

## **Реферат**

Курсовой проект представляет собой приложение, предоставляющее функционал поиска и бронирования туров в соответствии с заданными параметрами.

Реферат содержит 32 страницы, 16 иллюстраций, 1 таблицу, 2 приложения, 5 источников.

Ключевые слова: тур, сущность, C#, PostgreSQL, ORM.

## Оглавление

Введение .....	4
1. Аналитический раздел .....	5
1.1. Постановка задачи .....	5
1.2. Пользователи системы.....	5
1.3. Анализ существующих решений .....	6
1.3.1. Coral Travel.....	7
1.3.2. Библио-Глобус .....	9
1.4. Анализ моделей данных .....	10
1.4.1. Иерархическая модель.....	10
1.4.2. Сетевая модель.....	11
1.4.3. Графовая модель .....	11
1.4.4. Реляционная модель .....	11
1.5. Выводы.....	12
2. Конструкторский раздел.....	13
2.1. Сценарии пользователей .....	13
2.1.1. Гость .....	13
2.1.2. Турист.....	13
2.1.3. Менеджер .....	14
2.2. Формализация сущностей системы .....	15
2.3. Ролевая модель.....	16
2.4. Функции .....	17
2.5. Вывод.....	17
3. Технологический раздел.....	18
3.1. Анализ СУБД .....	18

3.1.1. Oracle .....	18
3.1.2. MySQL.....	18
3.1.3. PostgreSQL.....	19
3.2. Технологический стек .....	19
3.3. Реализация функции базы данных.....	19
3.4. Диаграмма базы данных.....	20
3.5. Презентационный уровень приложения.....	21
3.6. Вывод.....	25
Заключение .....	26
Литература .....	27
Приложение А .....	28
Приложение Б.....	32

## **Введение**

На сегодняшний день потребность человека в потреблении культурных, образовательных, развлекательных услуг постоянно возрастает, что влечет за собой развитие соответствующих сфер сервисной деятельности. Одной из таких сфер является туризм.

Туристическое обслуживание представляет собой достаточно сложный вид оказания услуг. Особенность обусловлена тем, что туристическая путевка включает в себя три основных компонента: услуги размещения, питания и трансфер. Покупка тура – это фактически покупка представлений туриста о путешествии.

Целью данной работы является разработка простого и в тоже время функционального сервиса поиска туров. Для достижения поставленной цели необходимо решить следующие задачи:

- разработать концептуальную модель;
- спроектировать базу данных;
- спроектировать архитектуру приложений;
- разработать приложение.

## **1. Аналитический раздел**

В данном разделе будут рассмотрены и проанализированы требования, выдвигаемые к сервису, а также приведены подходы к решению поставленных задач.

### **1.1. Постановка задачи**

Необходимо разработать десктоп-приложение подбора туров по заданным параметрам, функционал которого позволяет выполнять:

- авторизацию пользователя;
- поиск и сортировку туров/отелей/питания/трансфера в соответствии с заданными параметрами;
- бронирование и отмену бронирования туров;
- добавление/изменение/удаление туров, отелей, питания.

### **1.2. Пользователи системы**

В системе были выделены следующие пользователи:

- неавторизированный пользователь;
- авторизированный пользователь;
- менеджер.

Неавторизированному пользователю доступен поиск тура/отеля/питания/трансфера с указанием желаемых параметров. Авторизация для гостя не требуется.

Авторизированный пользователь может также осуществлять подбор тура. Дополнительно, пользователь должен иметь возможность бронирования и отмены бронирования указанного тура, а также просмотра забронированных тура/отеля/питания/трансфера.

Роль менеджера необходима для управления существующими турами. Помимо просмотра и выбора существующих туров, менеджер может добавлять, изменять и удалять тур/отель/питание/трансфер.

### 1.3. Анализ существующих решений

В таблице 1 представлен список крупнейших на данный момент организаторов туров.

Таблица 1 – Список турфирм с наибольшим количеством направлений для путешествий.

Название турфирмы	Дата основания	Количество туристических направлений
ICS Travel Group	1992	60
Good Time	2013	56
Coral Travel	1995	54
Art Tour	1994	27
NTK-Intourist	1929	23
AnexTour	1996	23
Pegas Touristik	1975	22

Как видно из таблицы, на туристическом рынке высокая конкуренция как опытных организаторов (NTK-Intourist), так и новичков (Good Time).

