МИНИСТЕРСТВО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

МЕЖДУНАРОДНЫЙ ИНСТИТУТ КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ И ЭНЕРГЕТИЧЕСКИХ СИСТЕМ

КАФЕДРА информатики и вычислительной техники

КУРСОВОЙ ПРОЕКТ

по дисциплине "БАЗЫ ДАННЫХ"

ТЕМА: Программное приложение «Городской аэропорт»

Расчетно – пояснительная записка

Разработал студент гр. ИВТо-201 /Артемов Е.А. /

подпись,дата инициалы,фамилия

Руководитель /Ганцева Е.А. /

подпись,дата инициалы,фамилия

Нормоконтролер / /

подпись,дата инициалы,фамилия

Защищен\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата

Воронеж 2024

МЕЖДУНАРОДНЫЙ ИНСТИТУТ КОМПЬЮТЕРНЫХ

ТЕХНОЛОГИЙ

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ СИСТЕМ

КАФЕДРА ИНФОРМАТИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

**ЗАДАНИЕ**

на курсовой проект

по дисциплине "Базы данных"

Тема работы: Программное приложение «Городской аэропорт»

Студент группы ИВТо-201 Артемов Евгений Артемович

фамилия, имя, отчество

Технические условия: Среда разработки программного обеспечения «Microsoft Visual Studio, Система управления реляционными базами данных «Microsoft SQL Server».

Содержание и объем проекта (графические работы, расчеты и прочее):

раcчетно - пояснительная записка – 29 страницы формата А4; поясняющий текст, 12 рисунков и 6 таблиц).

Срок защиты курсового проекта: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель / Ганцева Е.А. /

подпись, дата фамилия, инициалы

Задание принял студент гр.ИВТо-201 /Артемов Е.А./

подпись, дата фамилия, инициалы

# **Замечания руководителя**

Содержание

[Замечания руководителя 3](#_Toc158889428)

[Введение 5](#_Toc158889429)

[1 Теоретические сведения по БД 7](#_Toc158889430)

[1.1 Определение База данных 7](#_Toc158889431)

[1.2 Свойства базы данных 7](#_Toc158889432)

[1.3 Типы баз данных 8](#_Toc158889433)

[1.4 Популярные системы управления базами данных. 10](#_Toc158889434)

[2 Реализация проекта 12](#_Toc158889435)

[2.1 Постановка задачи 12](#_Toc158889436)

[2.2 Выбор программных средств 12](#_Toc158889437)

[2.3 Проектирование базы данных 13](#_Toc158889438)

[2.4 Создание Базы данных в СУБД Microsoft SQL Server 16](#_Toc158889439)

[2.5 Создание интерфейса в Microsoft Visual Studio 20](#_Toc158889440)

[Список литературы 33](#_Toc158889441)

# **Введение**

Базы данных – это совокупность структур, предназначенных для хранения больших объемов информации и программных модулей, осуществляющих управление данными, их выборку, сортировку и другие подобные действия [1].

Информация базы данных хранится в одной или нескольких таблицах. Любая таблица с базами состоит из набора однотипных записей, расположенных друг за другом. Они представляют собой строки таблицы, которые можно добавлять, удалять или изменять. Каждая запись является набором именованных полей, или ячеек, которые могут хранить самую разнообразную информацию, начиная от даты рождения и заканчивая подробным описание чего-либо. Однотипные поля разных записей образуют столбец таблицы.

Создав одну таблицу, вы уже получаете полноценную базу данных. Однако в реальной жизни структуры баз данных, а соответственно и способы из создания, намного сложнее.

В последние годы на первый план выдвигается новая отрасль – информационная индустрия, связанная с производством технических средств, методов, технологий для производства новых знаний. Эта индустрия тесно связана с развитием компьютерных технологий.

Возникло большое число избыточной информации, в которой иногда трудно сориентироваться и выбрать нужные сведения. Для решения подобных проблем применяются автоматизированные базы данных. Они стали неотъемлемой частью практически всех компьютерных систем – от отрасли до отдельных предприятий. За последние несколько лет вырос уровень потребительских качеств систем управляемыми базами данных (СУБД): разнообразие поддерживаемых функций, удобный для пользователя интерфейс, сопряжение с программными продуктами, в частности с другими СУБД, 6 возможности для работы в сети и т.д. СУБД позволяет сводить воедино информацию из самых разных источников (электронные таблицы, другие базы данных) и помогает быстро найти необходимую информацию, донести ее до окружающих с помощью отчетов, графиков или таблиц.

В данной курсовой работе разработано приложение для работы с базой данных городского аэропорта.

# **Теоретические сведения по БД**

## **Определение БД**

База данных – это упорядоченный набор структурированной информации или данных, которые обычно хранятся в электронном виде в компьютерной системе. База данных обычно управляется системой управления базами данных (СУБД). Данные вместе с СУБД, а также приложения, которые с ними связаны, называются системой баз данных, или, для краткости, просто базой данных.

Данные в наиболее распространенных типах современных баз данных обычно хранятся в виде строк и столбцов, формирующих таблицу. Этими данными можно легко управлять, изменять, обновлять, контролировать и упорядочивать. В большинстве баз данных для записи и запросов данных используется язык структурированных запросов (SQL).

## **Свойства базы данных**

Удобное использование баз данных основано на их свойствах:

1. Быстродействие

Современные БД проектируются по принципу «получить данные прямо сейчас», чтобы пользователь не ждал отклик на запрос.

