**АННОТАЦИЯ**

Данный эксплуатационный документ состоит из трех разделов и содержит необходимые сведения о программном продукте, представленном в курсовом проекте. В документе отражены назначения и область применения автоматизированной информационной системы «Аэропорт», технические характеристики, экономические затраты, проектирование программного продукта.

**ABSTRACT**

This operational document consists of three sections and contains necessary data on the software product presented in an academic year project. In the document appointments and a scope of the automated information "Airport" system, technical characteristics, economic expenses, design of the software product are reflected.

**СОДЕРЖАНИЕ**

|  |  |
| --- | --- |
| Введение.................................................................................................................................................... | 17 |
| 1. Среда программирования C# ............................................................................................................. | 18 |
| 1.1. История развития языка C# и его особенности.............................................................................. | 18 |
| 1.2. Основные принципы ООП............................................................................................................... | 19 |
| 1.3. Визуальные компоненты С#............................................................................................................. | 21 |
| 2. Назначение программного изделия.................................................................................................... | 25 |
| 2.1. Описание предметной области и назначения программы............................................................. | 25 |
| 2.2. Описание задач программного продукта для разных групп пользователей............................... | 34 |
| 2.3. Описание характеристик проекта ................................................................................................... | 34 |
| 3. Требования к аппаратному и программному обеспечению............................................................. | 36 |
| 3.1. Требования к составу и параметрам технических средств........................................................... | 36 |
| 3.2. Требования к программному обеспечению.................................................................................... | 36 |
| 3.3. Требования к документации ............................................................................................................ | 36 |
| 3.4. Описание структуры и схемы работы программы ........................................................................ | 37 |
| Заключение............................................................................................................................................... | 43 |
| Список терминов и условных сокращений............................................................................................ | 44 |
| Список использованной литературы...................................................................................................... | 45 |
| Приложение 1. Руководство пользователя............................................................................................ | 46 |

**ВВЕДЕНИЕ**

Аэропорт - комплекс сооружений, предназначенный для приёма, отправки, базирования воздушных судов и обслуживания [воздушных перевозок](http://ru.wikipedia.org/wiki/%D0%92%D0%BE%D0%B7%D0%B4%D1%83%D1%88%D0%BD%D1%8B%D0%B9_%D1%82%D1%80%D0%B0%D0%BD%D1%81%D0%BF%D0%BE%D1%80%D1%82), имеющий для этих целей [аэродром](http://ru.wikipedia.org/wiki/%D0%90%D1%8D%D1%80%D0%BE%D0%B4%D1%80%D0%BE%D0%BC), [аэровокзал](http://ru.wikipedia.org/wiki/%D0%90%D1%8D%D1%80%D0%BE%D0%B2%D0%BE%D0%BA%D0%B7%D0%B0%D0%BB) (в крупных аэропортах нередко несколько аэровокзалов), один или несколько грузовых терминалов и другие наземные сооружения и необходимое оборудование.

Одним из самых первых аэропортов мира стал [Кёнигсбергский](http://ru.wikipedia.org/wiki/%D0%9A%D1%91%D0%BD%D0%B8%D0%B3%D1%81%D0%B1%D0%B5%D1%80%D0%B3) аэропорт [Девау](http://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%B2%D0%B0%D1%83_(%D0%B0%D1%8D%D1%80%D0%BE%D0%BF%D0%BE%D1%80%D1%82)), открывшийся в [1919 году](http://ru.wikipedia.org/wiki/1919_%D0%B3%D0%BE%D0%B4).

Существуют [гидроаэропорты](http://ru.wikipedia.org/wiki/%D0%93%D0%B8%D0%B4%D1%80%D0%BE%D0%B0%D1%8D%D1%80%D0%BE%D0%BF%D0%BE%D1%80%D1%82) для обеспечения авиаперевозок на [гидросамолётах](http://ru.wikipedia.org/wiki/%D0%93%D0%B8%D0%B4%D1%80%D0%BE%D1%81%D0%B0%D0%BC%D0%BE%D0%BB%D1%91%D1%82). Такие аэропорты не имеют [взлётно-посадочную полосу](http://ru.wikipedia.org/wiki/%D0%92%D0%B7%D0%BB%D1%91%D1%82%D0%BD%D0%BE-%D0%BF%D0%BE%D1%81%D0%B0%D0%B4%D0%BE%D1%87%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BE%D0%BB%D0%BE%D1%81%D0%B0), — её роль выполняет водная поверхность [водоема](http://ru.wikipedia.org/wiki/%D0%92%D0%BE%D0%B4%D0%BE%D0%B5%D0%BC) — речная, озерная или морская акватория.

Международный аэропорт — аэропорт, который открыт для приёма и отправки [воздушных судов](http://ru.wikipedia.org/wiki/%D0%92%D0%BE%D0%B7%D0%B4%D1%83%D1%88%D0%BD%D0%BE%D0%B5_%D1%81%D1%83%D0%B4%D0%BD%D0%BE), выполняющих международные воздушные перевозки, и в котором осуществляется пограничный и таможенный контроль.

Цель курсового проектирования - создание автоматизированной информационной системы, способной обслуживать работу аэропорта.

Актуальность данного программного продукта обусловлена тем, что воздушный транспорт в настоящее время пользуется большим спросом, следовательно в аэропортах большое множество клиентов. В настоящее время автоматизация рабочих мест в аэропорте просто необходима.

Данный программный комплекс может быть использован в любом аэропорте мира.

**1. СРЕДА ПРОГРАММИРОВАНИЯ C#**

**1.1. История развития языка C# и его особенности**

Для разработки приложения используется интегрированная среда разработки VisualStudio 2005, язык C#.

C# – это новый язык программирования от компании Microsoft. Он входит в новую версию VisualStudio-VisualStudio.NET. Кроме C# в VisualStudio.NET входят VisualBasic.NET и VisualC++.

Одна из причин разработки нового языка компанией Microsoft – это создание компонентно-ориентированного языка для новой платформы.NET. Другие языки были созданы до появления платформы .NET, язык же C# создавался специально под эту платформу и не несет с собой груза совместимости с предыдущими версиями языков.

C# – объектно-ориентированный язык программирования. Разработан в 1998–2001 годах группой инженеров под руководством Андерса Хейлсберга в компании Microsoft как язык разработки приложений для платформы Microsoft .NET Framework и впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270.

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

Переняв многое от своих предшественников – языков C++, Java, Delphi, Модула и Smalltalk – С#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем, например, C# не поддерживает множественное наследование классов (в отличие от C++).

Язык C# — это простой, но в то же время мощный, строго типизированный и объектно-ориентированный язык, позволяющий программистам создавать разнообразные приложения. Вкупе с платформой .NET Framework, Visual C# позволяет создавать приложения Windows, веб-службы, средства баз данных, компоненты, элементы управления и многое другое.

Программный комплекс включает в себя клиентское приложение и базу данных.

Клиентское приложение представляет собой Windows-приложение, реализованное средствами VisualStudio 2005 на языке программирования C#.

Пользователи не имеют прямого доступа к базе данных, всю работу с данными выполняет приложение. Взаимодействие с базой данных осуществляется через приложение. Клиентское приложение обращается к базе данных с некоторым запросом. База данных передает необходимую информацию в приложение, приложение обрабатывает информацию и возвращает результат работы. База данных играет роль сервера, который возлагает на себя все операции обработки информации, а клиентское приложение отображает результат выполнения запроса.

Программа оперирует большим объемом информации, поэтому для более быстрого и удобного доступа к данным и для обеспечения целостности данных, информацию необходимо хранить в базе данных. Для хранения и обработки информации будет использоваться система управления базами данных (СУБД) Microsoft Access 2010.

Microsoft Access – реляционная СУБД корпорации Microsoft. Имеет широкий спектр функций, включая связанные запросы, связь с внешними таблицами и базами данных. Благодаря встроенному языку VBA, в самом Access можно писать приложения, работающие с базами данных.

Основные компоненты MS Access:

– построитель таблиц;

– построитель экранных форм;

– построитель SQL-запросов (язык SQL в MS Access не соответствует стандарту ANSI);

– построитель отчётов, выводимых на печать.

Они могут вызывать скрипты на языке VBA, поэтому MS Access позволяет разрабатывать приложения и БД практически «с нуля» или написать оболочку для внешней БД.

Microsoft Jet Database Engine, которая используется в качестве движка базы данных MS Access является файл-серверной СУБД и потому применима лишь к приложениям, работающим с небольшими объемами данных и при небольшом числе пользователей, одновременно работающих с этим данными. Непосредственно в Access отсутствует ряд механизмов, необходимых в многопользовательских БД, таких, например, как триггеры.

**1.2. Основные принципы ООП**

Все языки объектно-ориентированного программирования, включая C#, основаны на трёх основополагающих концепциях - полиморфизм, инкапсуляция, наследование, рассмотрим их подробнее.

Полиморфизм означает "многоформенность". Несколько объектов с одним именем - имеют различную реализацию. Полиморфизм бывает статическим и динамическим.

В этой статье мы разберем некоторые виды полиморфизма.

Инкапсуляция - сокрытие методов и свойств. В C# при инкапсуляции используются следующие модификаторы:

* public - публичный (виден всем)
* private - приватный (виден только объектам класса)
* protected - защищенный (виден только объектам класса и наследуемым классам)

Наследование - свойство системы, позволяющее описать новый класс на основе уже существующего. Класс, от которого производится наследование, называется базовым или родительским. Новый класс - потомком или дочерним.

Объектно-ориентированное программирование основано на «трех китах» - трех важнейших принципах, придающих объектам новые свойства. Этими принципами являются инкапсуляция, наследование и полиморфизм.

Инкапсуляция есть объединение в единое целое данных и алгоритмов обработки этих данных. В рамках ООП данные называются полями объекта, а алгоритмы - объектными методами.

Инкапсуляция позволяет в максимальной степени изолировать объект от внешнего окружения. Она существенно повышает надежность разрабатываемых программ, т.к. локализованные в объекте алгоритмы обмениваются с программой сравнительно небольшими объемами данных, причем количество и тип этих данных обычно тщательно контролируются. В результате замена или модификация алгоритмов и данных, инкапсулированных в объект, как правило, не влечет за собой плохо прослеживаемых последствий для программы в целом (в целях повышения защищенности программ в ООП почти не используются глобальные переменные).

Другим немаловажным следствием инкапсуляции является легкость обмена объектами, переноса их из одной программы в другую. Можно сказать, что ООП «провоцирует» разработку библиотек объектов, таких как Turbo Vision.

Наследование есть свойство объектов порождать своих потомков. Объект-потомок автоматически наследует от родителя все поля и методы, может дополнять объекты новыми полями и заменять (перекрывать) методы родителя или дополнять их.

Принцип наследования решает проблему модификации свойств объекта и придает ООП в целом исключительную гибкость. При работе с объектами программист обычно подбирает объект, наиболее близкий по своим свойствам для решения конкретной задачи, и создает одного или нескольких потомков от него, которые «умеют» делать то, что не реализовано в родителе.

Последовательное проведение в жизнь принципа «наследуй и изменяй» хорошо согласуется с поэтапным подходом к разработке крупных программных проектов и во многом стимулирует такой подход.

Полиморфизм - это свойство родственных объектов (т.е. объектов, имеющих одного общего родителя) решать схожие по смыслу проблемы разными способами. В рамках ООП поведенческие свойства объекта определяются набором входящих в него методов. Изменяя алгоритм того или иного метода в потомках объекта, программист может придавать этим потомкам отсутствующие у родителя специфические свойства. Для изменения метода необходимо перекрыть его в потомке, т.е. объявить в потомке одноименный метод и реализовать в нем нужные действия. В результате в объекте-родителе и объекте-потомке будут действовать два одноименных метода, имеющие разную алгоритмическую основу и, следовательно, придающие объектам разные свойства. Это и называется полиморфизмом объектов.

**1.3. Визуальные компоненты С#**

Элементы управления являются посредниками между приложением и пользователем. Они составляют так называемый графический интерфейс пользователя (GUI - Graphics User Interface). Их задача состоит в отображении информации для пользователя, обеспечении удобного ввода и генерации событий, отрабатывающих предпочтения пользователя.

Как правило, библиотечных элементов управления достаточно для решения большинства задач, но иногда программисту может потребоваться нестандартный элемент управления. В таком случае он может разработать свой элемент управления с нуля или расширить стандартный элемент. Такой элемент управления называют пользовательским.

Механизм наследования C# предоставляет широкие возможности модификации существующих элементов управления на уровне расширения классов. Беря за основу библиотечный элемент управления, мы можем использовать его в качестве базового и внутри производного класса (расширения) переопределять виртуальные методы и свойства, добавлять новые поля, свойства, события.

На рис. 1 представлены «Общие элементы управления», многие из них были использованы при разработке проекта «Аэропорт».