### 1.3.1. Coral Travel

Одним из крупнейших и самых популярных тур-организаторов является компания Coral Travel. На рисунке 1 представлен интерфейс сайта компании.

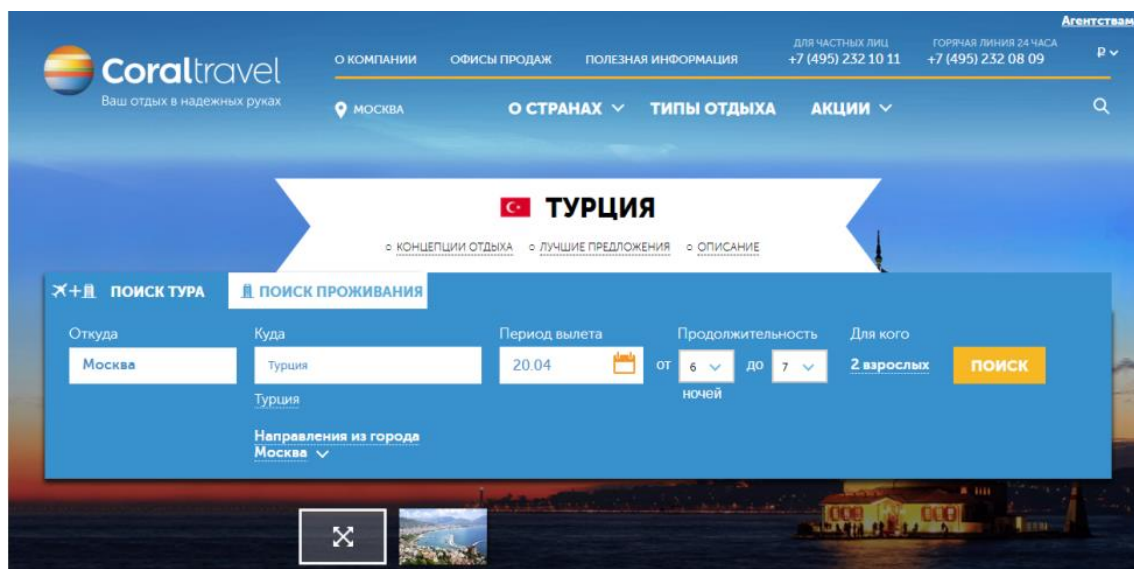


Рисунок 1. Интерфейс сервиса Coral Travel.

Пользователю доступны поиск и покупка тура, авиабилета; бронирование отеля. Также осуществляется подбор оптимальных дат поездки, в зависимости от указанного количества человек и продолжительности.

После того, как пользователь задал страну назначения, даты и количество взрослых, сервис предлагает соответствующие туры. Интерфейс подбора туров представлен на рисунке 2.