2. Простота получения и обновления данных

Какой бы высокой ни была скорость, это бессмысленно, если нужно сделать много сложных операций, чтобы получить, обновить или добавить данные в базу.

3. Независимость структуры

Изменения в любом количестве и качестве информации не должны влиять на структуру базы данных. Также изменения не должны касаться программного обеспечения и средств хранения, например жёсткого диска.

4. Стандартизация

Аналогично свойству независимости структуры: при обновлении программного обеспечения или СУБД (системы управления базами данных) база данных не должна менять свою структуру или свойства.

5. Безопасность данных

Для каждой категории пользователей делают список ограничений и доступов, согласно которым можно взаимодействовать с информацией из БД.

6. Интегрированность

Данные должны быть логически связаны. И эти связи должны прослеживаться по структуре таблиц.

7. Многопользовательский доступ

Удалённо вносить изменения и получать информацию из БД могут сразу несколько человек с разных устройств.

## **Типы баз данных**

Чаще всего базы данных классифицируются в зависимости от того, как в них структурирована информация и как с ней взаимодействовать.

1. Иерархические

Простейшая структура, где записи, как ветви, отходят от «родителя». Информация связана по аналогии с папками на рабочем столе. У каждой записи есть физическая связь только с одной предыдущей, а отношение многих ко многим невозможно.

2. Сетевые

В отличие от иерархической структуры, у каждой записи может быть более одного родителя. Сетевые БД представляют собой не древовидный, а общий граф.

3. Объектно-ориентированные

Базы данных, где информация о реальных вещах представлена в виде объектов под уникальным идентификатором. К ООБД обращаются на языке объектно-ориентированного программирования (ООП). Состояние объекта описывается атрибутами, а его поведение — набором методов. Объекты с одинаковыми атрибутами и методами образуют классы.

Объект в ООП создаётся как отдельная сущность со своими свойствами и методами работы. И как только объект создан, его можно вызвать по «имени», или коду, а не разрабатывать заново.

4. Реляционные

Их также называют SQL – как язык программирования, с помощью которого создают, преобразовывают и управляют данными в реляционных БД. Записи и связи между ними организованы при помощи таблиц. В таблицах есть поле для внешнего ключа со ссылками на другие таблицы. Благодаря высокой организации и гибкости структуры реляционные БД применяются для многих типов данных.

5. Нереляционные (NoSQL)

Эту группу называют также NoSQL, потому что к таким базам данных нужны отличные от SQL-запросы. К ним относятся:

* Базы данных «ключ-значение»

В таких базах данные сохраняются под ключами. Если хотите получить

объект, например, изображение или текст, нужно ввести ключ. Таким образом часто хранят информацию о состоянии объектов, представленную различными типами данных. Каждому хранилищу разрабатывают свою схему именования ключей в зависимости от форматов значений.

* Графовые базы данных

Связи в графовых базах данных обозначены узлами, рёбрами и свойствами. Записи в этих БД могут иметь любое количество связанных с ними свойств.

* Колоночные базы данных

Подобно реляционным, в этих базах данные хранятся в виде таблиц. Но структура колонок строго не регламентирована — они могут объединяться в колоночные семейства с определенным форматом. Строки колоночного семейства имеют уникальные идентификаторы.

## **Популярные системы управления базами данных.**

Системы управления базами данных (СУБД) – это инструменты, с помощью которых пользователь обращается к данным, изменяет их или создаёт. СУБД функционируют как менеджеры по работе с базами данных, которые «говорят» на их языке программирования. В российских и зарубежных компаниях используют шесть популярных СУБД.

1. Oracle

Объектно-реляционная СУБД, созданная одноимённой компанией-лидером на рынке. Преимущества Oracle: быстрая установка и настройка, возможность расширять функционал, практичность и надёжность. Но лицензия стоит дорого, поэтому Oracle обычно используют крупные корпорации.

2. MySQL

Реляционная СУБД с открытым исходным кодом, то есть доступна для просмотра, исправления ошибок и создания новых версий программ. MySQL – бесплатная, быстрая и гибкая система, подходящая для таблиц разных типов.

3. Microsoft SQL Server

Оптимальная СУБД для операционных систем Windows, но совместима и с Linux. Легко интегрируется с другими продуктами Microsoft, удобна в использовании, но потребляет много ресурсов, а лицензия стоит дорого.

4. PostgreSQL

Объектно-реляционная СУБД, которую используют для сайтов, сервисов и платформ. Бесплатный доступ и поддержка многих языков программирования делают эту СУБД одной из самых популярных. По её лицензии создано немало расширенных версий, в том числе для коммерческоо использования.

5. Apache Cassandra

В отличие от вышеназванных, Cassandra – нереляционная СУБД. Она разработана на языке Java и принадлежит фонду Apashe Software Foundation. Система хранит данные по модели семейства столбцов и «ключ-значение», распределяет данные в несколько дата-центров и легко масштабируется при увеличении объёма информации.

6. Redis

NoSQL резидентная СУБД, которая использует модель «ключ-значение». Она написана на языках С и С++, а применяется для атомарных операций, например, записи и чтения быстро изменяющихся данных.

