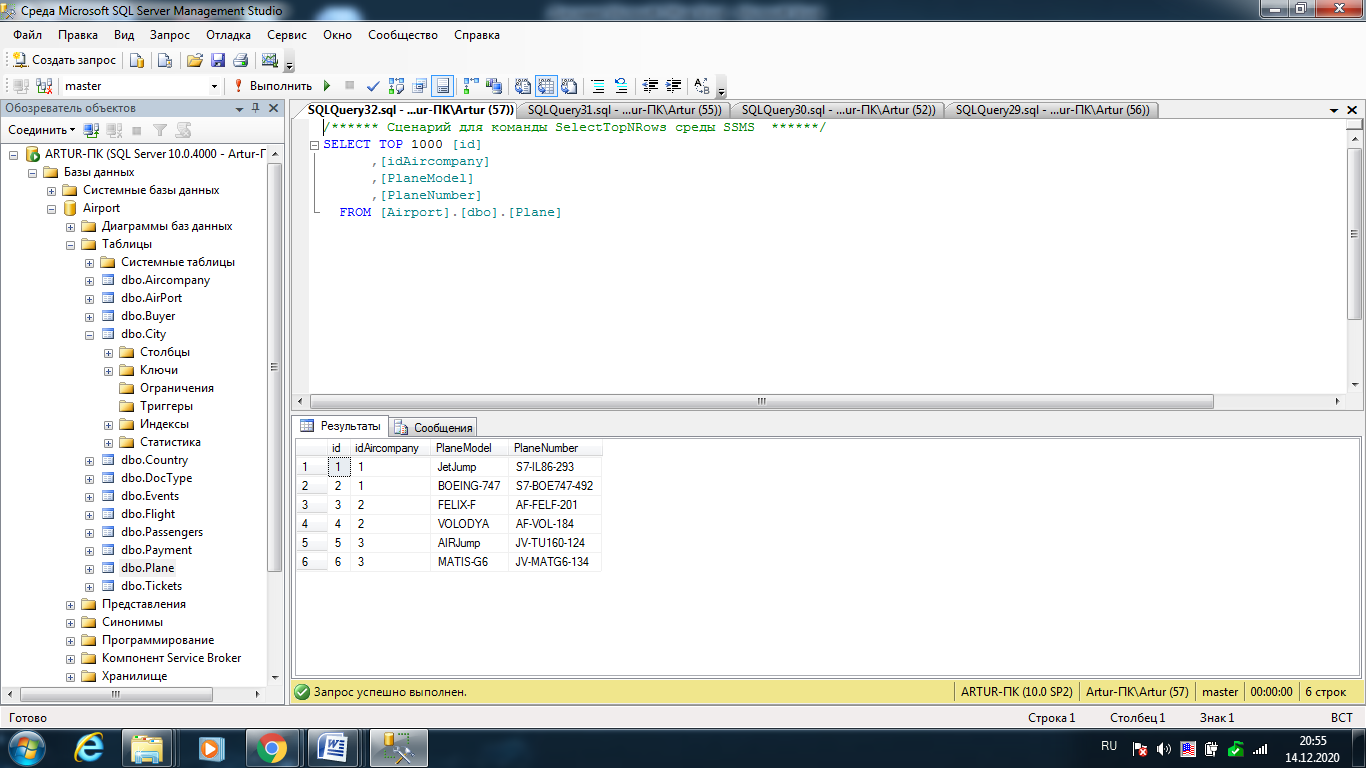


Рисунок 19 – Таблица Plane

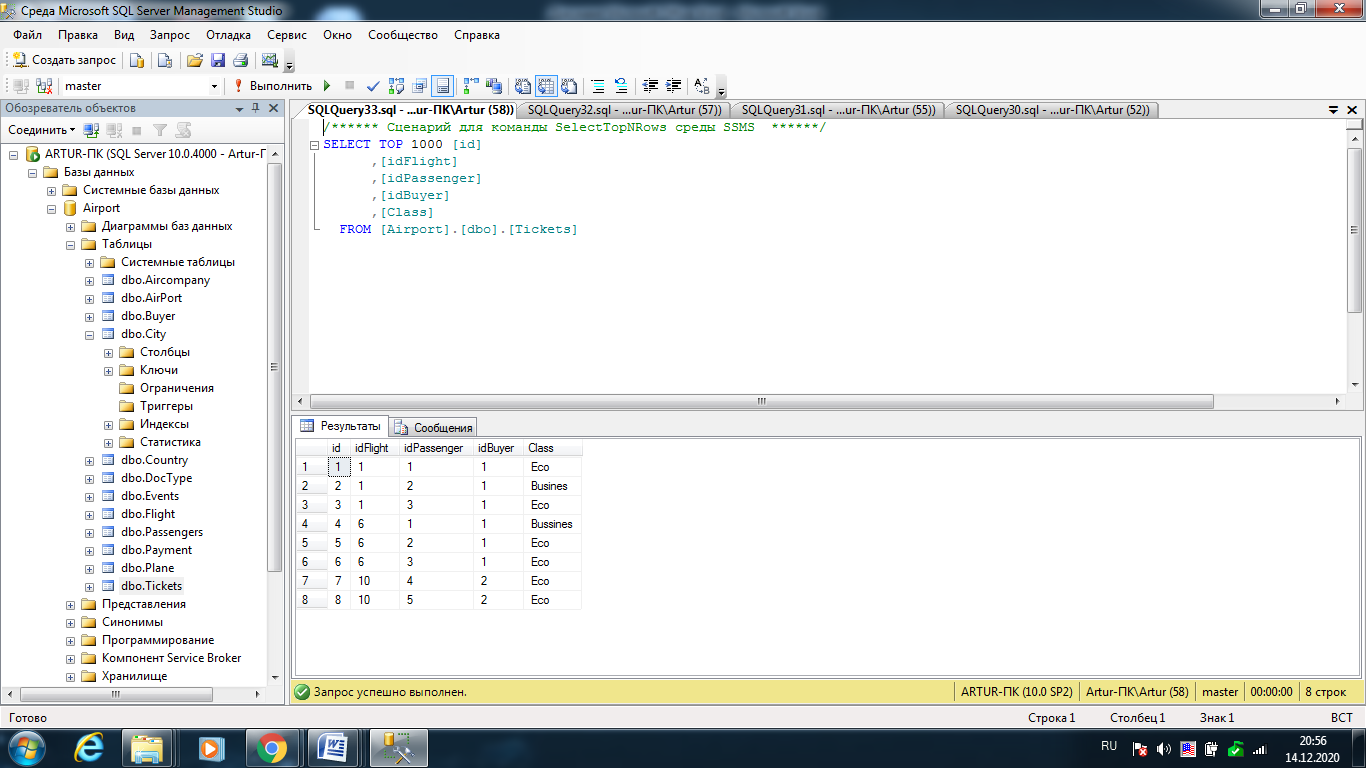


Рисунок 19 – Таблица Tickets

# Заключение

Была сформулирована и создана АИС Аэропорта на основе БД sql АИС, основанная на базе данных, служит для сбора, накопления, хранения информации. АИС может быть определена как комплекс автоматизированных информационных технологий, предназначенных для информационного обслуживания – пользователя.

## 3 Третья неделя: Разработка мобильного приложения

3.1 Подробное описание

Среда разработки программы, в котором будет разработыватся приложение называется Android Studio. код программы будет на языке Java.

Приложение будет называтся Airoport С самого начала, в нижнем углу справа будет изображение белого плюсика в синнем фоне. При нажатии на плюсик у нас открывается Интерфейс разработчика. Это нужно для того, чтобы добавлять расписание рейсов.

В интерфейсе будет представлено следующее:

- Заголовок рейса, где разработчик будет писать место отправки и прибытия.

- Описание рейса, где разработчик пишет всю необходимую информацию о рейсе.

- День недели рейса, где разработчик указывает день отправки рейса.

- Таблица из номеров, где цифра 1– Отмена рейса, цифра 2– Вылет рейса, цифра 3 – Прилет Рейса.

- Добавить рейс, при нажатии на который разработчик добавляет рейс в пользовательский интерфейс.