**ОБЩИЕ ЭЛЕМЕНТЫ УПРАВЛЕНИЯ**

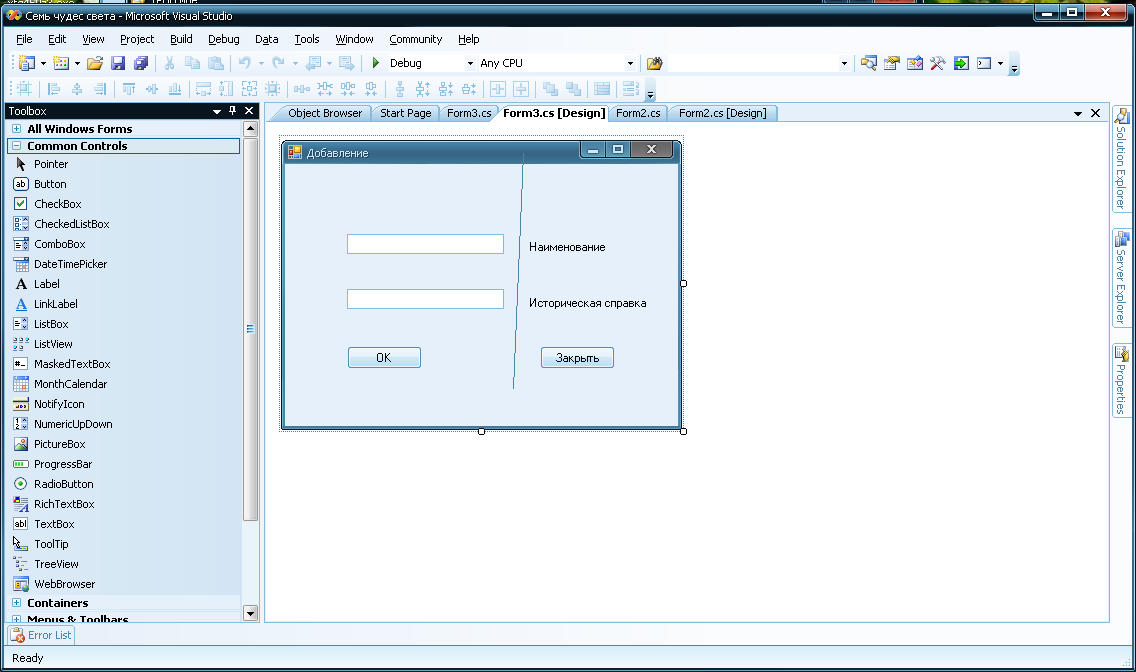


Рис. 1

Одними из наиболее распространенных компонентов в программировании .NET Framework являются визуальные элементы управления, добавляемые в Windows Forms, такие как [Элемент управления Button (Windows Forms)](http://msdn.microsoft.com/ru-ru/library/s0scsyfb(v=vs.90).aspx), [Элемент управления ComboBox (Windows Forms)](http://msdn.microsoft.com/ru-ru/library/9h80cyht(v=vs.90).aspx) и другие. В число не визуальных компонентов входят [Timer Control](http://msdn.microsoft.com/ru-ru/library/hby83ha1(v=vs.90).aspx), [SerialPort](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport(v=vs.90).aspx) и [ServiceController](http://msdn.microsoft.com/ru-ru/library/system.serviceprocess.servicecontroller(v=vs.90).aspx).

**Объект Form**

Форма является основным компонентом визуального приложения. На форме располагаются остальные элементы приложения (визуальные и не визуальные, такие например как таймер), то есть форма является контейнером объектов.

Основные свойства объекта Form:

1. Name - имя объекта. По умолчанию - Form1. В программе по имени осуществляется доступ к свойствам объекта. Имя можно изменять. Так же как и для имён переменных, имя объекта должно начинаться с латинской буквы. Дальше могут следовать как буквы, так и цифры. Например: MainForm; Form\_2115; F10head.
2. BackColor - цвет заливки (фон) объекта.
3. Cursor - вид курсора мыши отображаемого на фоне окна.
4. Font - свойство Font задаёт тип шрифта - имя, размер, начертание. Шрифт, заданный в форме, переходит на остальные объекты, расположенные на форме (наследование).
5. ForeColor - цвет отображения (текста) на элементах (визуальных объектах) формы.
6. FormBorderStyle - тип бордюра (рамки) формы:

* Sizable - (по умолчанию) размер формы можно изменять в процессе выполнения программы;
* None - форма не имеет бордюра, строки заголовка и системных кнопок;
* FixedSingle - бордюр одинарный, размер формы изменять нельзя;
* Fixed3D - то же, что и FixedSingle, но бордюр объёмный;
* FixedDialog - то же, что и FixedSingle и предназначена для конструирования
* диалоговых панелей - не имеет системной кнопки в строке заголовка;
* FixedToolWindow - окно для изготовления инструментальных панелей, имеет только одну системную кнопку "Закрыть";
* SizableToolWindow - то же, что и FixedToolWindow, но размер окна можно изменять.

1. Icon - рисунок (иконка в формате ico) появляющийся в строке заголовка формы.
2. Opacity - непрозрачность. Уменьшение процентного показателя ведёт к появлению эффекта прозрачности формы вместе с её содержимым.
3. Size - размер окна (в пикселях) - ширина; высота.
4. StartPosition - положение окна при запуске программы:

* WindowDefaultLocation - окно на экране в позиции по умолчанию;
* CenterScreen - окно в центре экрана.

1. WindowState - поведение окна при запуске:

* normal - нормальное отображение;
* Minimized - окно минимизировано на панель инструментов Windows;
* Maximized - окно развёрнуто на весь экран.

**Объект Panel**

Объект панель является контейнером для расположенных на нём визуальных объектов. Панель удобно использовать для группирования элементов интерфейса программы, например изготовление инструментальной панели, панели работы с файлами и тому подобное. Очень полезным свойством панели, является свойство Dock.

**ОБЪЕКТ PANEL И ЕГО СВОЙСТВА**

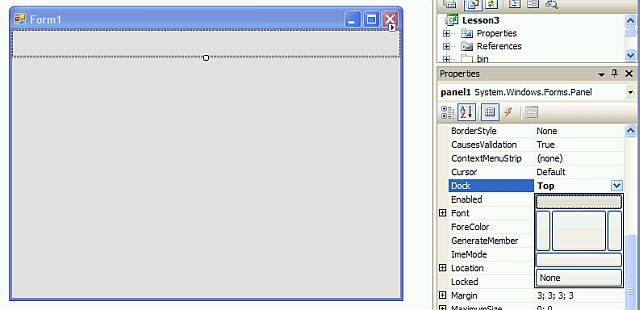


Рис. 2

Используя это свойство панель можно выровнять по верхнему, нижнему, правому или левому краям окна, а так же по центру. Причём при работе программы и изменению размеров окна, панель будет синхронно изменять свои размеры в зависимости от выбранного режима выравнивания. На рисунке 2 показана панель, выровненная по верхнему краю окна.

Панель не несёт каких-либо функций управления программой, а используется как элемент оформления.

У панели есть свойство Visible (видимость / невидимость панели). Это свойство удобно использовать, когда нужно в процессе исполнения программы прятать или наоборот показывать необходимые инструменты, расположенные на определённых панелях.

**Объект dataGridView**

Настраиваемый элемент для отображения данных.

Свойства dataGridView:

1. AccessibleRole - возвращает или задает роль элемента управления в поддержке специальных возможностей.
2. AdvancedRowHeadersBorderStyle - возвращает стиль границы ячеек заголовков строк DataGridView.
3. AdvancedColumnHeadersBorderStyle - возвращает стиль границы ячейки заголовков строк DataGridView.

**2. НАЗНАЧЕНИЕ ПРОГРАММНОГО ИЗДЕЛИЯ**

**2.1. Описание предметной области и назначения программы**

Воздушный транспорт — понятие, включающее как собственно [воздушные суда](http://ru.wikipedia.org/wiki/%D0%92%D0%BE%D0%B7%D0%B4%D1%83%D1%88%D0%BD%D0%BE%D0%B5_%D1%81%D1%83%D0%B4%D0%BD%D0%BE), так и необходимую для их эксплуатации инфраструктуру: [аэропорты](http://ru.wikipedia.org/wiki/%D0%90%D1%8D%D1%80%D0%BE%D0%BF%D0%BE%D1%80%D1%82), диспетчерские и технические службы.

**ПАССАЖИРСКИЙ РЕАКТИВНЫЙ САМОЛЁТ AIRBUS A319**

[](http://commons.wikimedia.org/wiki/File:Airbus_A319-100LR_Privatair_D-APAD.jpg?uselang=ru)

Рис. 3

Воздушный транспорт — самый быстрый и в то же время самый дорогой вид транспорта. Основная сфера применения воздушного транспорта — пассажирские перевозки на расстояниях свыше тысячи километров. Также осуществляются и грузовые перевозки, но их доля очень низка. В основном авиатранспортом перевозят скоропортящиеся продукты и особо ценные грузы, а также [почту](http://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%87%D1%82%D0%B0). Во многих труднодоступных районах (в горах, районах Крайнего Севера) воздушному транспорту нет альтернатив. В таких случаях, когда в месте посадки отсутствует аэродром (например, доставка научных групп в труднодоступные районы) используют не [самолёты](http://ru.wikipedia.org/wiki/%D0%A1%D0%B0%D0%BC%D0%BE%D0%BB%D1%91%D1%82), а [вертолёты](http://ru.wikipedia.org/wiki/%D0%92%D0%B5%D1%80%D1%82%D0%BE%D0%BB%D1%91%D1%82), которые не нуждаются в посадочной полосе. Большая проблема современных самолётов — шум, производимый ими при взлёте, который значительно портит качество жизни обитателей расположенных рядом с аэропортами районов.

**ДИРИЖАБЛЬ В-6 «ОСОАВИАХИМ» 30-Е ГОДЫ, СССР**

[](http://commons.wikimedia.org/wiki/File:W-6_Ossoaviachim_wiki.jpg?uselang=ru)

Рис. 4

В настоящее время понятия [авиация](http://ru.wikipedia.org/wiki/%D0%90%D0%B2%D0%B8%D0%B0%D1%86%D0%B8%D1%8F) и воздушный транспорт фактически стали синонимами, так как воздушные перевозки осуществляются исключительно [воздушными судами](http://ru.wikipedia.org/wiki/%D0%92%D0%BE%D0%B7%D0%B4%D1%83%D1%88%D0%BD%D0%BE%D0%B5_%D1%81%D1%83%D0%B4%D0%BD%D0%BE) тяжелее [воздуха](http://ru.wikipedia.org/wiki/%D0%92%D0%BE%D0%B7%D0%B4%D1%83%D1%85). Однако первые воздушные суда были легче воздуха. В 1709 году был запущен первый [воздушный шар](http://ru.wikipedia.org/wiki/%D0%90%D1%8D%D1%80%D0%BE%D1%81%D1%82%D0%B0%D1%82). Впрочем, воздушные шары были неуправляемы. К концу XIX века доминировать в воздухе стали огромные воздушные корабли — [дирижабли](http://ru.wikipedia.org/wiki/%D0%94%D0%B8%D1%80%D0%B8%D0%B6%D0%B0%D0%B1%D0%BB%D1%8C). Их золотой век пришёлся на первую половину XX века, когда пассажирские дирижабли совершали регулярные перелёты между Европой и [Америкой](http://ru.wikipedia.org/wiki/%D0%90%D0%BC%D0%B5%D1%80%D0%B8%D0%BA%D0%B0). Эпоха дирижаблей кончилась в [1937 году](http://ru.wikipedia.org/wiki/1937_%D0%B3%D0%BE%D0%B4), когда прямо в [аэропорту](http://ru.wikipedia.org/wiki/%D0%90%D1%8D%D1%80%D0%BE%D0%BF%D0%BE%D1%80%D1%82) [Нью-Йорка](http://ru.wikipedia.org/wiki/%D0%9D%D1%8C%D1%8E-%D0%99%D0%BE%D1%80%D0%BA) сгорел немецкий пассажирский дирижабль-лайнер «[Гинденбург](http://ru.wikipedia.org/wiki/%D0%93%D0%B8%D0%BD%D0%B4%D0%B5%D0%BD%D0%B1%D1%83%D1%80%D0%B3_(%D0%B4%D0%B8%D1%80%D0%B8%D0%B6%D0%B0%D0%B1%D0%BB%D1%8C))». В конце XX века возобновился интерес к дирижаблям: теперь вместо взрывоопасного [водорода](http://ru.wikipedia.org/wiki/%D0%92%D0%BE%D0%B4%D0%BE%D1%80%D0%BE%D0%B4) применяется инертный [гелий](http://ru.wikipedia.org/wiki/%D0%93%D0%B5%D0%BB%D0%B8%D0%B9), дирижабли хоть и намного медленнее самолётов, но зато намного экономичнее. Тем не менее до сих пор сфера их применения остаётся маргинальной: рекламные и увеселительные полёты, наблюдение за дорожным движением. Дирижабли также предлагаются в качестве климатически приемлемой альтернативы самолётам.