# **Реализация проекта**

## **Постановка задачи**

Данная система разработана для компании городского аэропорта, или личного пользования как тренировка перед работой с настоящей базой данных компании. В ней будут содержаться данные о пассажирах, самолетах, авиакомпаниях, рейсах,. Программа должна позволить пользователю удобно выбрать рейс для своего перелета. Для внесения новых данных должно быть реализовано отдельное окно, в котором можно будет занести данные в таблицу. Просмотр таблиц будет в одном окне, но в разных вкладках. Интерфейс должен быть простой и понятный для пользователя.

Необходимо разработать структуру базовых таблиц базы данных, удовлетворяющих требованиям целостности. В таблицах в соответствии с типом данных, размещенных в каждом поле, определить наиболее подходящий тип для каждого поля.

Нужно создать структуры базовых таблиц, установить межтабличные связи между ними (схема данных) и наполнить их содержимым. При создании структуры таблиц целесообразно задать ключевые (уникальные) поля. Это поможет в дальнейшем для организации связей между таблицами.

Требуется несколько триггеров, для заполнения отдельных таблиц данными, связанными с структурой базы данных и вывода нужной для пользователя информации.

## **Выбор программных средств**

Программные средства представляют собой сложный комплекс, обеспечивающий взаимодействие всех частей ИС (Информационная система) при ее функционировании.

Основу программных средств представляет СУБД. В ней можно выделить ядро СУБД, обеспечивающее организацию ввода, обработки и хранения данных, а также другие компоненты, обеспечивающие настройку системы, средства тестирования, утилиты, обеспечивающие выполнение вспомогательных функций таких как восстановление баз данных, сбор статистики о функционировании и др. Важной компонентной СУБД является трансляторы или компиляторы для используемых его языковых средства.

Подавляющее большинство СУБД работает в среде универсальных операционных систем и взаимодействует с ОС при обработке обращений. Поэтому можно считать, что ОС также входит в состав. Для обработки запросов к БД пишутся соответствующие программы, которые представляют прикладное программное обеспечение.

В качестве программ для реализации были выбраны Microsoft SQL Server и Microsoft Visual Studio. Microsoft SQL Server был выбран с учетом того, что этот программный продукт имеет хорошую совместимость с продукцией компании Microsoft. Он нам нужен для реализации нашей базы данных, а Microsoft Visual Studio для создания интерфейса для пользования это самой базой данных.

## **Проектирование базы данных**

Необходимо спроектировать базу данных «Городской аэропорт». Для создания базы данных необходимо проанализировать информацию, создать на ее основе необходимые таблицы, установить определенные связи между ними, а также создать триггеры. Поэтому задачи, которые нужно будет решить в процессе анализа, заключаются в разработке связей между таблицами базы данных, устранении избыточности в таблицах, их обработке и редактирования.

Таблица 1 – Авиакомпании (airlines)

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание поля |
| airline\_id | int | уникальный идентификатор авиакомпании |
| airline\_name | nvarchar(255) | название авиакомпании |
| airline\_address | nvarchar(255) | адрес авиакомпании |
| airline\_phone | nchar(11) | контактный телефон авиакомпании |

Таблица 1 содержит информацию о авиакомпаниях, их названиях, адресах и контактных телефонах.

Таблица 2 – Самолеты (аirplanes)

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание поля |
| airplane\_id | int | уникальный идентификатор самолета |
| airplane\_name | nvarchar(255) | название самолета |
| airplane\_capacity | int | вместимость самолета |
| airline\_id | int | идентификатор авиакомпании, которой принадлежит самолет |

Таблица 2 хранит данные о самолетах, включая их названия, вместимость и связь с соответствующей авиакомпанией.

Таблица 3 – Полеты (Flights)

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание поля |
| flight\_id | int | уникальный идентификатор рейса |
| flight\_date | datetime | дата и время вылета |
| flight\_duration | time | продолжительность полёта |
| flight\_origin | nvarchar(255) | место отправления |
| flight\_destination | nvarchar(255) | место назначения |
| flight\_price | float | цена билета на рейс |
| airplane\_id | int | идентификатор самолета, используемого для рейса |

Таблица 3 включает в себя информацию о рейсах, включая дату и время вылета, продолжительность полета, места отправления и прибытия, цену билета и связь с соответствующим самолетом.

Таблица 4 – Пассажиры (passengers)

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание поля |
| passenger\_id | int | уникальный идентификатор пассажира |
| passenger\_firstname | nvarchar(255) | имя пассажира |
| passenger\_lastname | nvarchar(255) | фамилия пассажира |
| passenger\_address | nvarchar(255) | адрес пассажира |
| passenger\_phone | nvarchar(20) | контактный телефон пассажира |

Таблице 4 содержит данные о пассажирах, включая имена, фамилии, адреса и контактные телефоны.

Таблица 5 – Места (seats)

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание поля |
| seat\_id | int | уникальный идентификатор места |
| seat\_number | nvarchar(20) | класс места (например, эконом, бизнес) |
| seat\_class | nvarchar(20) | Дата занесения информации |
| airplane\_id | int | идентификатор самолета, к которому относится место |

Таблица 5 содержит информацию о местах на самолете, включая номер места, класс и связь с соответствующим самолетом.

Таблица 6 – Бронь (reservations)

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание поля |
| reservation\_id | int | уникальный идентификатор бронирования |
| reservation\_date | datetime | дата бронирования |
| flight\_id | int | идентификатор рейса, на который произведено бронирование |
| passenger\_id | int | идентификатор пассажира, совершившего бронирование |
| seat\_id | int | идентификатор забронированного места |

Таблица 6 хранит информацию о бронированиях, включая дату бронирования, связь с рейсом, пассажиром и забронированным местом.