При открытии приложения, пользователь видит в поле пользовательского интерфейса расписание рейсов и всю необходимую информацию о них (вылетов, прилетов и отмененных рейсов). Плюсик, который находится в нижнем правом углу. Это приложение будет создано для того, чтобы пользователю не тратить время и не искать сайты с расписанием.

### 3.2 Ход выполнения задания

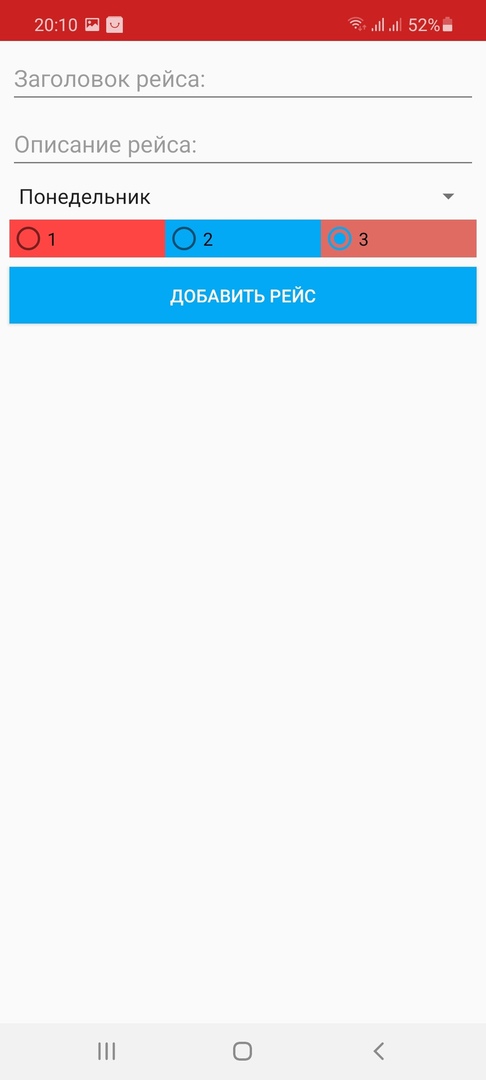


Рисунок 1 – Меню разаботчика

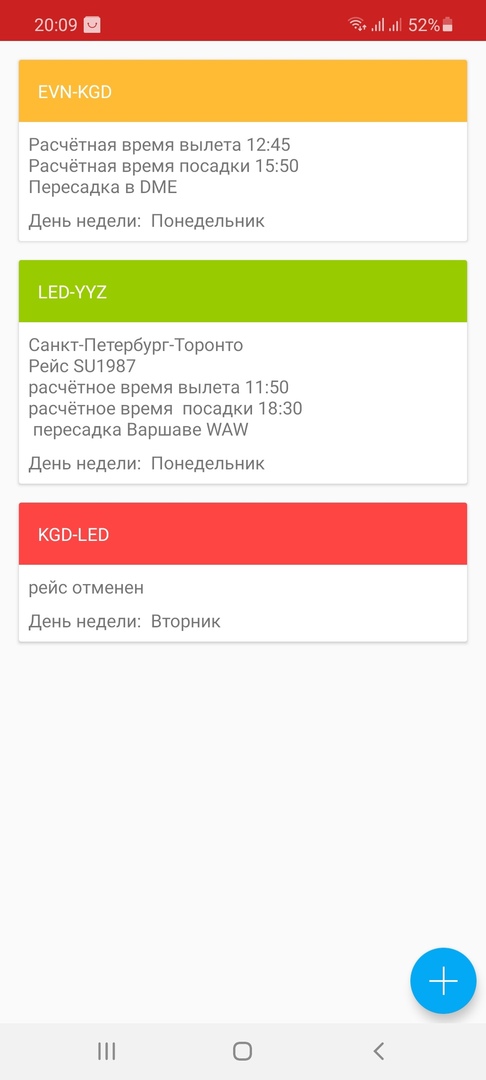


Рисунок 2 – Интерфейс пользователя

## 4 Сайт и оценка эффективности

4.1 Ход выполнения задания

Сам сайт создан на конструкторе   сайтов wix - это простой и эффективный способ создания работающего сайта. Сам сайт содержит готовые шаблоны для разработки собственного своего будущего сайта, что помогает программистам осваивать структуру заполнения и реализации своего будущего сайта.