* Транспортные средства: [самолёты](http://ru.wikipedia.org/wiki/%D0%A1%D0%B0%D0%BC%D0%BE%D0%BB%D1%91%D1%82) и [вертолёты](http://ru.wikipedia.org/wiki/%D0%92%D0%B5%D1%80%D1%82%D0%BE%D0%BB%D1%91%D1%82)
* Пути сообщения: [воздушные коридоры](http://ru.wikipedia.org/wiki/%D0%92%D0%BE%D0%B7%D0%B4%D1%83%D1%88%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D1%80%D0%B8%D0%B4%D0%BE%D1%80)
* Сигнализация и управление: [авиамаяки](http://ru.wikipedia.org/wiki/%D0%90%D0%B2%D0%B8%D0%B0%D0%BC%D0%B0%D1%8F%D0%BA), [диспетчерская служба](http://ru.wikipedia.org/w/index.php?title=%D0%94%D0%B8%D1%81%D0%BF%D0%B5%D1%82%D1%87%D0%B5%D1%80%D1%81%D0%BA%D0%B0%D1%8F_%D1%81%D0%BB%D1%83%D0%B6%D0%B1%D0%B0_(%D0%B2%D0%BE%D0%B7%D0%B4%D1%83%D1%88%D0%BD%D1%8B%D0%B9_%D1%82%D1%80%D0%B0%D0%BD%D1%81%D0%BF%D0%BE%D1%80%D1%82)&action=edit&redlink=1)
* Транспортные узлы: [аэропорты](http://ru.wikipedia.org/wiki/%D0%90%D1%8D%D1%80%D0%BE%D0%BF%D0%BE%D1%80%D1%82)

### Аэровокзальный комплекс (пассажирский терминал)

Включает в себя собственно аэровокзал, предназначенный для обслуживания [пассажиров](http://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%81%D1%81%D0%B0%D0%B6%D0%B8%D1%80). В аэровокзале базируются большинство служб, обслуживающих пассажиров от момента входа на территорию аэропорта до вылета и от момента подачи [трапа](http://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BB%D0%B5%D1%81%D0%BA%D0%BE%D0%BF%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D1%82%D1%80%D0%B0%D0%BF) к [самолёту](http://ru.wikipedia.org/wiki/%D0%A1%D0%B0%D0%BC%D0%BE%D0%BB%D1%91%D1%82) до покидания аэропорта:

* представительства авиакомпаний;
* служба организации пассажирских перевозок;
* службы безопасности;
* багажная служба;
* службы пограничного, иммиграционного и [таможенного](http://ru.wikipedia.org/wiki/%D0%A2%D0%B0%D0%BC%D0%BE%D0%B6%D0%BD%D1%8F) контроля;
* различные организации и предприятия, обеспечивающие [отдых](http://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%B4%D1%8B%D1%85), питание, досуг пассажиров и т. п.: [рестораны](http://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%81%D1%82%D0%BE%D1%80%D0%B0%D0%BD) и [кафе](http://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%84%D0%B5), точки торговли периодикой и [сувенирами](http://ru.wikipedia.org/wiki/%D0%A1%D1%83%D0%B2%D0%B5%D0%BD%D0%B8%D1%80), [магазины](http://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D0%B3%D0%B0%D0%B7%D0%B8%D0%BD_(%D1%82%D0%BE%D1%80%D0%B3%D0%BE%D0%B2%D0%BB%D1%8F)), и т. д.

### Грузовой комплекс

Принимает к отправке, оформляет, обрабатывает, загружает на борт [воздушных судов](http://ru.wikipedia.org/wiki/%D0%92%D0%BE%D0%B7%D0%B4%D1%83%D1%88%D0%BD%D0%BE%D0%B5_%D1%81%D1%83%D0%B4%D0%BD%D0%BE) [груз](http://ru.wikipedia.org/wiki/%D0%93%D1%80%D1%83%D0%B7) и [почту](http://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%87%D1%82%D0%B0). Оснащается крытым отапливаемым [складом](http://ru.wikipedia.org/wiki/%D0%A1%D0%BA%D0%BB%D0%B0%D0%B4), средствами доставки и механизированной погрузки-разгрузки, средствами обработки груза «в навал» и в [контейнерах](http://ru.wikipedia.org/wiki/ISO-%D0%BA%D0%BE%D0%BD%D1%82%D0%B5%D0%B9%D0%BD%D0%B5%D1%80).

**Класс аэропорта**

Класс аэропорта определяется годовым объёмом пассажирских перевозок (пассажирообменом), то есть суммарным количеством всех прилетающих и вылетающих пассажиров, включая транзитных пассажиров (с пересадкой из одного воздушного судна в другое).

Таблица 1.

**КЛАССИФИКАЦИЯ АЭРОПОРТОВ В ЗАВИСИМОСТИ**

**ОТ ГОДОВОГО ОБЪЁМА ПАССАЖИРСКИХ ПЕРЕВОЗОК**

|  |  |
| --- | --- |
| **Класс аэропорта** | **Годовой объём пассажирских перевозок,**  **тыс. человек** |
| I | 10000—7000 |
| II | 7000—4000 |
| III | 4000—2000 |
| IV | 2000—500 |
| V | 500—100 |

Аэропорты с годовым объёмом перевозок более 10 млн чел. относятся к внеклассным, а с годовым объёмом перевозок менее 100 тыс. чел. — к неклассифицированным.

Неклассифицированные аэропорты [местных воздушных линий](http://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%81%D1%82%D0%BD%D1%8B%D0%B5_%D0%B2%D0%BE%D0%B7%D0%B4%D1%83%D1%88%D0%BD%D1%8B%D0%B5_%D0%BB%D0%B8%D0%BD%D0%B8%D0%B8) располагаются на [аэродромах](http://ru.wikipedia.org/wiki/%D0%90%D1%8D%D1%80%D0%BE%D0%B4%D1%80%D0%BE%D0%BC) 3 или 4 класса, с искусственным или грунтовым покрытием [ВПП](http://ru.wikipedia.org/wiki/%D0%92%D0%B7%D0%BB%D1%91%D1%82%D0%BD%D0%BE-%D0%BF%D0%BE%D1%81%D0%B0%D0%B4%D0%BE%D1%87%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BE%D0%BB%D0%BE%D1%81%D0%B0) (искусственное покрытие ВПП в аэропортах 4 класса нередко автодорожное).

**Сотрудники аэропорта**

Среди рабочих средней квалификации: водители, диспетчеры транспортного цеха, диспетчеры аэровокзала, маляры, агенты отдела материально-технического снабжения, грузчики, техник связи.

На этапе анализа предметной области было выделено 4 сущности: "Самолеты", "Рейсы", "Пассажиры", "Вылеты". Инфологические модели данных сущностей представлены на рис. 5

**ИНФОЛОГИЧЕСКАЯ МОДЕЛЬ ОБЪЕКТА «САМОЛЕТЫ»**

Самолеты

Номер самолета

Тип самолета

Количество мест

Название авиакомпании

Рис. 5

**S**

**S**

**S**

**D**

**ИНФОЛОГИЧЕСКАЯ МОДЕЛЬ ОБЪЕКТА «РЕЙСЫ»**

Рейсы

Номер рейса

Название рейса

Номер самолета

Время вылета

Рис. 6

**S**

**D**

**S**

**D**

Стоимость

**D**

**ИНФОЛОГИЧЕСКАЯ МОДЕЛЬ ОБЪЕКТА «ПАССАЖИРЫ»**

Пассажиры

Номер паспорта

ФИО

Номер рейса

Номер места

Рис. 7

**S**

**S**

**S**

**S**

**ИНФОЛОГИЧЕСКАЯ МОДЕЛЬ ОБЪЕКТА «ВЫЛЕТЫ»**

Вылеты

Номер рейса

Номер самолета

Дата вылета

Рис. 8

**S**

**S**

**S**

На этапе логического проектирования модели базы данных происходит описание перевода инфологической модели базы данных в абстрактные таблицы. Результатом этого этапа являются таблица 2, таблица 3, таблица 4, таблица 5, как модели реальной базы данных.

Таблица2.

**ДАТАЛОГИЧЕСКАЯ МОДЕЛЬ ТАБЛИЦЫ БД «САМОЛЕТЫ»**

|  |  |  |
| --- | --- | --- |
| *Имя поля* | *Тип* | *Размер* |
| № самолета | Числовой | 10 |
| Тип самолета | Текстовый | 10 |
| Кол-во мест | Числовой | 10 |
| Название авиакомпании | Текстовый | 20 |

Таблица 3.

**ДАТАЛОГИЧЕСКАЯ МОДЕЛЬ ТАБЛИЦЫ БД «РЕЙСЫ»**

|  |  |  |  |
| --- | --- | --- | --- |
| *Имя поля* | *Тип* | *Размер* | *Число знаков после запятой* |
| № рейса | Числовой | 10 | - |
| Название рейса | Текстовый | 20 | - |
| № самолета | Числовой | 10 | - |
| Время вылета | Дата/Время | 10 | - |
| Стоимость | Денежный | 10 | 2 |

Таблица 4.

**ДАТАЛОГИЧЕСКАЯ МОДЕЛЬ ТАБЛИЦЫ БД «ПАССАЖИРЫ»**

|  |  |  |
| --- | --- | --- |
| *Имя поля* | *Тип* | *Размер* |
| № паспорта | Числовой | 10 |
| ФИО | Текстовый | 50 |
| № рейса | Числовой | 10 |
| № места | Числовой | 10 |

Таблица 5.

**ДАТАЛОГИЧЕСКАЯ МОДЕЛЬ ТАБЛИЦЫ БД «ВЫЛЕТЫ»**

|  |  |  |
| --- | --- | --- |
| *Имя поля* | *Тип* | *Размер* |
| № рейса | Числовой | 10 |
| Дата вылета | Дата/Время | 10 |
| № самолета | Числовой | 10 |

Для создания базы данных была выбрана система управления базами данных Microsoft Access. MS Access входит в состав самого популярного пакета Microsoft Office. Основные преимущества: знаком многим конечным пользователям и обладает высокой устойчивостью данных, позволяет готовить отчеты из баз данных различных форматов.

Предназначен для создания отчетов произвольной формы на основании различных данных. Microsoft Access обеспечивает средства для создания клиентских частей в приложениях «клиент-сервер», которые сочетают в себе средства просмотра, графический интерфейс и средства построения запросов.

**СТРУКТУРА ТАБЛИЦЫ «ВЫЛЕТЫ» В РЕЖИМЕ КОНСТРУКТОРА**

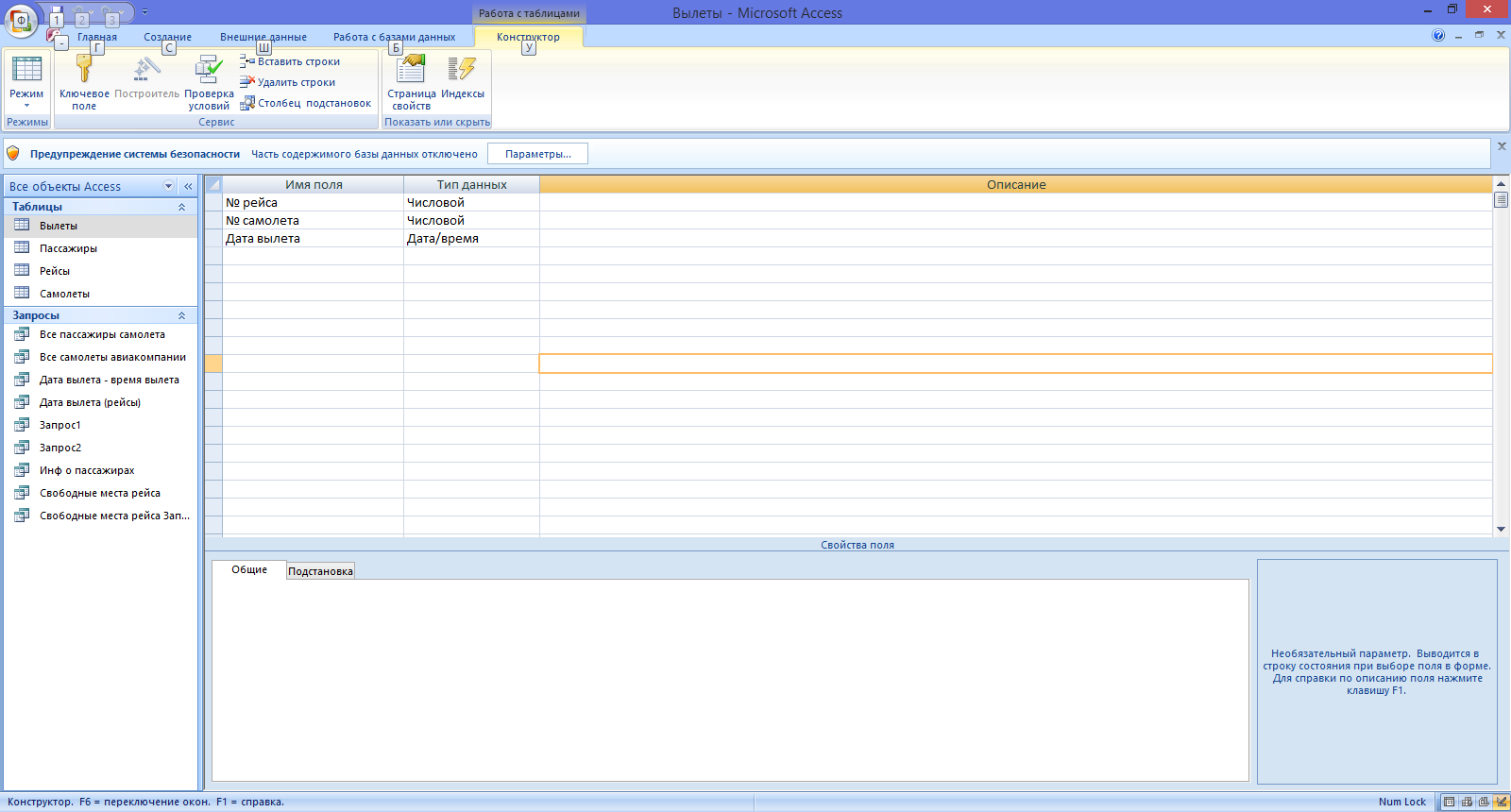


Рис. 9

Таблица Вылеты содержит поле № рейса для идентификации вылета, поле № самолета для идентификации самолета, на котором осуществляется вылет, поле дата вылета содержит информацию о дате и времени вылета. Данная таблица не содержит ключевых полей.