## **Создание Базы данных в СУБД Microsoft SQL Server**

Для проектирования базы данных будем использовать СУБД Microsoft SQL Server. Для начала нужно создать базу данных через запрос, используя команду: CREATE DATABASE CityAirport.

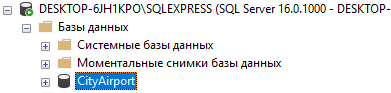


Рисунок 1 – Визуальное обозначение созданной БД

После появления базы данных в обозревателе объектов, можно приступить к созданию таблиц.

Последующим шагом вводим команды для создания каждой отдельной таблицы и в ней прописываем названия каждого столбца, который в ней должен быть, а также какие данные, какого формата и какой длинны могут в ней содержаться. Одному столбцу в таблице надо указать первичный ключ, а также некоторым другим внешний. Первичный ключ (primary key) – особенное поле в SQL-таблице, которое позволяет однозначно индентифицировать каждую запись в ней. Внешний ключ (foreign key) нужен для того, чтобы связать две разные SQL-таблицы между собой. Внешний ключ таблицы должен соответствовать значению первичного ключа таблицы, с которой он связан. Это помогает сохранять согласованность базы данных путем обеспечения так называемой «ссылочной целостности». Каждый столбец имеет свой тип данных для понимания каким форматов строка в столбце будет заполнятся, а также какие у нее ограничения.

Таблица 9 – Используемые типы данных

|  |  |
| --- | --- |
| Тип данных | Используемые данные |
| int | Используются для хранения чисел от –2 147 483 648 до 2 147 483 647 |
| nvarchar | Слова размером, не больше указанного в скобках (60,20,100 и т.д.) |
| nchar | Символы размером. Не больше указанного в скобках (до 4 000) |
| datetime | Дата и время от Jan 1, 1753 до Dec 31, 9999 |
| float | Используемые для числовых данных с плавающей запятой и хранит числа от –1.79E+308 до 1.79E+308 |
| date | Дата в формате ГГГГ-ММ-ДД (2001-05-07) |

Дальше записываем запрос создания таблиц, он должен получится вида:

CREATE TABLE Airlines (

airline\_id INT IDENTITY PRIMARY KEY,

airline\_name VARCHAR(255),

airline\_address VARCHAR(255),

airline\_phone VARCHAR(20)

);

CREATE TABLE Airplanes (

airplane\_id INT IDENTITY PRIMARY KEY,

airplane\_name VARCHAR(255),

airplane\_capacity INT,

airline\_id INT,

FOREIGN KEY (airline\_id) REFERENCES Airlines (airline\_id)

);

CREATE TABLE Flights (

flight\_id INT IDENTITY PRIMARY KEY,

flight\_date DATETIME,

flight\_duration TIME,

flight\_origin VARCHAR(255),

flight\_destination VARCHAR(255),

flight\_price FLOAT,

airplane\_id INT,

FOREIGN KEY (airplane\_id) REFERENCES Airplanes (airplane\_id)

);

CREATE TABLE Passengers (

passenger\_id INT IDENTITY PRIMARY KEY,

passenger\_firstname VARCHAR(255),

passenger\_lastname VARCHAR(255),

passenger\_address VARCHAR(255),

passenger\_phone VARCHAR(20)

);

CREATE TABLE Seats (

seat\_id INT IDENTITY PRIMARY KEY,

seat\_number VARCHAR(20),

seat\_class VARCHAR(20),

airplane\_id INT,

FOREIGN KEY (airplane\_id) REFERENCES Airplanes (airplane\_id)

);

CREATE TABLE Reservations (

reservation\_id INT IDENTITY PRIMARY KEY,

reservation\_date DATETIME,

flight\_id INT,

passenger\_id INT,

seat\_id INT,

FOREIGN KEY (flight\_id) REFERENCES Flights (flight\_id),

FOREIGN KEY (passenger\_id) REFERENCES Passengers (passenger\_id),

FOREIGN KEY (seat\_id) REFERENCES Seats (seat\_id)

);

create table register(

id\_user int identity primary key,

login\_user varchar(50) not null,

password\_user varchar(50) not null,

);

После выполнения запроса у нас появится диаграмма базы данных, в которой можно посмотреть все связи созданных нами таблиц. Она имеет вид:

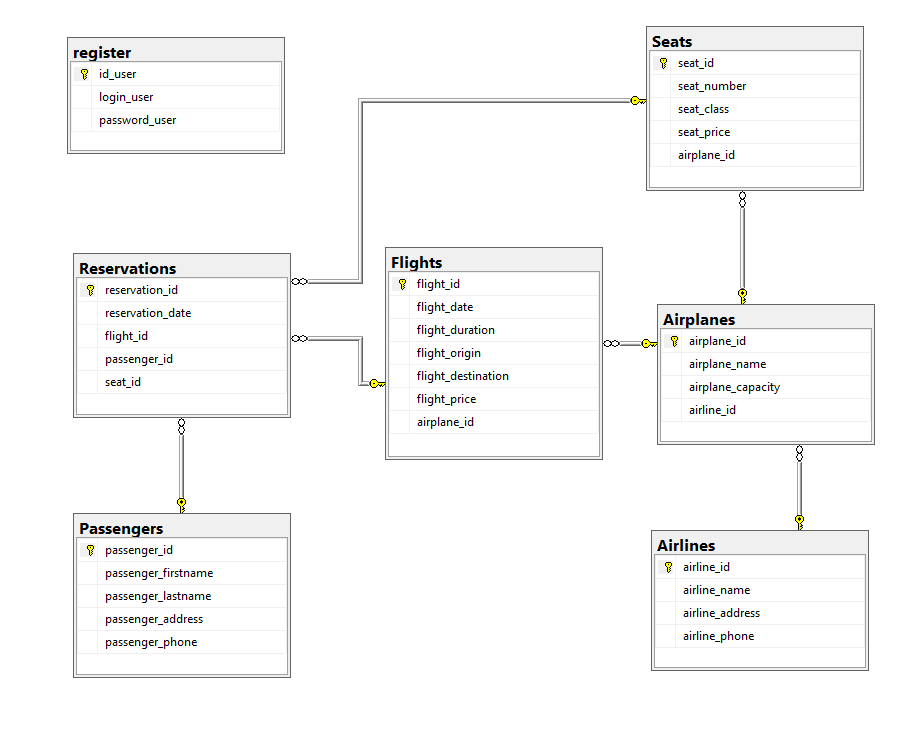


Рисунок 2 – Диаграмма базы данных

Ключ на соединительной линии обозначает, что в таблице, на которую тот указывает есть первичный ключ, связанных с другой таблицей в котором присутствует внешний ключ с таким же названием, что и у первичной.

Далее необходимо создать триггер. Создадим его через SQL запрос вот таким способом:

CREATE TRIGGER ReduceAvailableSeats

ON Reservations

AFTER INSERT

AS

BEGIN

DECLARE @flight\_id INT;

DECLARE @airplane\_id INT;

-- Получаем идентификатор рейса и места из вставленной брони

SELECT @flight\_id = flight\_id, @airplane\_id = airplane\_id FROM inserted;

-- Уменьшаем количество доступных мест на один для выбранного рейса

UPDATE Airplanes

SET airplane\_capacity = airplane\_capacity - 1

WHERE airplane\_id = @airplane\_id;

END;

При добавлении записи с в таблицу брони, будет срабатывать триггер и уменьшать количество доступных мест на один для выбранного рейса.

## **Создание интерфейса в Microsoft Visual Studio**

Для создания интерфейса была выбрана платформа Visual Studio, язык программирования C#. В созданном проекте добавляем несколько форм WindowsForms, для вывода таблиц и добавлений записей. Далее настраиваем связь с базой данных: для этого в обозревателе решений добавляем новую базу данных, спроектированную заранее в Microsoft SQL Server. В обозревателе решений выбираем источником данных эту базу данных, а после указываем имя сервера и проверяем подключение к базе. После удачного подключения на компоненте основной формы было размещено несколько компонентов DataGridView. Эти компоненты необходимы для размещения таблиц из базы данных. Настройка свойства DataSource позволяет отобразить таблицу из базы данных на компоненте DataGridView. Так же добавлены кнопки «Обновить», «Изменить», «Добавить», «Сохранить», «Удалить», «Войти», «Найти», «Регистрация», «Зарегистрироваться».

После всех выполненных действий получим такой результат:

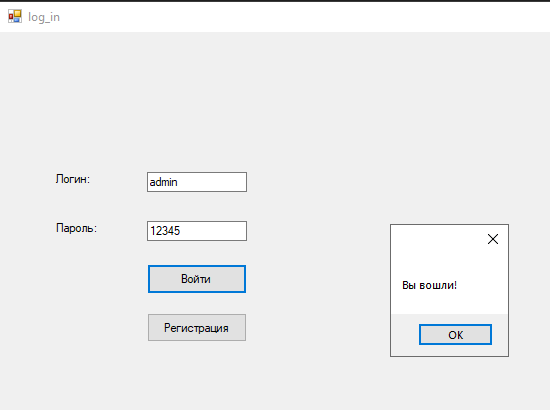


Рисунок 3 – Форма авторизации

На рисунке 3 при входе в приложение ожидается авторизация пользователя, необходимо ввести данные в текстовые поля и выполнить вход нажав на кнопку «Войти» для дальнейшего пользования программой.

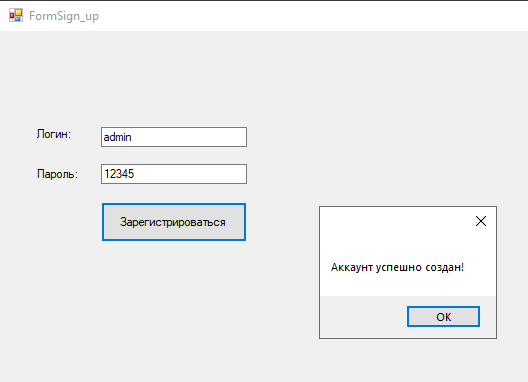


Рисунок 4 – Форма регистрации

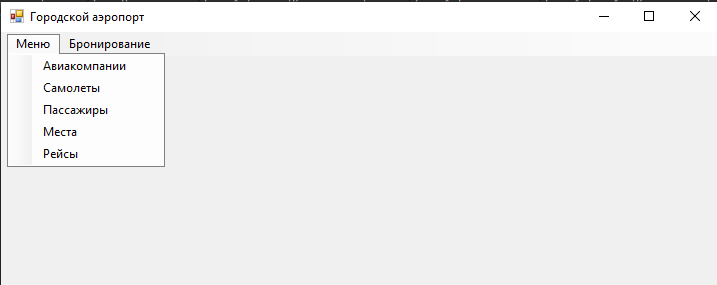
На рисунке 4 представлено окно регистрации, необходимое для регистрации пользователей без учетной записи. Для регистрации пользователь должен ввести данные в тектсовые поля и нажать на кнопку «Зарегистрироваться».