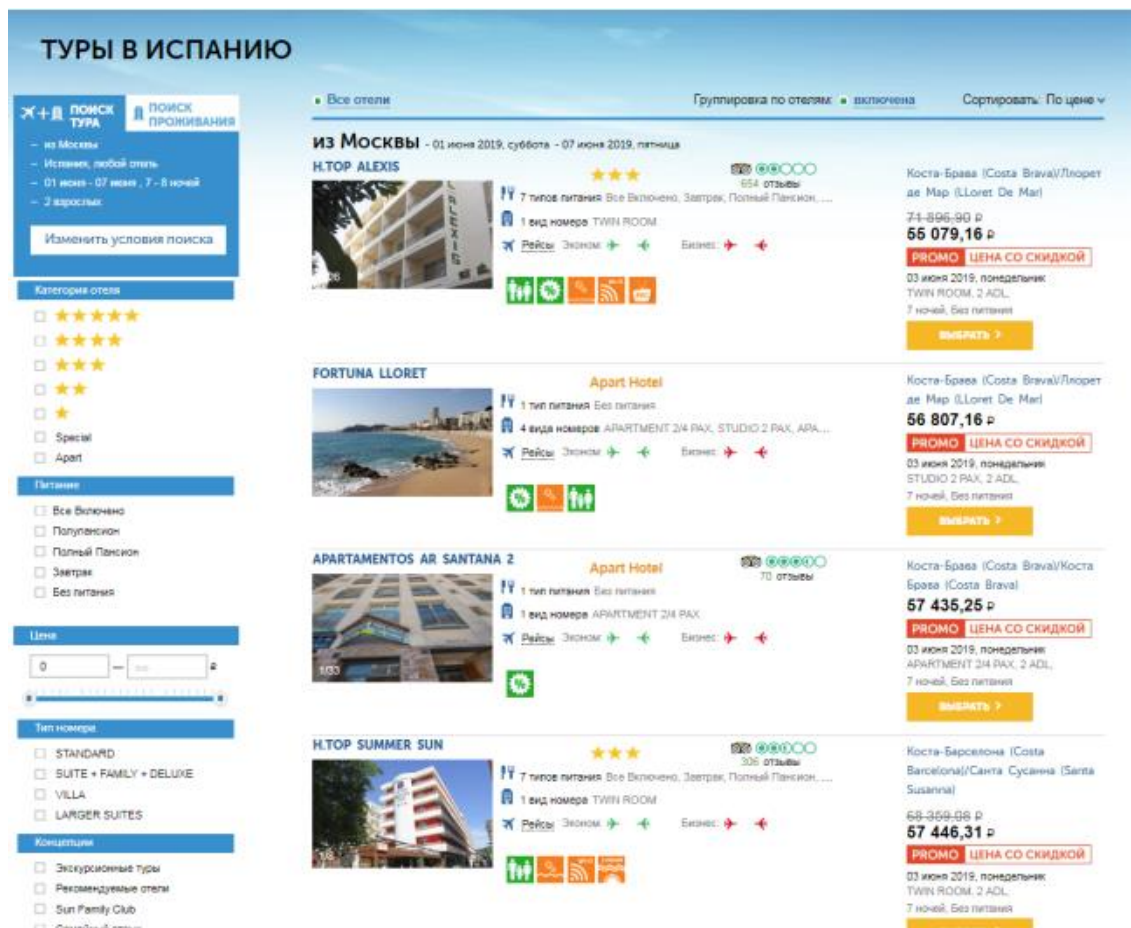


Рисунок 2. Интерфейс подбора туров.

Поиск организуется по следующим параметрам:

- категория отеля (количество звезд);
- питание;
- цена(диапазон);
- тип номера;
- концепции;
- доступность.

Также возможна сортировка по цене, рекомендации, скидке, популярности.



### 1.3.2. Библио-Глобус

Еще одним крупнейшим представителем тур-организаторов на рынке является компания Библио-Глобус. На рисунке 3 представлен интерфейс ее сервиса.