**СТРУКТУРА ТАБЛИЦЫ «ПАССАЖИРЫ» В РЕЖИМЕ КОНСТРУКТОРА**

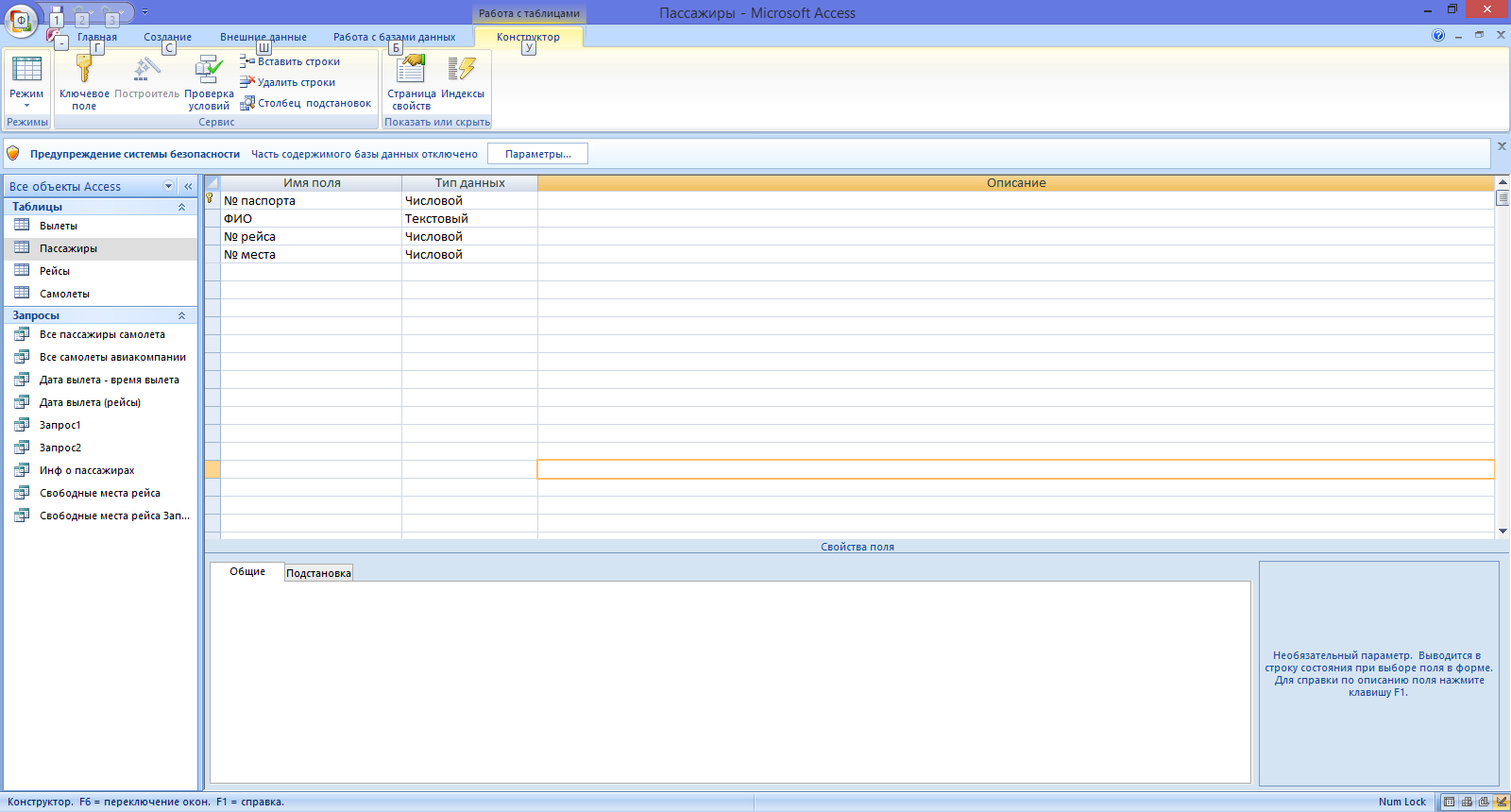
****

Рис. 10

Таблица Пассажиры содержит ключевое поле № паспорта для идентификации пассажиров, поле ФИО для идентификации пассажира, поля № рейса и № места для хранения информации о купленном пассажиром билете.

**СТРУКТУРА ТАБЛИЦЫ «РЕЙСЫ» В РЕЖИМЕ КОНСТРУКТОРА**

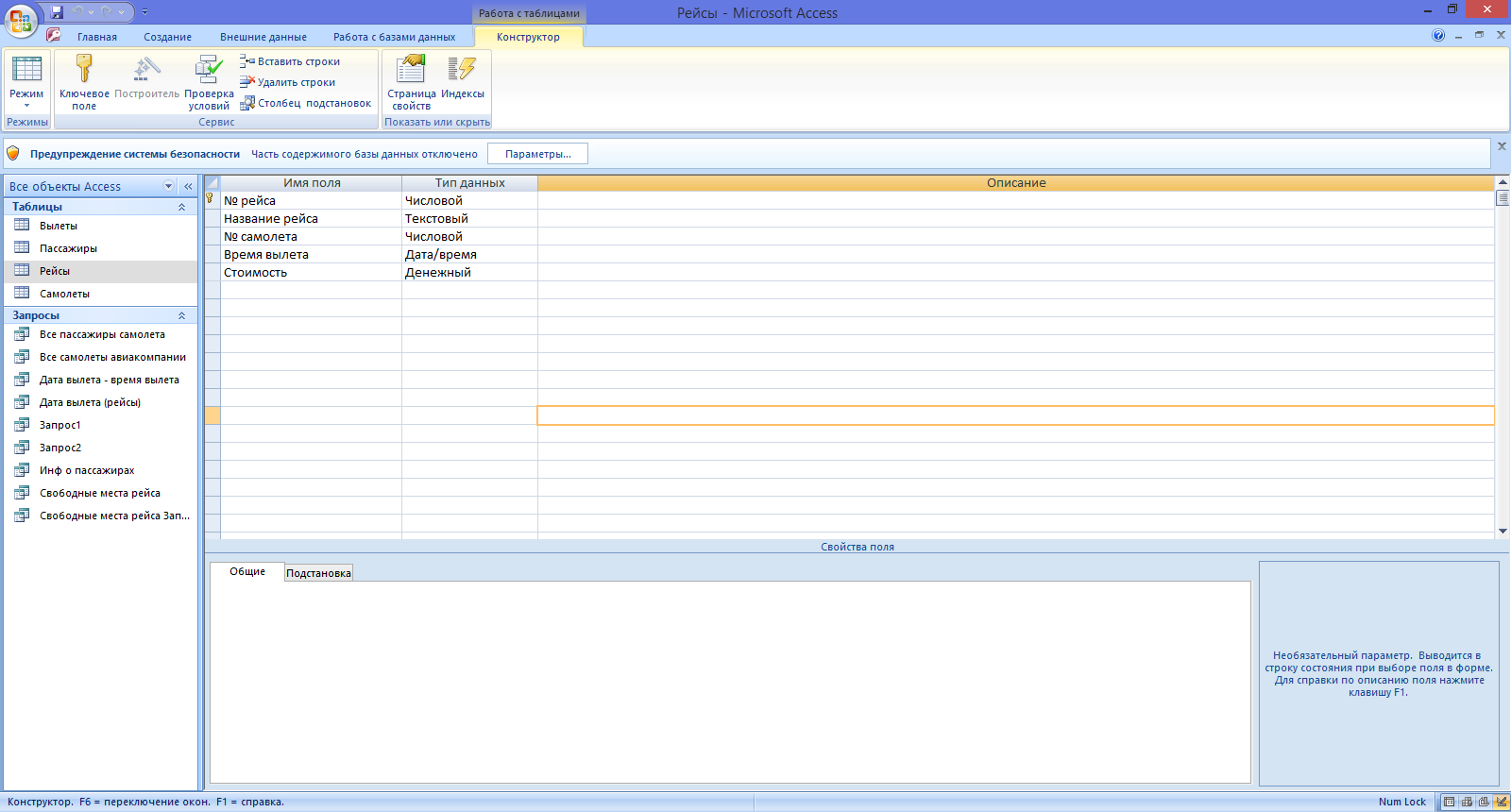


Рис. 11

Таблица Рейсы содержит ключевое поле № рейса для идентификации рейса, поле Название рейса для идентификации вылета, поле № самолета для идентификации самолета, осуществляющего рейс, поле Стоимость для хранения информации о цене билета.

**СТРУКТУРА ТАБЛИЦЫ «САМОЛЕТЫ» В РЕЖИМЕ КОНСТРУКТОРА**

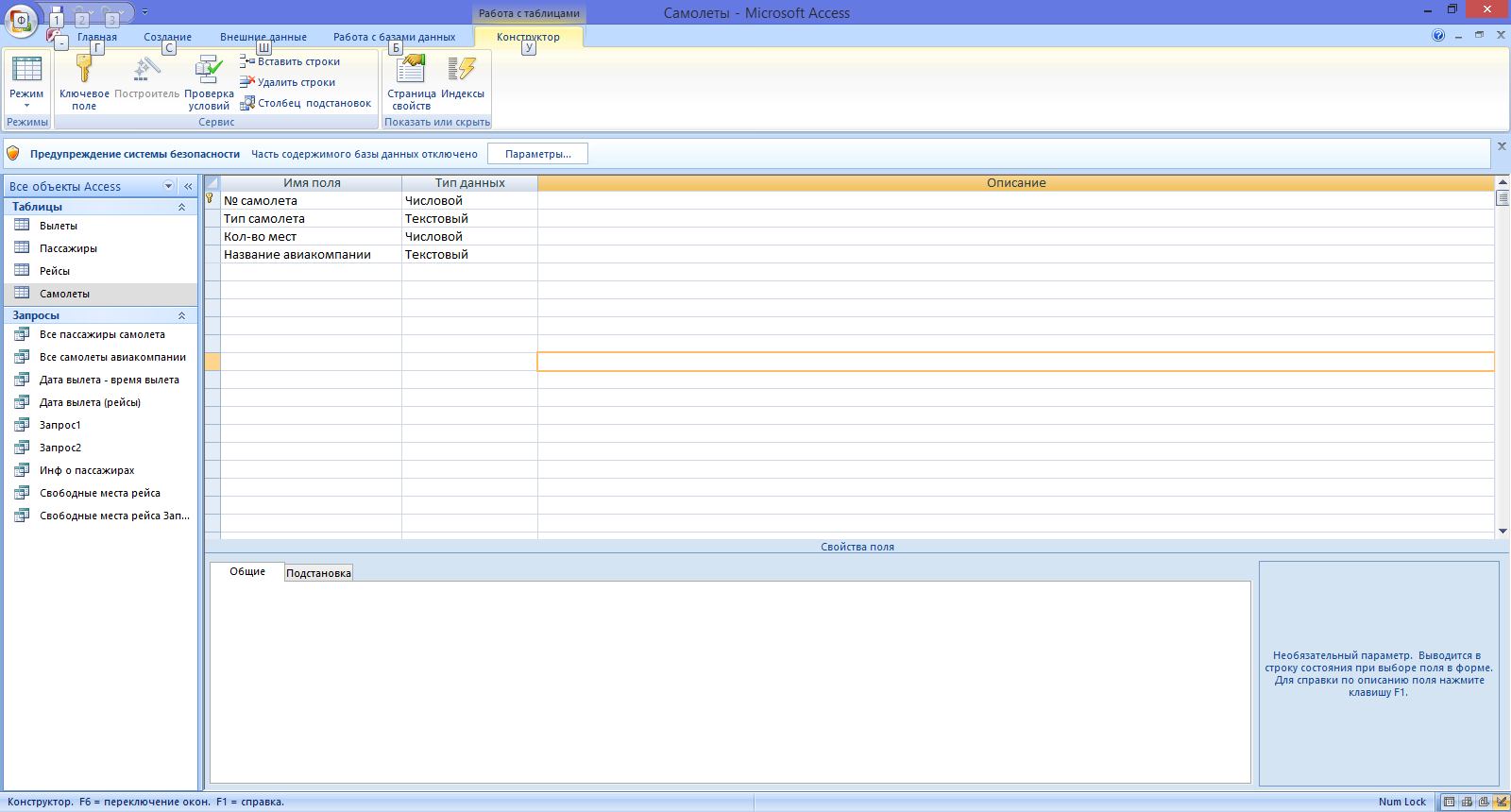


Рис. 12

Таблица Самолеты содержит ключевое поле № самолета для идентификации самолета, поле Тип самолета для описания самолета, поле Кол-во мест для указания количества мест в самолете, поле Название авиакомпании для хранения информации о названии авиакомпании, владеющей самолетом.