В ходе выполнения работы, брался за основу основной проект Аэропорта города Ереван. в сайте есть несколько разделов «Расписание рейсов и информация о самом Аэропорте (рис.2). Также расположены контактные данные. где пользователи в случаи каких-нибудь вопросов, могут обратиться к нам.

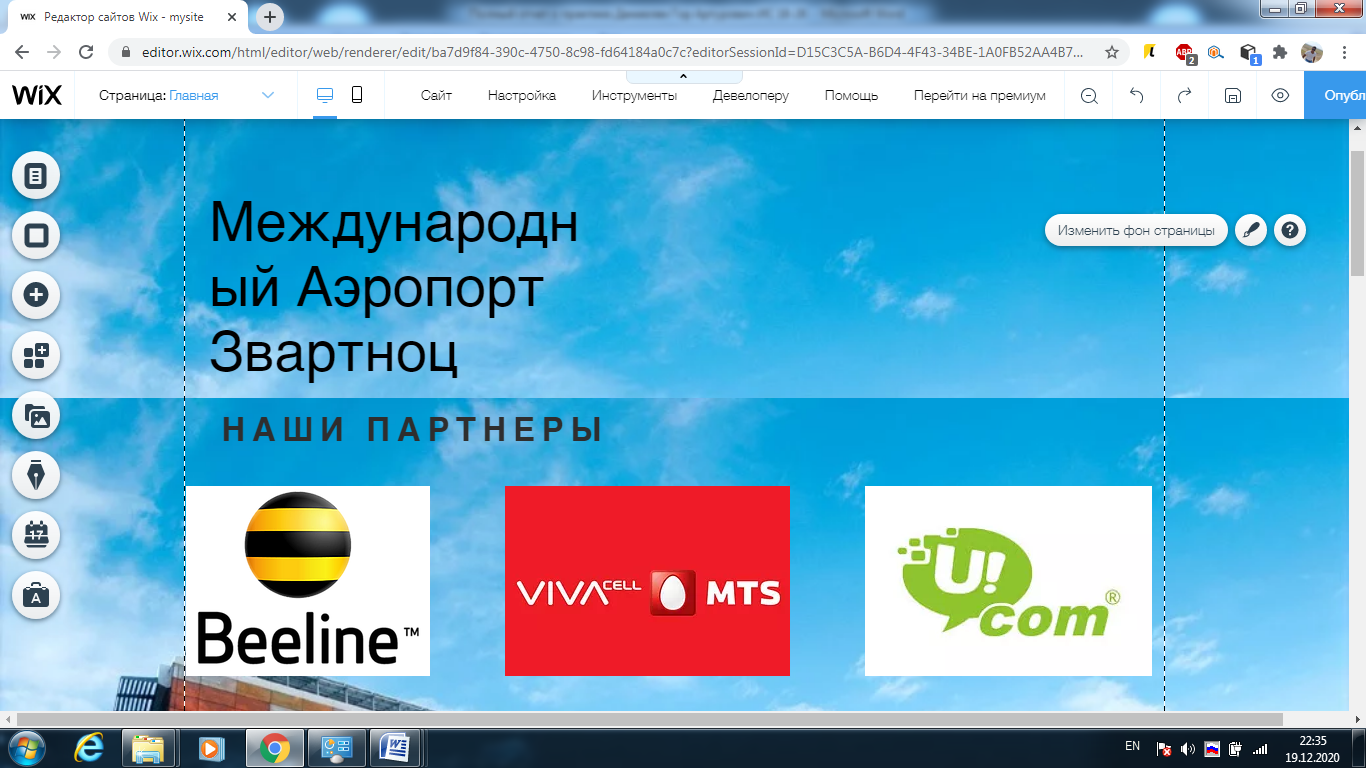


Рисунок 1 – Главная страница

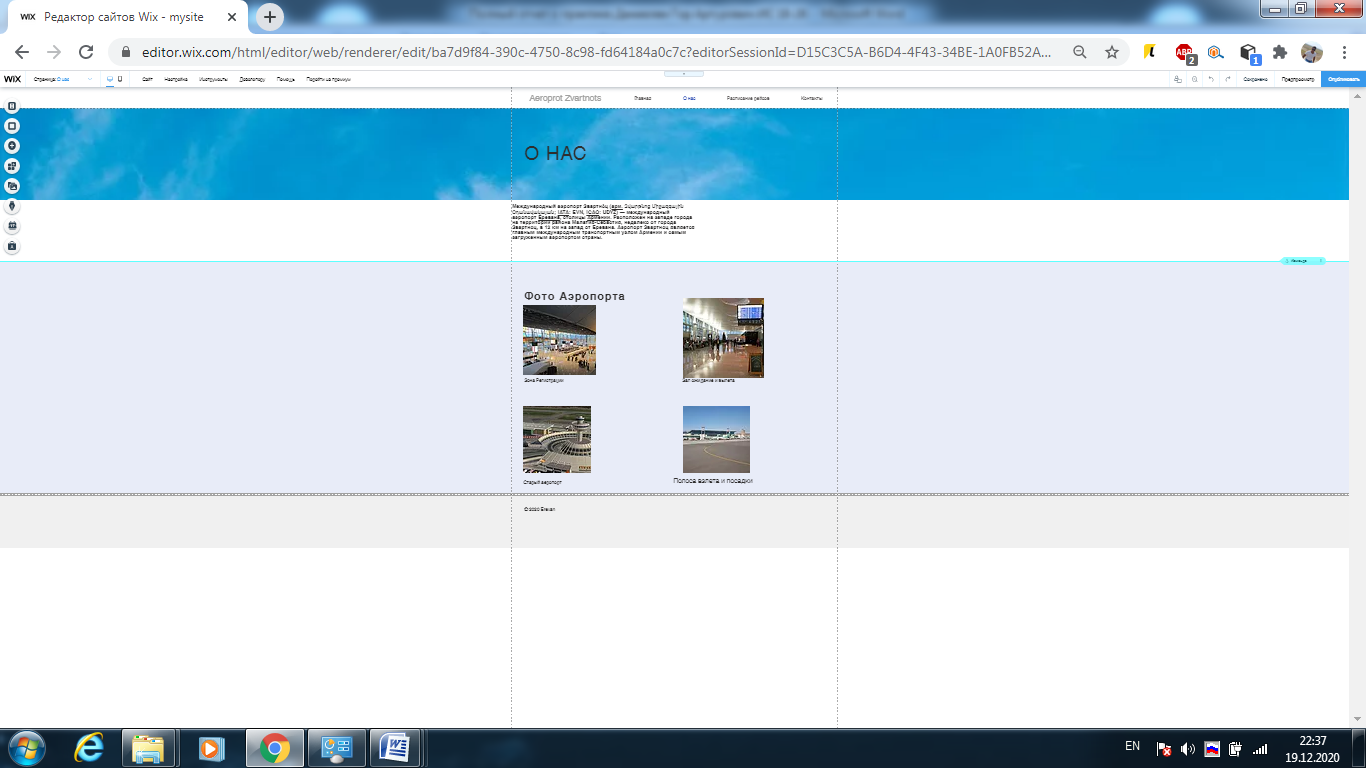


Рисунок 2 – Об аэропорте

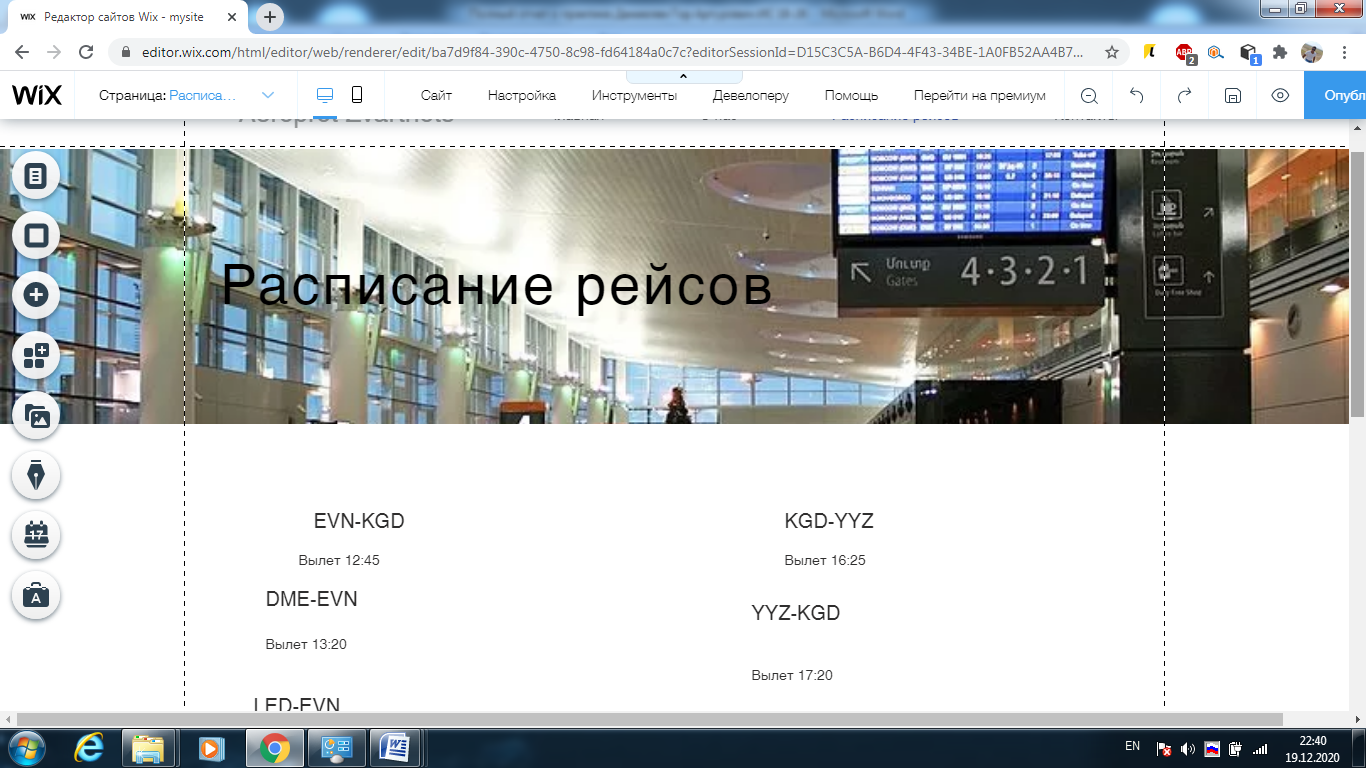


Рисунок 3 – Расписание рейсов

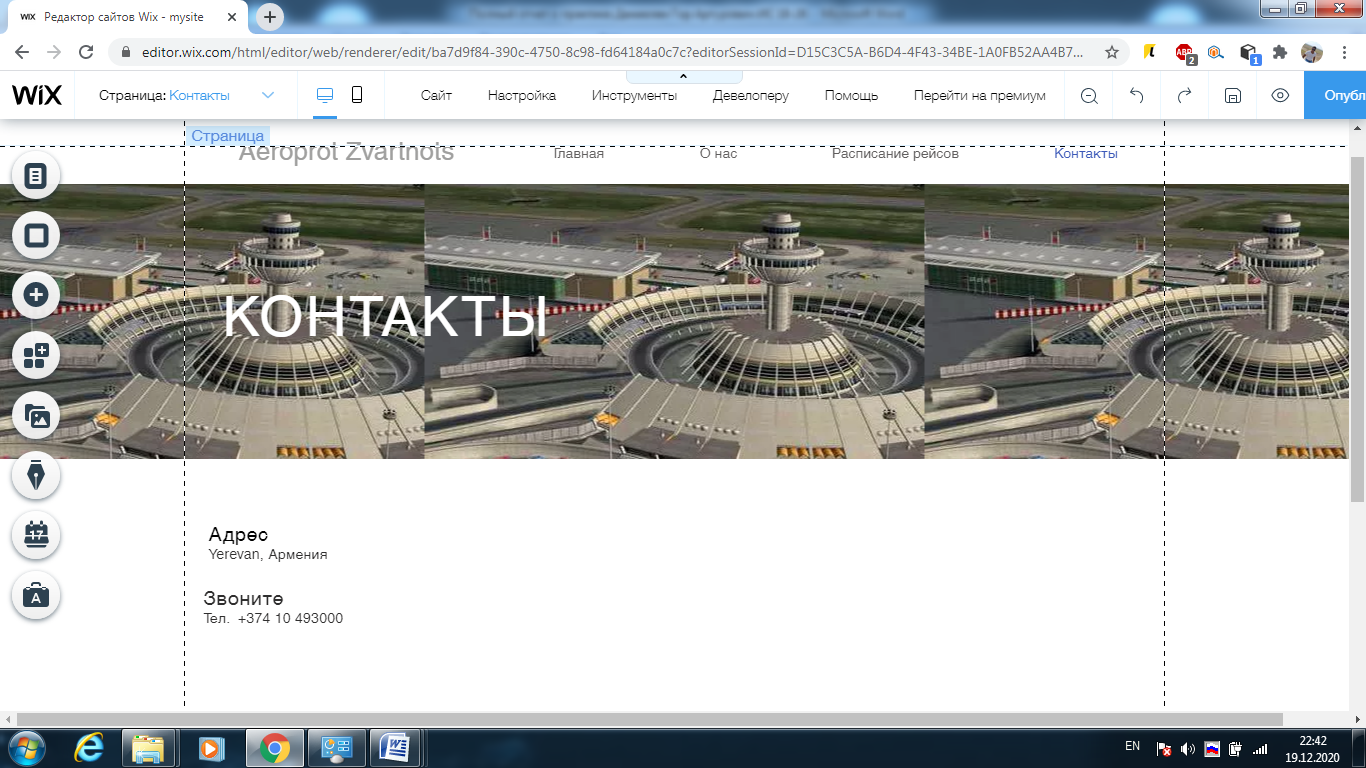


Рисунок 4 – Контакты

### 4.2 Результаты работы

<https://danielyangor2002.wixsite.com/zvartnots> -Сайт