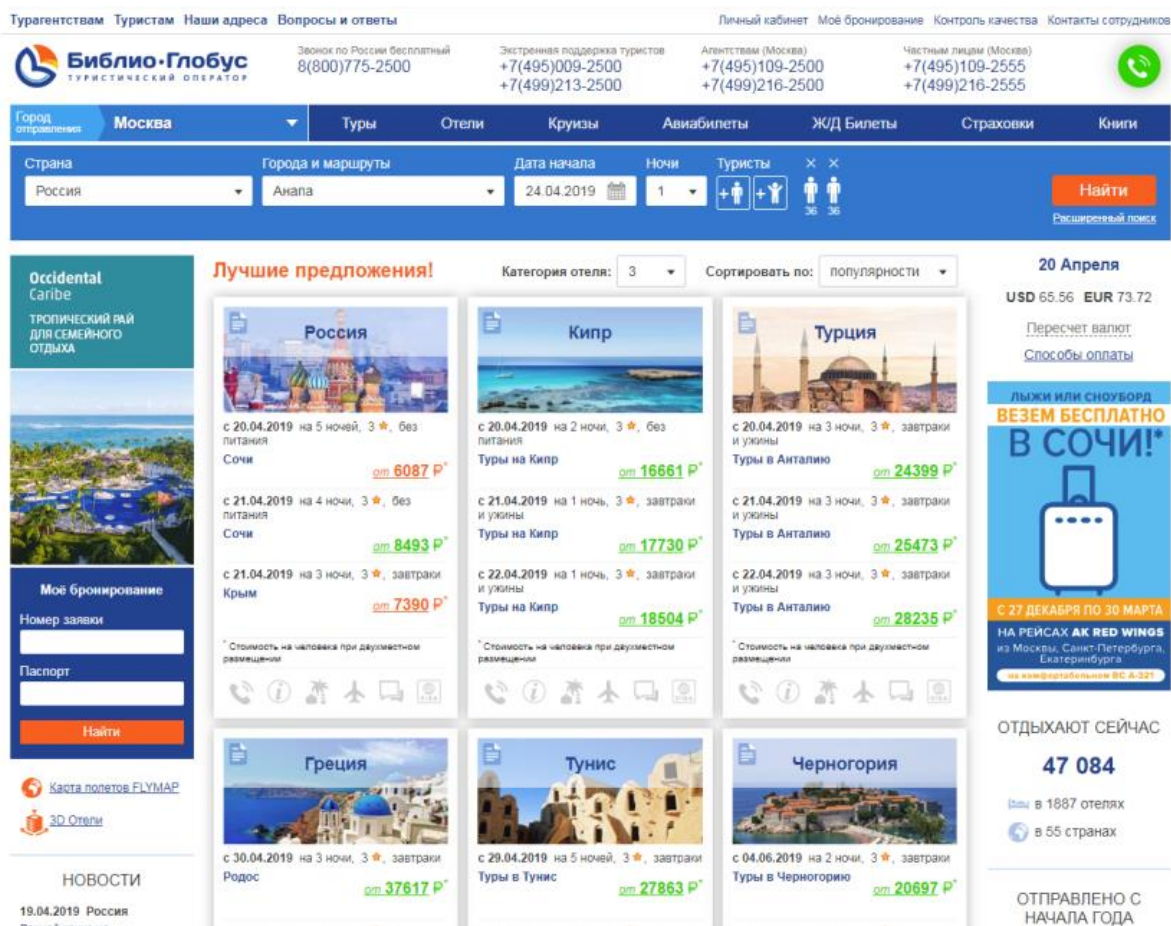


Рисунок 3. Интерфейс сервиса Библио-Глобус.

Пользователю доступны:

- моментальная покупка тура;
- бронирование отеля, авиа и ж/д билетов, страховки;
- вход в личный кабинет.

В результате рассмотрения и анализа турфирм, представленных на рынке сегодня, можно утверждать о наличии высокой конкуренции среди тур

организаторов. Турфирмы вынуждены реагировать и находить новые способы завоевания и удержания клиентов для успешного и эффективного существования на туристическом рынке. И одним из таких способов может послужить десктоп-приложение. В соответствии с результатом анализа, были выдвинуты следующие требования к разрабатываемому приложению:

- поиск тура по направлению, датам, количеству звезд и категории отеля, категории питания, вида трансфера;
- сортировка найденных туров по указанным параметрам;
- бронирование выбранного тура.

#### **1.4. Анализ моделей данных**

Модель данных — это совокупность взаимосвязанных структур данных, операций над ними и множества ограничений для хранимых данных. Далее будут рассмотрены четыре основные модели.

##### **1.4.1. Иерархическая модель**

Одна из первых моделей данных. Иерархические базы данных имеют форму деревьев с дугами-связями и узлами-элементами данных. Иерархическая структура предполагала неравноправие между данными – одни жестко подчинены другим.

Достоинствами этой модели являются простота понимания и использования. Такая модель удобна для работы с иерархически упорядоченной информацией.

Недостаток модели — отсутствие универсальности: для большинства задач требуется дублирование данных, возможна потеря данных, связи «многие-ко-многим» могут быть реализованы только искусственно при избыточности данных [1].

#### **1.4.2. Сетевая модель**

Сетевая модель формулируется теми же терминами, что и иерархическая. Однако наряду с вертикальными реализованы горизонтальные связи и организуется связь один-ко-многим.

Сетевая модель унаследовала многие недостатки иерархической. Она предполагает рассмотрение связей только на уровне групповых отношений, что не позволяет эффективно обеспечивать целостность информации и полностью избавиться от дублирования данных. Также недостатками сетевой модели является жесткость модели, что приводит к необходимости полного пересмотра модели при появлении новых условий в предметной области, требующих внесения изменений в структуры данных [2].