Ниже представлена схема данных предметной области «Аэропорт» в СУБД Microsoft Access.

**СХЕМА ДАННЫХ**

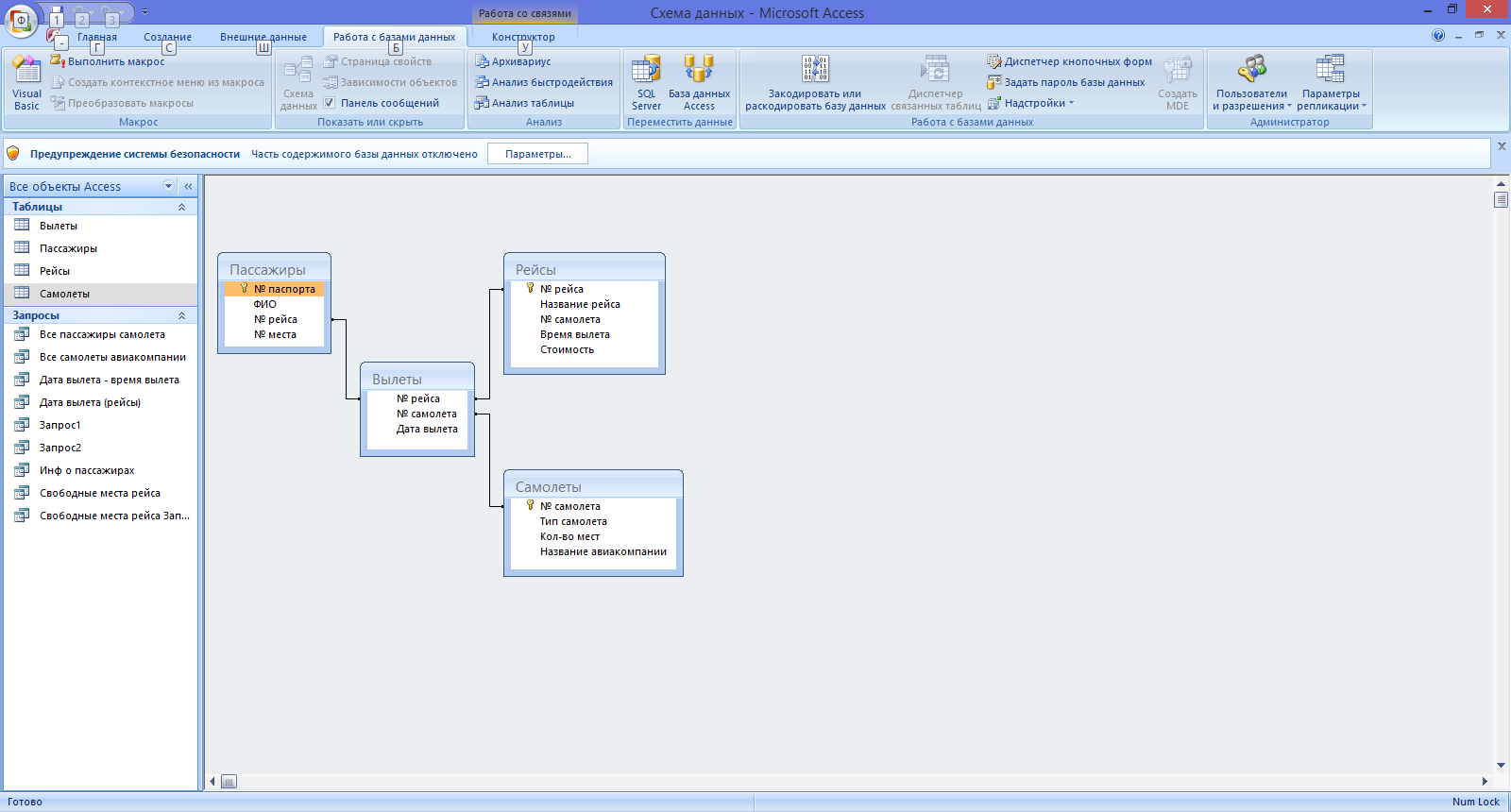


Рис. 13

На рисунках 14, 15, 16, 17 представлены все таблицы из базы данных.