<https://drive.google.com/file/d/1FUL2kXngmZSKNAdXJw20fDGklsKEHxsK/view?usp=sharing> – Приложения

# Заключение

В ходе учебной практики осуществилось ознакомление и приобретение основных навыков практической работы по специальности – "Информационные системы и программирования"

В ходе учебной практики были закреплены и проверены знания по следующим вопросам:

- Решение задач и работа в консольном режиме Java

- Разработка АИС (анализ предметной области, проектирование и выполнение компьютерной реализации автоматизированной информационной системы для заданной предметной области).

- В ходе анализа предметной области, разработка и реализация технического задания

- На основе разработки АИС, разработать свое мобильное приложение по варианту из списка.

- Создать сайт разработанного приложения.

Затруднения:

- создание сайта и наполнение сайта контентом о разработанных приложениях

# Список использованных источников

1. А.М. Вендров. CASE-технологии. Современные методы и средства проектирования информационных систем. М.: Финансы и статистика, 1998, -176с.
2. Грейди Буч, Джеймс Рамбо, Айвар Джекобсон. UML. Руководство пользователя. М. ДМК 2000. 432 с.
3. Джеймс Рамбо, Айвар Якобсон, Гради Буч «UML Специальный справочник». - СПб.: «Питер», 2002.
4. Калянов Г.Н. CASE структурный системный анализ (автоматизация и применение). - М.: ЛОРИ, 1996.
5. Кватрани Т. Rational Rose 2000 и UML. Визуальное моделирование: Пер. с англ. - М.: ДМК Пресс, 2001 - 176с.: ил.
6. Кознов Д.В «Языки визуального моделирования: проектирование и визуализация программного обеспечения.
7. Кознов Д.В Программная инженерия. Часть I. Методическое пособие СПб.: Изд-во СПбГУ, 2005, 40 с.
8. Леоненков А.В «Объектно-ориентированный анализ и проектирование с использованием UML и IBM Rational Rose»