Рисунок 5 – Главная форма

На рисунке 5 после успешной авторизации нас приветствует главное окно программы, в котором содержится меня всех таблиц и раздел бронирования.

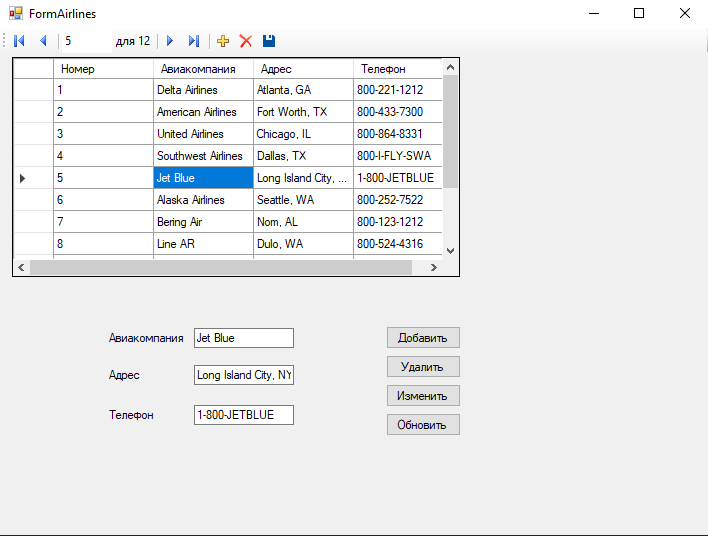


Рисунок 6 – Таблица Авиакомпаний

На рисунке 6 представлена таблица авиакомпаний и кнопки для ее редактирования.

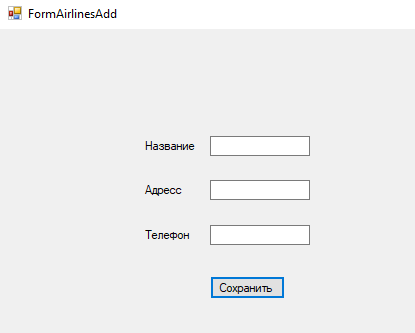


Рисунок 7 – Форма добавления авиакомпаний

На рисунке 7 результат действия нажатия кнопки «Добавить». Данная форма позволяет внести данные для добавления их в таблицу путем ввода в текстовые поля и нажатия кнопки «Сохранить».

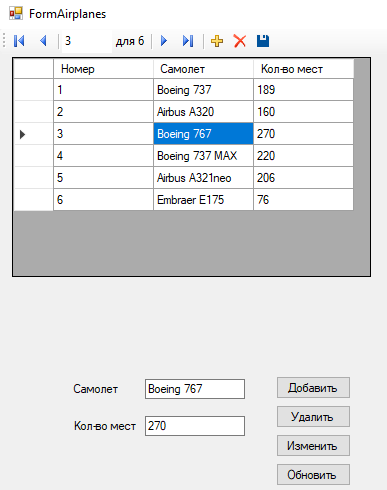


Рисунок 8 – Таблица самолетов

На рисунке 8 представлена таблица пассажиров и кнопки для ее редактирования.

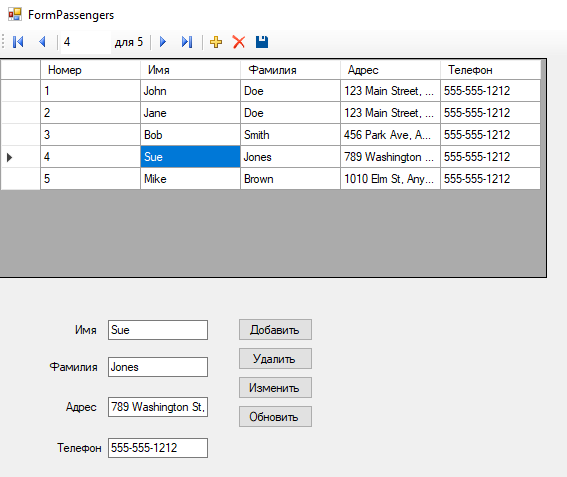


Рисунок 9 – Таблица пассажиров

На рисунке 9 представлена таблица пассажиров и кнопки для ее редактирования.

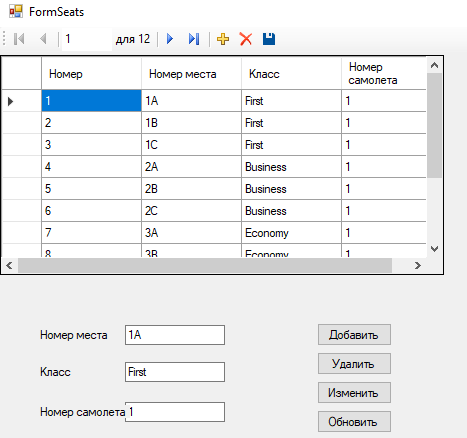


Рисунок 10 – Таблица мест

На рисунке 10 представлена таблица мест и кнопки для ее редактирования.