**ТАБЛИЦА «ВЫЛЕТЫ»**

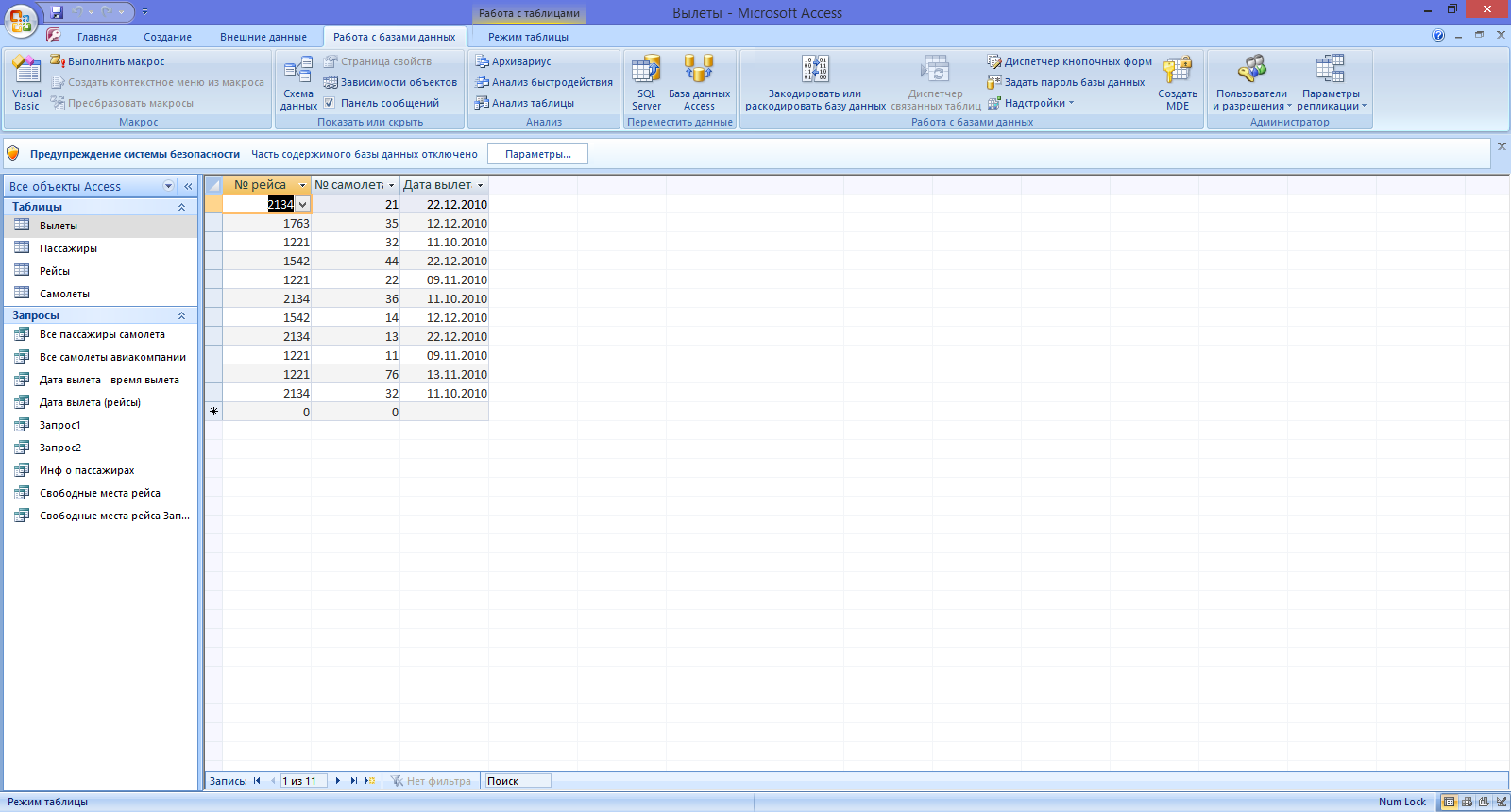


Рис. 14

**ТАБЛИЦА «ПАССАЖИРЫ»**

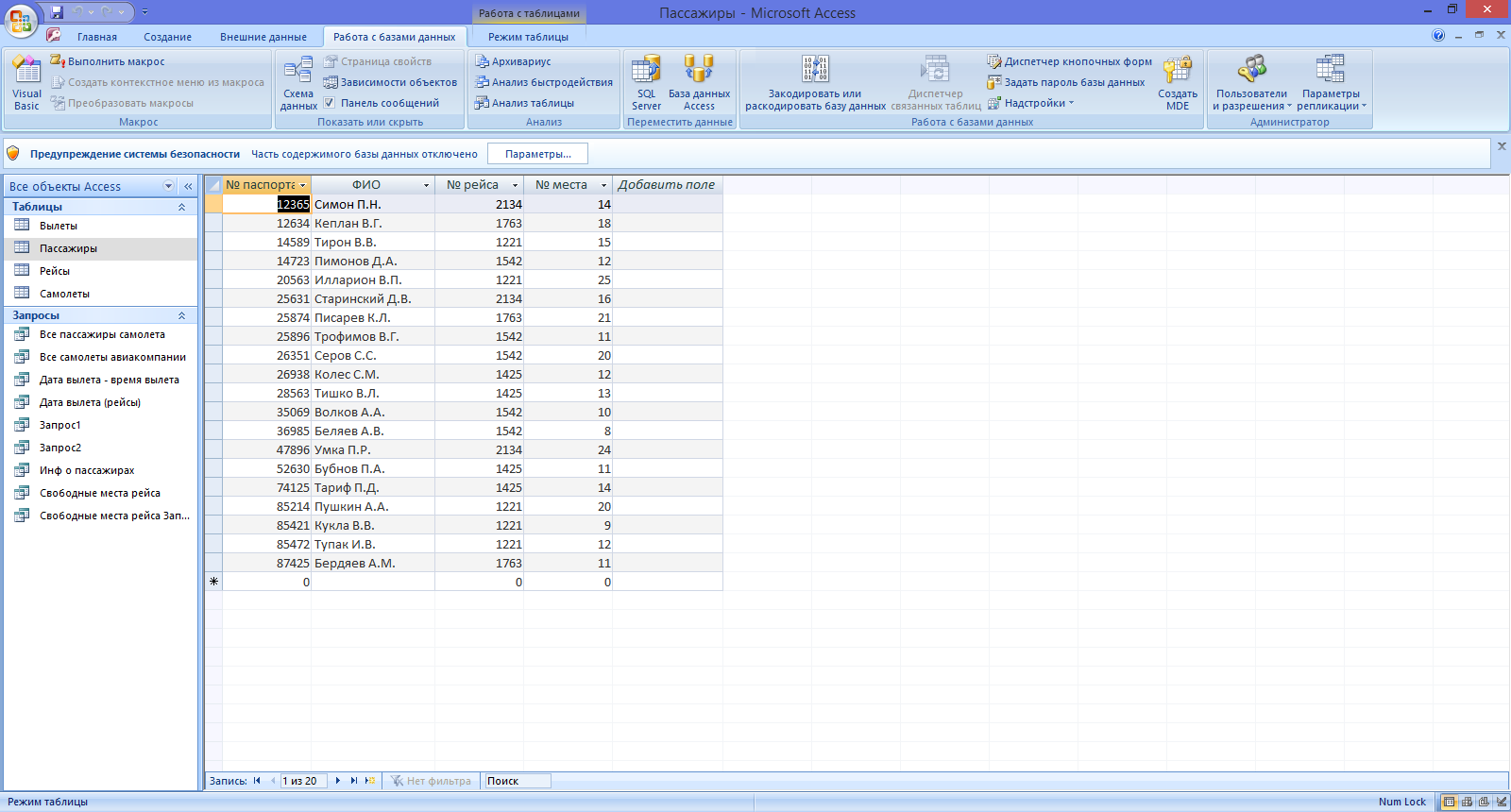


Рис. 15

**ТАБЛИЦА «РЕЙСЫ»**

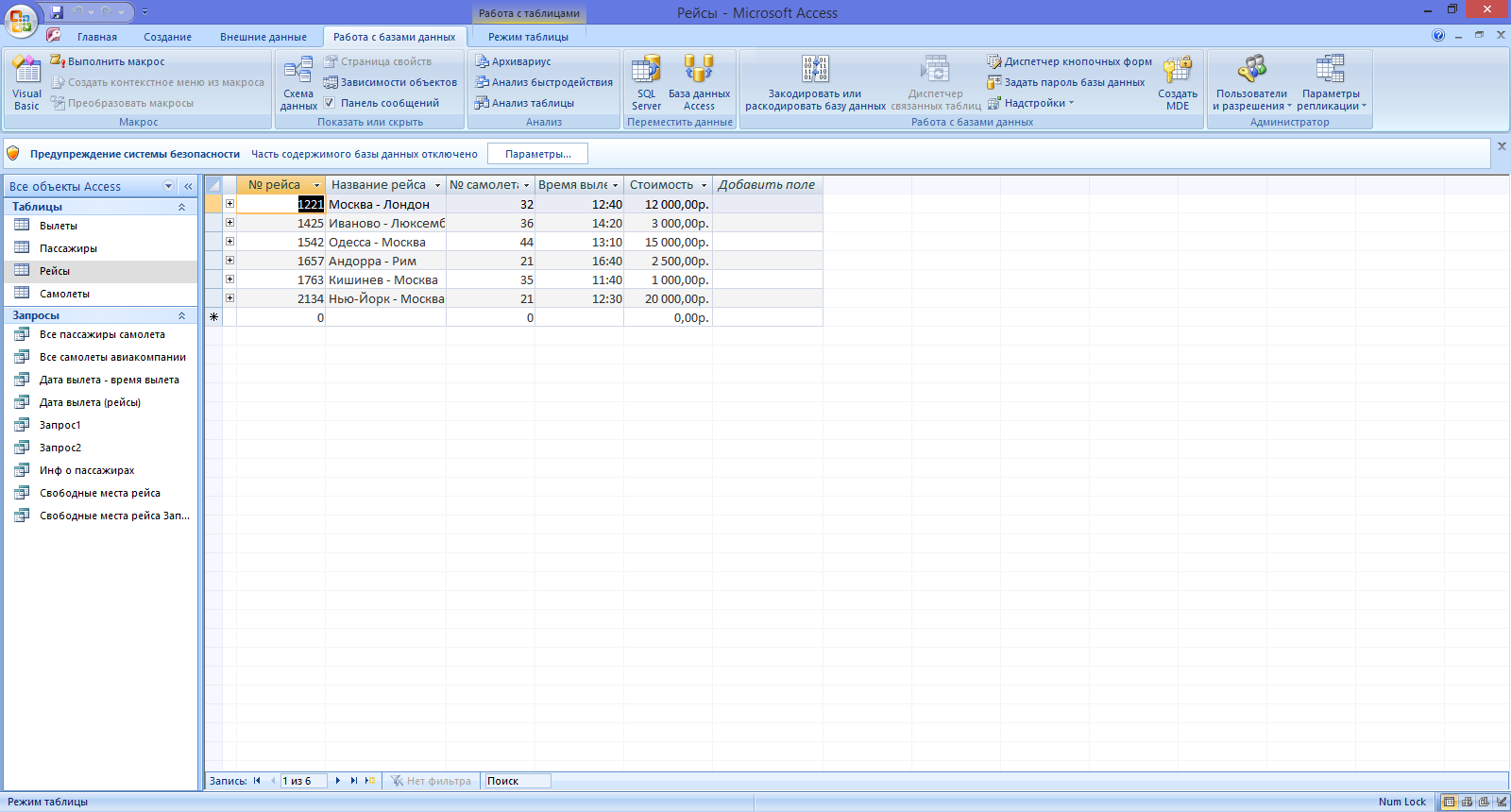


Рис. 16

**ТАБЛИЦА «САМОЛЕТЫ»**

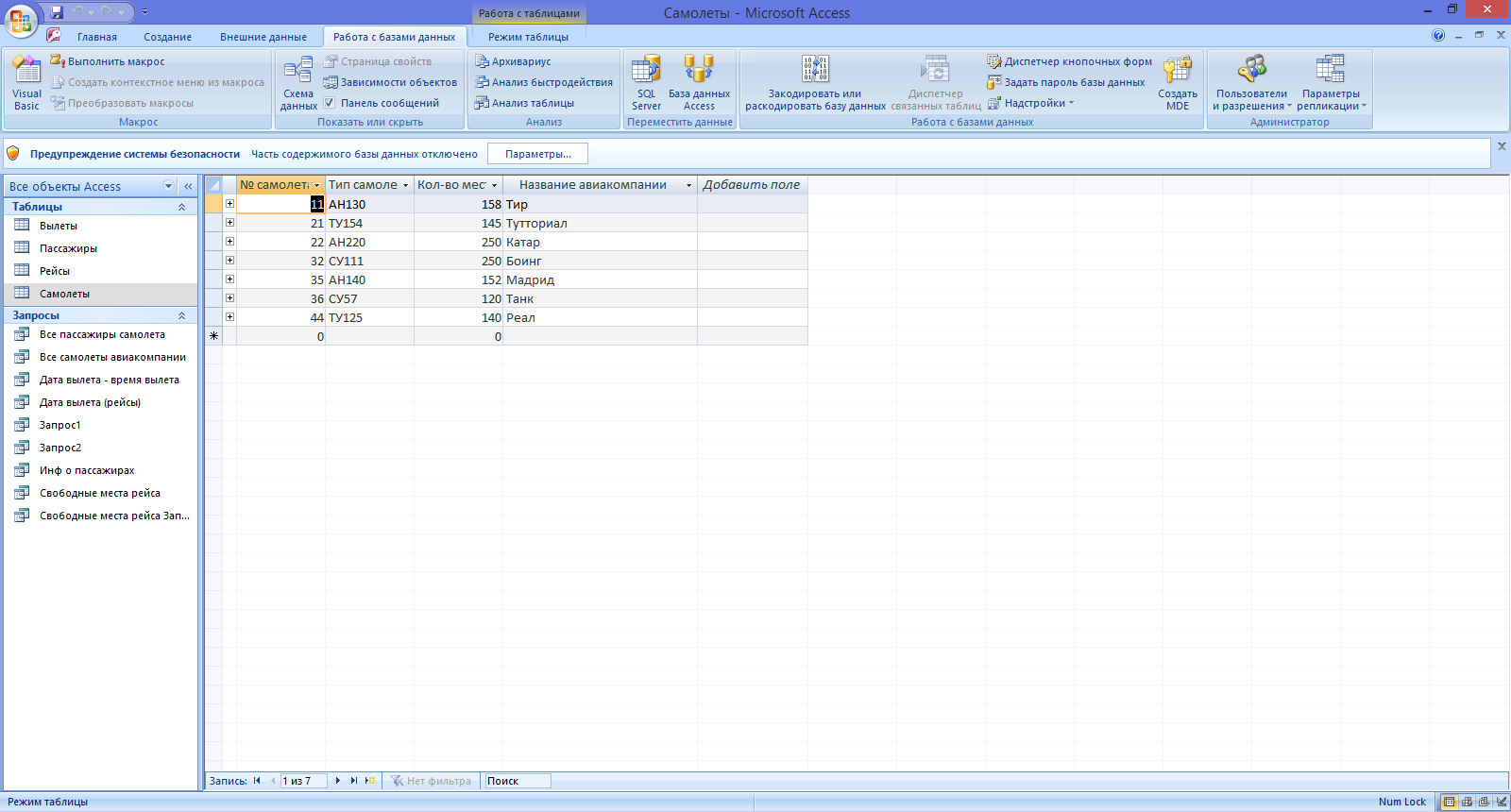


Рис. 17

**2.2. Описание задач программного продукта для разных групп пользователей**

В данном программном продукте предусмотрено три вида пользователей: администратор, пользователь и эксперт.

Администратор и эксперт имеют пароль для входа в программу. Пользователь заходит без пароля. Пользователь имеет право просмотра и поиска информации.

Пользователь не имеет прав доступа к редактированию и удалению информации. У администратора и эксперта такие права есть. Разграничение прав доступа осуществляется с помощью авторизации. Такое ограничение прав защищает от несанкционированного доступа к информации, предотвращает изменение данных.

**2.3. Описание характеристик проекта**

В таблице 6 показаны затраты на внедрение системы.

Таблица 6.

**ЗАТРАТЫ НА ВНЕДРЕНИЕ СИСТЕМЫ**

|  |  |
| --- | --- |
| Капитальные вложения на построение АИС «Аэропорт» | 50% |
| Управление и администрирование АИС «Аэропорт» | 12% |
| Техническая поддержка и обновление оборудования и программного обеспечения | 16% |
| Затраты связанные с активностью непосредственно пользователей рабочих мест АИС «Аэропорт» | 22% |

Наиболее ощутимыми выгодами внедрения АИС «Аэропорт» являются:

* сокращение времени на ведение документации;
* защищенность от потери данных;
* возможность быстрого доступа к любой информации, что обеспечивает меньшее количество ошибок при заполнении системы.

Косвенные показатели:

* повышение качества ведения документации;
* сокращение времени получения данных по запросам пользователей;
* сокращение числа ошибок за счет экспертной системы контроля значений.

В таблице 7 показана общая смета затрат на проектирование.

Таблица 7.

**СМЕТА ЗАТРАТ НА ПРОЕКТИРОВАНИЕ**

|  |  |  |
| --- | --- | --- |
| **Статьи** | **Затраты** | |
| **Сумма (руб.)** | **Удельный вес статьи в общей стоимости (%)** |
| Материалы | 167,50 | 3,01 |
| Основная заработная плата | 2260,00 | 48,02 |
| Дополнительная заработная плата | 122,00 | 3,07 |
| Отчисления на единый социальный налог | 1000,68 | 12,26 |
| Затраты на отладку программы | 980,00 | 14,43 |
| Накладные расходы | 1400,00 | 19,21 |
| ИТОГО: | 5930,18 | 100 |

**3. ТРЕБОВАНИЯ К АППАРАТНОМУ И ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ**

**3.1. Требования к составу и параметрам технических средств**

В данном пункте рассматривается минимальный и оптимальный набор технических средств, для нормального функционирования данной программы.

Минимальный набор технических средств для функционирования программного продукта в среде Microsoft Visual Studio C#: персональный компьютер с процессором Intel Celeron; объем свободного места на жестком диске для программного продукта составляет 500 Mb; объем ОЗУ – 192 Mb; монитор SVGA – 15”; клавиатура.

Оптимальный набор технических средств для функционирования программного продукта: персональный компьютер Intel Pentium 3 или Intel Pentium Celeron 2.40 GHz; объем свободного места на жестком диске для программного изделия составляет 700 Mb; объем ОЗУ - 1000 Mb; монитор SVGA – 15”; клавиатура; мышь.

**3.2. Требования к программному обеспечению**

Для правильного функционирования всех модулей программного продукта необходимо наличие следующих программных продуктов и библиотек:

1. операционная система Windows 2000 или Windows XP;

2. Microsoft Visual Studio C#;

3. Microsoft Office Access 2003

4. библиотеки .Net Framework 2.0;

5. SQL Server 2005

6. запускаемый файл Аэропорт.exe, весь проект занимает на жестком диске 22,7 Mb.

**3.3. Требования к документации**

ГОСТ 2.105-79. Общие требования к текстовым документам.

ГОСТ 2.106-68. Текстовые документы.

ГОСТ 7.1-84. ЕСКД. Оформление ссылок.

ГОСТ 19.150-78. Общие требования к программным документам.

ГОСТ 19.104–78. Основные надписи титульного листа и листа утверждения.

ГОСТ 19.106–78. Второй и последующие листы утверждения.

ГОСТ 1.5–68. ЕСПД. Оформление таблиц.

ГОСТ 19.101-77. Виды программ и программных документов.

ГОСТ 19.103-77. Обозначение программ и программных документов.

ГОСТ 19.104–78. Основные надписи.

ГОСТ 19.404–79. Пояснительная записка.

ГОСТ 7.1-84. Оформление ссылок.

ГОСТ 19.201-78. Технического задания.

ГОСТ 19.503-79. Руководство системного программиста.

ГОСТ 19.504-79. Руководства программиста.

ГОСТ 19.505-79. Руководства оператора.

ГОСТ 19.506-79. Описание языка.

ГОСТ 19.507-79. Ведомость эксплуатационных документов.

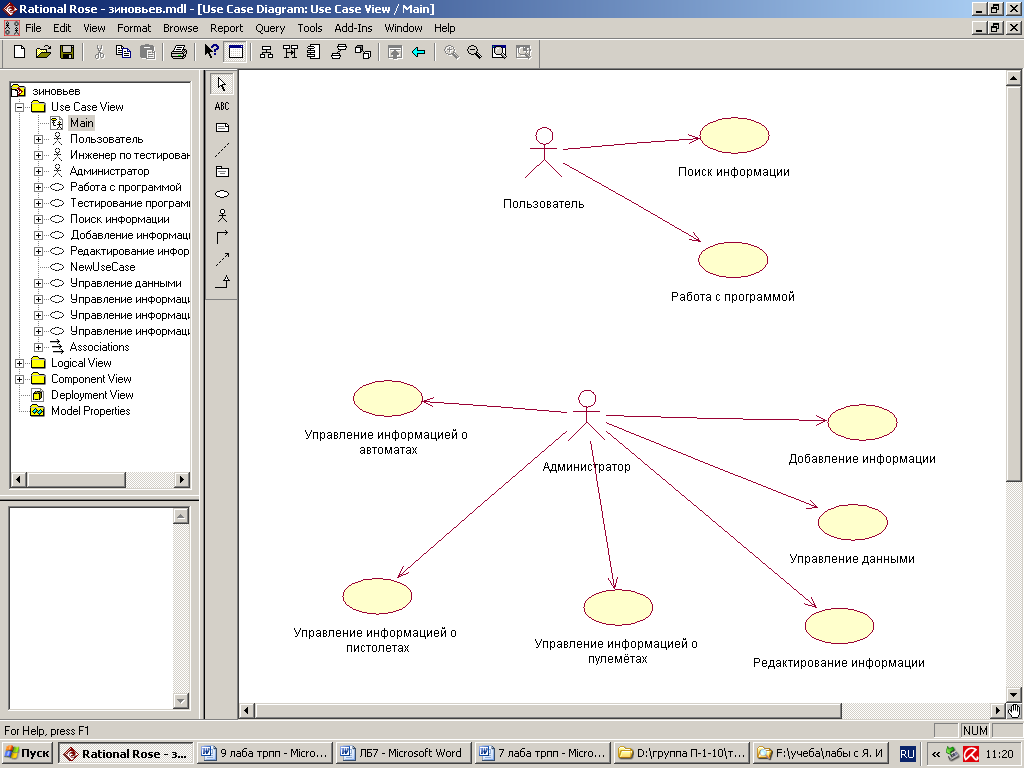
**3.4. Описание структуры и схемы работы программы**

Одним из важнейших этапов является проектирование.

На этапе проектирования были созданы диаграммы: диаграмма вариантов использования, диаграмма классов, диаграмма взаимодействий, диаграмма коопераций.

В диаграмме вариантов использования показаны варианты использования данного программного продукта различными группами пользователей (рис. 18).

**ДИАГРАММА ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ**



Управление информацией

о пассажирах

Управление информацией

о вылетах и рейсах

Управление информацией

о самолетах

Рис. 18

В данной диаграмме выделено две группы пользователей: пользователь и администратор. В таблице 8 указаны актеры, и краткое описание как они могут использовать данный электронный справочник.

Таблица 8.

**ТАБЛИЦА АКТЕРОВ**

|  |  |
| --- | --- |
| **Актер** | **Краткое описание** |
| Пользователь | Человек, использующий ресурсы программы. Он может только просматривать информацию и пользоваться поиском. |
| Администратор или эксперт | Сотрудник, который управляет информацией в программе, редактирует, добавляет информацию и удаляет её. |

В таблице 9 показаны основные функции и их краткое описание.

Таблица 9.

**ТАБЛИЦА ПРЕЦЕДЕНТОВ**

|  |  |
| --- | --- |
| **Прецедент** | **Краткое описание** |
| Редактирование информации | Позволяет редактировать данные в таблицах. |
| Поиск информации | Позволяет производить поиск необходимой информации в программе. |
| Работа с программой | Позволяет знакомиться с информацией, содержащейся в программе. |
| Добавление информации | Позволяет добавлять необходимую информацию в программу. |
| Управление данными | Позволяет управлять данными. |
| Управление информацией о автоматах | Позволяет управлять информацией в таблице «Автоматы» |
| Управление информацией о пистолетах | Позволяет управлять информацией в таблице «Пистолеты» |
| Управление информацией о пулемётах | Позволяет управлять информацией в таблице «Пулемёты» |

На рисунке 19 показана диаграмма классов данной предметной области.

**ДИАГРАММА КЛАССОВ**



Рис. 19

В таблице 10 представлены эти классы со своими полями и методами.

Таблица 10.

**ОПИСАНИЕ КЛАССОВ, ИСПОЛЬЗУЕМЫХ**

**В ПРОГРАММНОМ ПРОДУКТЕ**

|  |  |  |
| --- | --- | --- |
| **Класс** | **Краткое описание** | **Методы класса** |
| *Journal* | Абстрактный класс для ведения лог-файла, в котором будут храниться данные о совершенных операциях в приложении. Методы этого класса дают считать путь к этому файлу, а также добавить в него новую строку. | 1. *public abstract string JouranlPath;* 2. *public abstract void AddRow (LogType row);* |
| *SerializableCore* | Абстрактный класс для сериализации объекта (сохранения его состояния для последующего восстановления). Методы этого класса позволяют считать либо записать путь к сериализованному объекту, а также реализуют непосредственно его сериализацию и десериализацию. | 1. *public abstract string Path;* 2. *public abstract void Serialize (SerializeObj module);* 3. *public abstract SerializeObj Desserialize();* |
| *IQueryControl* | Интерфейсный класс, в котором объявлены методы для работы с базой. Методы данного класса предназначены для подключения к БД, осуществления выборки, добавления, удаления, изменения данных из БД, а также для cохранения информации о произведенных действиях. | 1. *DataTable Execute(string SQL);* 2. *int ExecuteNonQuery(string SQL);* 3. *void ConnectTo(string connection\_string);* 4. *void AddRowLog(LogType row);* 5. *void Disconnect();* |
| *SerializableObject* | Класс, в котором описывается сериализация объектов. Является наследником класса *SerializableCore*. | 1. *public override string Path;* 2. *public override void Serialize (SerializeType module);* 3. *public override SerializeType Desserialize();* |
| *JournalControl* | Данный класс является наследником класса *JournalRow*. Он содержит объект, в котором хранится вся информация об операции, еще не добавленной в лог-файл. Метод этого класса добавляет запись в лог-файл. | 1. *public override string JouranlPath;* 2. *public override void AddRow (JournalRow row);* |
| *JournalRow* | Этот класс хранит информацию о запросе, которая необходима для добавления в лог-файл. Все методы этого класса являются методами-свойствами. | 1. *public int CountOfRow;* 2. *public string Time;* 3. *public string QueryType;* 4. *public string Success;* |
| *QueryControl* | Этот класс является наследником интерфейсного класса *IQueryControl* и, следовательно, переопределяет все его методы. Также в этом классе описан метод, предназначенный для обработки события данного класса. | 1. *public delegate void Failure-SampleEventHandler(object sender, string SQL);* 2. *public event FailureSample-EventHandler FailureSample;* |

Из диаграммы видно, что в приложении используются абстрактные классы Journal, SerializableCore и интерфейсный класс IQueryControl. Абст-рактные методы и классы на диаграмме выделены курсивом. Интерфейсный класс обозначается кругом. Реализация интерфейсного класса показана с по-мощью пунктирной стрелки с треугольным наконечником. Сплошные линии с треугольными стрелками на одном из концов исходят от классов-потомков к классам-родителям. Таким образом показаны отношения обобщения между классами SerializableCore и SerializableObject, Journal и JournalControl. Для каждого класса выделены его атрибуты и методы. Стрелка с пустым наконечником в виде ромба указывает на отношение агрегации между классами JournalControl и QueryControl. Это значит что класс QueryControl в качестве одного из своих полей использует объект класса JournalControl. С помощью пунктирной стрелки между классами JournalRow и JournalControl показана связь по использованию. Это значит что класс JournalRow является типом параметра операции класса JournalControl.

На рисунке 20 показана диаграмма взаимодействий. На данной диаграмме можно увидеть процесс работы с базой данных.

**ДИАГРАММА ВЗАИМОДЕЙСТВИЙ**

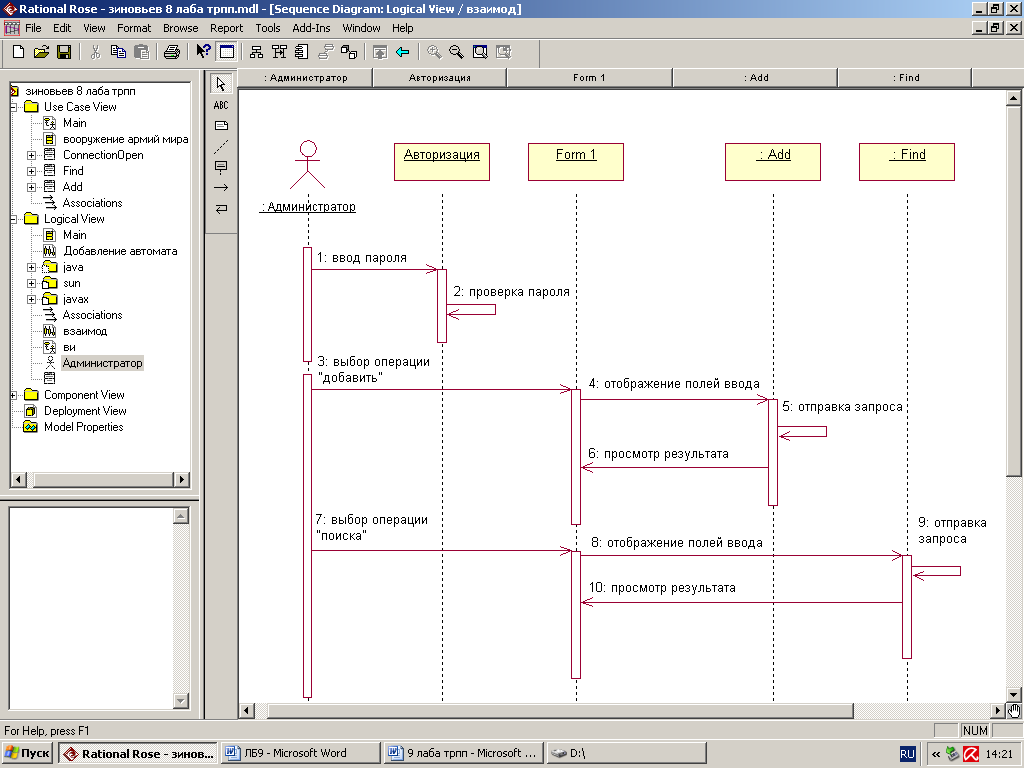


Рис. 20

На диаграмме коопераций показаны потоки данных (рис. 21).