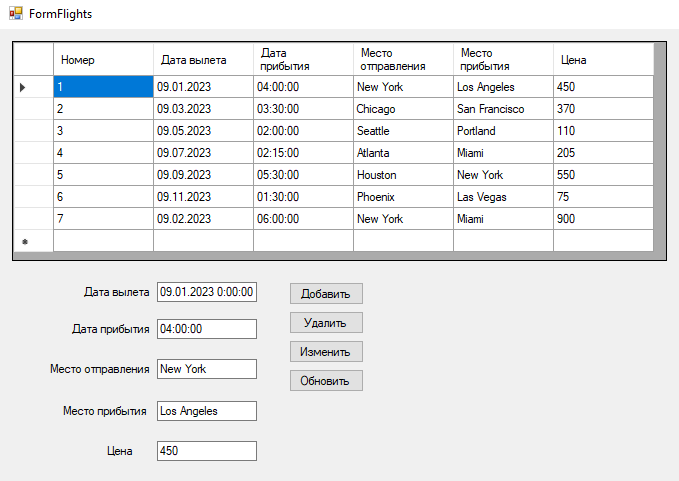


Рисунок 11 – Таблица рейсов

На рисунке 11 представлена таблица рейсов и кнопки для ее редактирования.

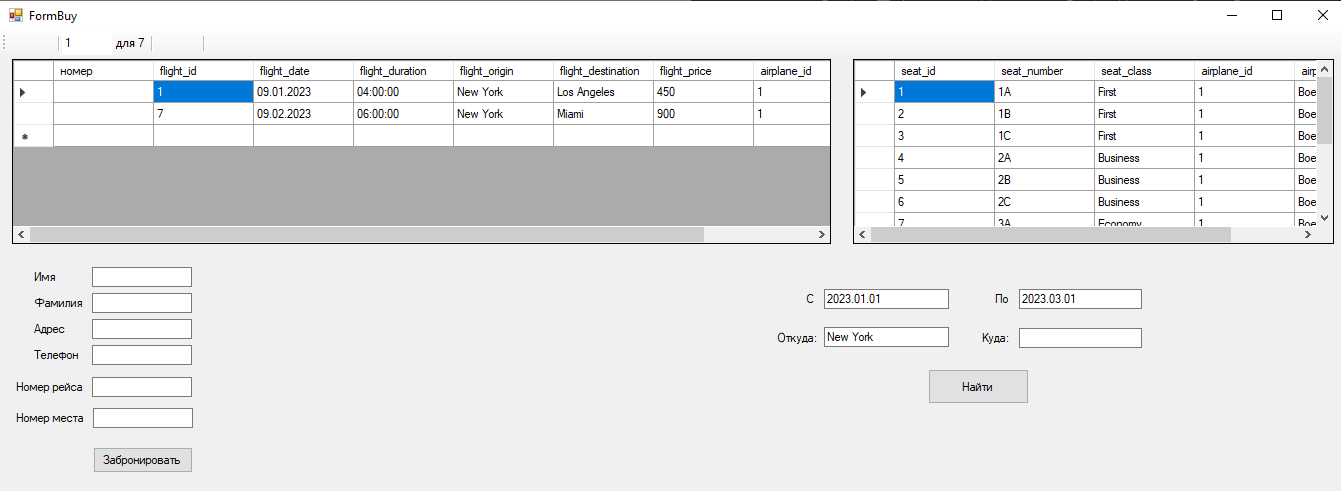


Рисунок 12 – Форма бронирования

На рисунке 12 представлена форма бронирования, позволяющая удобно выбрать нужный рейс с помощью фильтров, а также произвести бронирование.

На кнопку «Найти» запишем действие фильтрации:

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

{

//2023.09.01

DateTime dateStart = DateTime.Parse(textBox1.Text);

DateTime dateEnd = DateTime.Parse(textBox2.Text);

if (textBox3.Text != String.Empty && textBox4.Text != String.Empty)

{

(dataGridView1.DataSource as DataTable).DefaultView.RowFilter = String.Format("flight\_origin LIKE '{0}' AND flight\_destination LIKE '{1}' AND flight\_date >= #{2}# AND flight\_date <= #{3}#",

textBox3.Text, textBox4.Text, dateStart.ToString("yyyy-MM-dd"), dateEnd.ToString("yyyy-MM-dd"));

}

else if (textBox3.Text == "" && textBox4.Text != String.Empty)

{

(dataGridView1.DataSource as DataTable).DefaultView.RowFilter = String.Format("flight\_destination LIKE '{0}' AND flight\_date >= #{1}# AND flight\_date <= #{2}#",

textBox4.Text, dateStart.ToString("yyyy-MM-dd"), dateEnd.ToString("yyyy-MM-dd"));

}

else if (textBox3.Text != String.Empty && textBox4.Text == "")

{

(dataGridView1.DataSource as DataTable).DefaultView.RowFilter = String.Format("flight\_origin LIKE '{0}' AND flight\_date >= #{1}# AND flight\_date <= #{2}#",

textBox3.Text, dateStart.ToString("yyyy-MM-dd"), dateEnd.ToString("yyyy-MM-dd"));

}

else

{

(dataGridView1.DataSource as DataTable).DefaultView.RowFilter = "";

}

}

На кнопку «Добавить» запишем действие открытия новой формы для добавления:

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

{

FormAirlinesAdd addform = new FormAirlinesAdd();

addform.Show();

}

На кнопку «Удалить» запишем действие удаления данных с таблицы:

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

{

string delete = $"delete from airlines where airline\_id = {Convert.ToInt32(airlinesDataGridView.Rows[selectedRow].Cells[0].Value)}";

dataBase.openConnection();

SqlCommand cmd = new SqlCommand(delete, dataBase.GetConnection());

cmd.ExecuteNonQuery();

dataBase.closeConnection();

}

На кнопку «Изменить» запишем действия, которые обновляют данные в таблице Airlines базы данных. Он использует значения из текстовых полей textBox\_name, textBox\_address, textBox\_phone для обновления соответствующих столбцов airline\_name, airline\_address, airline\_phone. Обновление происходит для выбранной строки в airlinesDataGridView.