**ДИАГРАММА КООПЕРАЦИЙ**

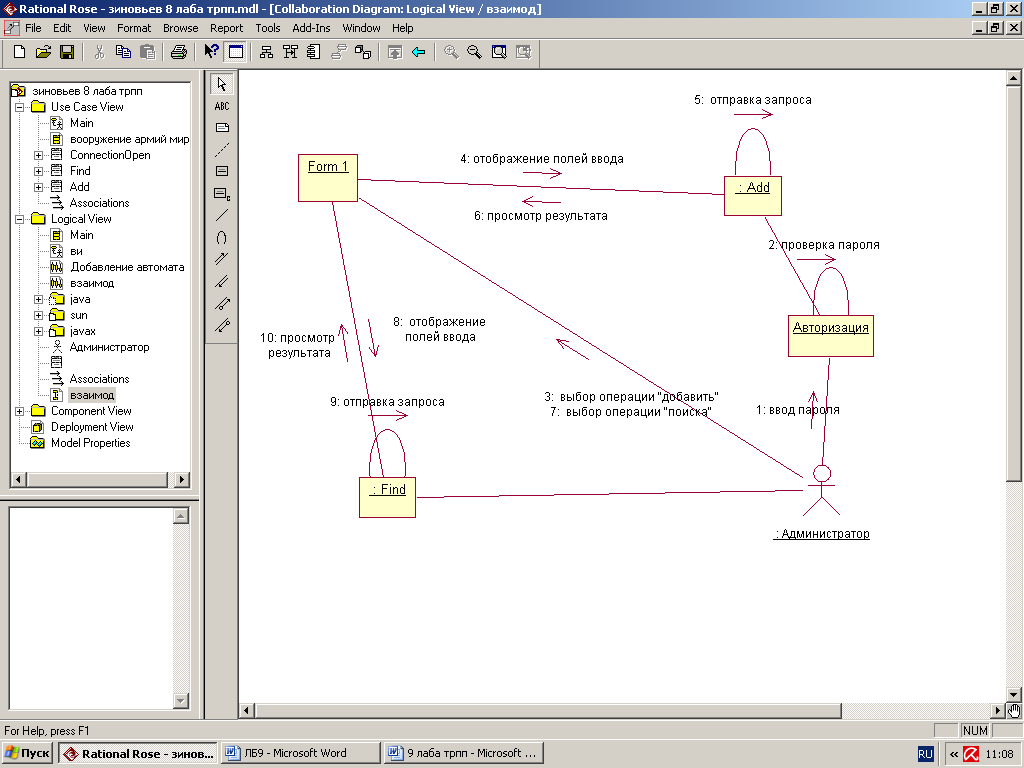


Рис. 21

­­­На рисунке 22 показана структура АИС «Аэропорт».

**СТРУКТУРА АИС «АЭРОПОРТ»**

Form3.cs

(заставка)

Form2.cs

(режим пользователя)

Form1.cs

(разграничение прав доступа)

Form5.cs

(режим эксперта)

Добавление

Поиск

Form4.cs

(авторизация)

Просмотр

Редактирование

Просмотр, поиск

Удаление

Рис. 22

**ЗАКЛЮЧЕНИЕ**

Во время курсового проектирования была изучена среда программирования Visual Studio C#.

Выполнены следующие задачи:

1. Создание в СУБД MS Access базы данных.

2. Создание графического интерфейса пользователя.

3. Проектирование программного продукта.

4. Формирование запросов к БД через среду C# посредством SQL-запросов.

5. Создание Windows-приложения в C#.

6. Разграничение прав доступа.

Была изучена предметная область «Аэропорт» и написана автоматизированная информационная система, позволяющая упростить работу персонала аэропорта.

**СПИСОК ТЕРМИНОВ И УСЛОВНЫХ СОКРАЩЕНИЙ**

SQL – Structured Query Language. Язык структурирования запросов

АИС – автоматизированная информационная система

БД – базы данных

СУБД – системы управления базами данных

ИЛМ – инфологическая модель данных

ДЛМ – даталогическая модель данных

ОЗУ – оперативно-запоминающее устройство

Пер. – перевод

Англ. – английский

Ред. – редактирование

С. – страниц

Эл. – электронный

Г. – год

**СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ**

1. Стоцкий Ю. А., Васильев А. А., Телина И. С. – Microsoft Office 2010. Самоучитель. – СПб.: Питер, 2011. – 432 с.: ил. (эл. вид) ;

2. Эндрю Троелсен. – Язык программирования C# 2005 и платформа .NET 2.0: 3-е издание. Перевод с англ. под ред. А. Г. Сивака – М.: Вильямс. 2007. – 1168 с.: ил. (эл. вид);

3. Жарков В. А. – Компьютерная графика, мультимедиа и игры на Visual C# 2005. М.: Жарков Пресс, 2005. – 812 с. (эл. вид)

4. Кетков Ю. Л., Кетков А. Ю., Шапошников Д. Е.: Школьная Энциклопедия «Персональный компьютер», Москва научное издательство «Большая Российская энциклопедия» издательский дом «Дрофа».

5. Герберт Шилдт. – C#: Учебный курс. – СПб.: Питер; К.: Издательская группа BHV, 2003. – 512 с.: ил. (эл. вид);

6. Герберт Шилдт. – Полный справочник по C#.: Пер. с англ. – М.: Вильямс, 2004. – 752 с.: ил. (эл. вид) ;

7. Астахова И. Ф., Толстобров А. П., Мельников В. М. – SQL в примерах и задачах: Учебное пособие. – Мн.: Новое знание, 2002. – 176 с. (эл. вид).

8. Рудаков А. В.: «Технология разработки программных продуктов». Москва; Издательский центр «Академия»;

9. Рудаков А. В., Федорова А. В.: «Технология разработки программных продуктов». Практикум. Москва; Издательский центр «Академия».

**ПРИЛОЖЕНИЕ 1**

**Листинг программы**

**Form1**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using System.Threading;

namespace ЛБ\_5

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button6\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private void button1\_Click(object sender, EventArgs e)

{

Form2 r2 = new Form2();

r2.Show();

}

private void button2\_Click(object sender, EventArgs e)

{

Form3 r3 = new Form3();

r3.Show();

}

private void button3\_Click(object sender, EventArgs e)

{

Form4 r4 = new Form4();

r4.Show();

}

private void button4\_Click(object sender, EventArgs e)

{

Form5 r5 = new Form5();

r5.Show();

}

private void button5\_Click(object sender, EventArgs e)

{

Form6 r6 = new Form6();

r6.Show();

}

private void Form1\_Load(object sender, EventArgs e)

{

Form7 F = new Form7();//заставка

F.Show();

Thread.Sleep(3000);

F.Close();

}

private void справкаToolStripMenuItem\_Click(object sender, EventArgs e)

{

Form8 r8 = new Form8();

r8.Show();

}

private void статьиToolStripMenuItem\_Click(object sender, EventArgs e)

{

Form9 r8 = new Form9();

r8.Show();

}

}

}

**Form2**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using System.Data.OleDb;

namespace ЛБ\_5

{

public partial class Form2 : Form

{

string InstConnectString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + @"Аэропорт.mdb";

OleDbConnection con;

public Form2()

{

InitializeComponent();

con = new OleDbConnection(InstConnectString);

con.Open();

}

private void Form2\_Load(object sender, EventArgs e)

{

this.рейсыTableAdapter.Fill(this.аэропортDataSet.Рейсы);

}

private void button1\_Click(object sender, EventArgs e)

{

this.Close();

}

private void button1\_Click\_1(object sender, EventArgs e)

{

OleDbCommand insCom = new OleDbCommand("Delete from [Рейсы] where (([Рейсы].[№ рейса])=" + int.Parse(textBox1.Text) + ");", con);

if (insCom.ExecuteNonQuery() == 1)

MessageBox.Show("Сведения изменены", "изменение", MessageBoxButtons.OK);

else

MessageBox.Show("Сведения не изменены", "изменение", MessageBoxButtons.OK);

this.рейсыTableAdapter.Fill(this.аэропортDataSet.Рейсы);

}

private void button2\_Click(object sender, EventArgs e)

{

OleDbCommand InsCom = new OleDbCommand("Insert into [Рейсы] values(" + int.Parse(textBox2.Text) + ",'" + textBox3.Text + "'," + int.Parse(textBox4.Text) + ",'" + textBox5.Text + "'," + textBox10.Text + ")", con);

if (InsCom.ExecuteNonQuery() == 1)

MessageBox.Show("Сведения успешно добавлены", "Добавление", MessageBoxButtons.OK);

this.рейсыTableAdapter.Fill(this.аэропортDataSet.Рейсы);

}

private void button3\_Click(object sender, EventArgs e)

{

OleDbCommand insCom = new OleDbCommand("UPDATE [Рейсы] SET [Рейсы].[Название рейса]='" +

textBox8.Text + "', Рейсы.[№ самолета]=" + int.Parse(textBox7.Text) + ",[Рейсы].[Время вылета]='" +

textBox6.Text + "',[Рейсы].[Стоимость]=" + int.Parse(textBox11.Text) + " WHERE ((([Рейсы].[№ рейса])=" + int.Parse(textBox9.Text) + "))", con);

if (insCom.ExecuteNonQuery() == 1)

MessageBox.Show("Сведения изменены", "изменение", MessageBoxButtons.OK);

else

MessageBox.Show("Сведения не изменены", "изменение", MessageBoxButtons.OK);

this.рейсыTableAdapter.Fill(this.аэропортDataSet.Рейсы);

}

}

}

**Form3**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

namespace ЛБ\_5

{

public partial class Form3 : Form

{

public Form3()

{

InitializeComponent();

}

private void Form3\_Load(object sender, EventArgs e)

{

this.пассажирыTableAdapter.Fill(this.аэропортDataSet.Пассажиры);

}

private void button1\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

**Form4**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using System.Data.OleDb;

namespace ЛБ\_5

{

public partial class Form4 : Form

{

string InstConnectString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + @"Аэропорт.mdb";

OleDbConnection con;

public Form4()

{

InitializeComponent();

con = new OleDbConnection(InstConnectString);

con.Open();

}

private void Form4\_Load(object sender, EventArgs e)

{

this.самолетыTableAdapter.Fill(this.аэропортDataSet.Самолеты);

}

private void button1\_Click(object sender, EventArgs e)

{

this.Close();

}

private void button1\_Click\_1(object sender, EventArgs e)

{

OleDbCommand insCom = new OleDbCommand("Delete from [Самолеты] where (([Самолеты].[№ самолета])=" + int.Parse(textBox1.Text) + ");", con);

if (insCom.ExecuteNonQuery() == 1)

MessageBox.Show("Сведения изменены", "изменение", MessageBoxButtons.OK);

else

MessageBox.Show("Сведения не изменены", "изменение", MessageBoxButtons.OK);

this.самолетыTableAdapter.Fill(this.аэропортDataSet.Самолеты);

}

private void button2\_Click(object sender, EventArgs e)

{

OleDbCommand InsCom = new OleDbCommand("Insert into [Самолеты] values(" + int.Parse(textBox2.Text) + ",'"+ textBox3.Text + "'," + int.Parse(textBox4.Text) + ",'" + textBox5.Text + "')", con);

if (InsCom.ExecuteNonQuery() == 1)

MessageBox.Show("Сведения успешно добавлены", "Добавление",

MessageBoxButtons.OK);

this.самолетыTableAdapter.Fill(this.аэропортDataSet.Самолеты);

}

private void button3\_Click(object sender, EventArgs e)

{

OleDbCommand insCom = new OleDbCommand("UPDATE [Самолеты] SET [Самолеты].[Тип самолета]='" +

textBox8.Text + "', Самолеты.[Кол-во мест]=" + int.Parse(textBox7.Text) + ",[Самолеты].[Название авиакомпании]='"+textBox6.Text +"' WHERE ((([Самолеты].[№ самолета])=" +int.Parse(textBox9.Text)+"))", con);

if (insCom.ExecuteNonQuery() == 1)

MessageBox.Show("Сведения изменены", "изменение", MessageBoxButtons.OK);

else

MessageBox.Show("Сведения не изменены", "изменение", MessageBoxButtons.OK);

this.самолетыTableAdapter.Fill(this.аэропортDataSet.Самолеты);

}

}

}

**Form5**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

namespace ЛБ\_5

{

public partial class Form5 : Form

{

public Form5()

{

InitializeComponent();

}

private void Form5\_Load(object sender, EventArgs e)

{

this.вылетыTableAdapter.Fill(this.аэропортDataSet.Вылеты);

}

private void button1\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

**Form6**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

namespace ЛБ\_5

{

public partial class Form6 : Form

{

public Form6()

{

InitializeComponent();

}

private void Form6\_Load(object sender, EventArgs e)

{

this.дата\_вылета\_\_рейсы\_TableAdapter.Fill(this.аэропортDataSet.\_Дата\_вылета\_\_рейсы\_);

this.инф\_о\_пассажирахTableAdapter.Fill(this.аэропортDataSet.Инф\_о\_пассажирах);

this.самолетыTableAdapter.Fill(this.аэропортDataSet.Самолеты);

this.все\_самолеты\_авиакомпанииTableAdapter.Fill(this.аэропортDataSet.Все\_самолеты\_авиакомпании);

this.самолетыTableAdapter.Fill(this.аэропортDataSet.Самолеты);

}

private void button1\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

**Form9**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

namespace ЛБ\_5

{

public partial class Form9 : Form

{

string str;

public Form9()

{

InitializeComponent();

}

private void linkLabel3\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

str = Application.StartupPath + "\\" + linkLabel3.Text + ".mht";

webBrowser1.Url = new Uri(str, UriKind.Absolute);

webBrowser1.Focus();

}

private void linkLabel1\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

str = Application.StartupPath + "\\" + linkLabel1.Text + ".mht";

webBrowser1.Url = new Uri(str, UriKind.Absolute);

webBrowser1.Focus();

}

private void linkLabel2\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

str = Application.StartupPath + "\\" + linkLabel2.Text + ".mht";

webBrowser1.Url = new Uri(str, UriKind.Absolute);

webBrowser1.Focus();

}

}

}