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

{

string edit = $"update airlines set airline\_name = '{textBox\_name.Text}', airline\_address = '{textBox\_address.Text}', airline\_phone = '{textBox\_phone.Text}' where " +

$"airline\_id = {Convert.ToInt32(airlinesDataGridView.Rows[selectedRow].Cells[0].Value)}";

dataBase.openConnection();

SqlCommand cmd = new SqlCommand(edit, dataBase.GetConnection());

cmd.ExecuteNonQuery();

dataBase.closeConnection();

}

На кнопку «Обновить» обновим адаптер для обновления данных в табилце:

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

{

airlinesTableAdapter.Fill(cityAirportDataSet.Airlines);

}

На кнопку «Сохранить» происходит добавление новой записи об авиакомпании в таблицу Airlines базы данных. При нажатии кнопки "Save" метод получает значения из текстовых полей textBox1 (название авиакомпании), textBox2 (адрес) и textBox3 (телефон), формирует SQL-запрос для добавления новой записи с этими данными в таблицу Airlines, выполняет запрос на добавление записи через метод ExecuteNonQuery() объекта SqlCommand, и закрывает соединение с базой данных.

На кнопку «Войти» метод проверяет наличие пользователя в базе данных по введенному логину и паролю. Затем выводит соответствующее сообщение и открывает новую форму в случае успеха.

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

{

var loginUser = textBox1.Text;

var PasswordUser = textBox2.Text;

SqlDataAdapter adapter = new SqlDataAdapter();

DataTable table = new DataTable();

string querystring = $"select id\_user, login\_user, password\_user from register where login\_user = '{loginUser}' and password\_user = '{PasswordUser}'";

SqlCommand command = new SqlCommand(querystring, database.GetConnection());

adapter.SelectCommand = command;

adapter.Fill(table);

if (table.Rows.Count > 0)

{

MessageBox.Show("Вы вошли!");

this.Hide();

Form1 form = new Form1();

form.ShowDialog();

this.Show();

}

else

{

MessageBox.Show("Такого аккаунта не существует");

}

}

На кнопку «Регистрация» выполняется метод button1\_Click, который отвечает за создание нового аккаунта, где введенные пользователем логин и пароль добавляются в таблицу register. После выполнения операции вставки, отображается соответствующее сообщение, и открывается форма входа. А также выполняется Метод checkuser() выполняет проверку наличия пользователя в базе данных по введенному логину и паролю. Если такой пользователь уже существует, выводится соответствующее сообщение.

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

{

var loginUser = textBox1.Text;

var PasswordUser = textBox2.Text;

string querystring = $"insert into register(login\_user, password\_user) values('{loginUser}', '{PasswordUser}')";

SqlCommand command = new SqlCommand(querystring, dataBase.GetConnection());

dataBase.openConnection();

if(command.ExecuteNonQuery() == 1)

{

MessageBox.Show("Аккаунт успешно создан!");

FormLog\_in formLog\_In = new FormLog\_in();

this.Hide();

formLog\_In.ShowDialog();

}

else

{

MessageBox.Show("Аккаунт не создан");

}

dataBase.closeConnection();

}

private Boolean checkuser()

{

var loginUser = textBox1.Text;

var PasswordUser = textBox2.Text;

SqlDataAdapter adapter = new SqlDataAdapter();

DataTable table = new DataTable();

string querystring = $"select id\_user, login\_user, password\_user from register where login\_user = '{loginUser}' and password\_user = '{PasswordUser}'";

SqlCommand command = new SqlCommand(querystring, dataBase.GetConnection());

adapter.SelectCommand = command;

adapter.Fill(table);

if (table.Rows.Count > 0)

{

MessageBox.Show("Пользователь создан!");

return true;

}

else

{

return false;

}

}

**Заключение**

Таким образом, разработанная база данных предназначена для эффективного управления информацией об авиакомпании. Она включает в себя таблицы для хранения данных об авиакомпаниях, самолетах, рейсах, пуассажирах и бронировании мест. Эта база данных может быть использована в приложении для управления рейсами, регистрации пассажиров, бронирования мест и для генерации отчетности о проданных билетах. Также данные из этой базы могут быть использованы для анализа и улучшения управления рейсами и обслуживания пассажиров.

# Список литературы

1. Голицына, О. Л. Базы данных / О.Л. Голицына, Н.В. Максимов, И.И. Попов. - М.: Форум, 2015. - 400 c.

2. Илюшечкин, В. М. Основы использования и проектирования баз данных. Учебник / В.М. Илюшечкин. - М.: Юрайт, 2015. - 214 c.

3. Костин, А. Е. Организация и обработка структур данных в вычислительных системах. Учебное пособие / А.Е. Костин, В.Ф. Шаньгин. - М.: Высшая школа, 2014. - 248 c.