#### **1.4.3. Графовая модель**

Графовую модель данных обычно рассматривают как обобщение сетевой модели данных. Основными элементами модели являются узлы и связи.

В графовых СУБД, как правило, разделяют хранилище и механизм обработки. Для задач с естественной графовой структурой данных графовые СУБД могут существенно превосходить реляционные по производительности, а также иметь преимущества в наглядности представления и простоте внесения изменений в схему базы данных [3].

#### **1.4.4. Реляционная модель**

Реляционная модель появилась вследствие стремления сделать базу данных как можно более гибкой. Построение реляционной модели данных основывается на том понимании, что любой набор данных может быть представлен в виде отношения, оформляемого в форме таблицы, где данные

представляются атрибутами и значениями на пересечении соответствующего атрибута с записью (кортежем).

Основными преимуществами реляционной модели являются представление информации в простой и понятной для пользователя форме и независимость данных от изменения в прикладной программе при изменении.

Из-за широкой популярности реляционной модели, а также в связи с относительной независимостью данных, было принято решение о ее применении к поставленной задаче.

### **1.5. Выводы**

В данном разделе были сформулированы основные требования, выдвигаемые к сервису подбора туров, которые были основаны на рассмотрении и анализе существующих решений. Также был проведен анализ моделей базы данных, в результате которого было принято решение об использовании реляционной модели.

## 2. Конструкторский раздел

В данном разделе будут выполнены описание ролевой модели, формализация сущностей, проектирование базы данных.

### 2.1. Сценарии пользователей

В результате анализа существующих сервисов, были сделаны выводы о необходимости авторизации пользователей для бронирования туров. В связи с этим были выделены следующие роли:

- гость;
- турист;
- менеджер.

#### 2.1.1. Гость

В роли *гостя* пользователю доступен минимальный функционал, необходимый для поиска тура/отеля/питания/трансфера с указанием желаемых параметров. Авторизация гостю не требуется. На рисунке 4 представлен сценарий пользователя в роли гостя.

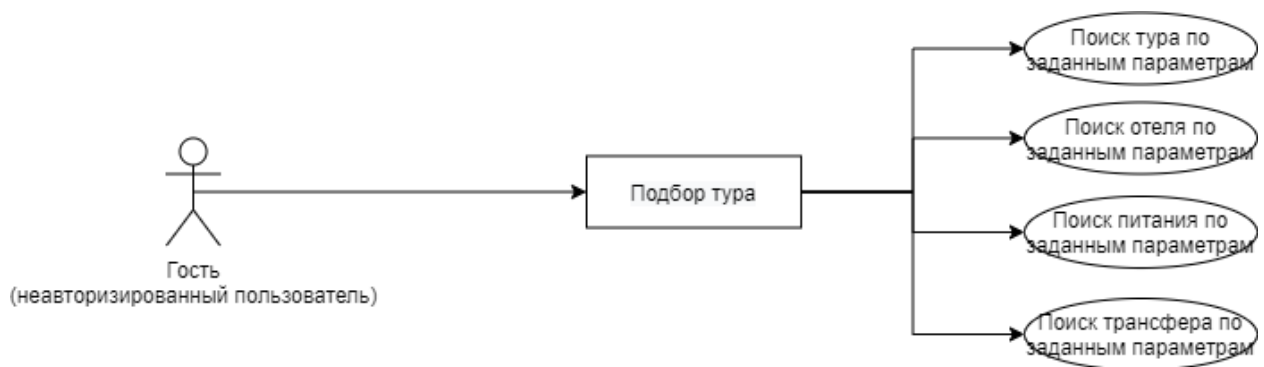


Рисунок 4. Сценарий пользователя в роли гостя.

#### 2.1.2. Турист

В роли *туриста* пользователь может также осуществлять поиск тура. Дополнительно турист может выполнять бронирование и отмену

бронирования указанного тура и просмотр своего забронированного тура/отеля/питания/трансфера. Сценарий пользователя в роли туриста представлен на рисунке 5.



Рисунок 5. Сценарий пользователя в роли туриста.

### 2.1.3. Менеджер

Роль менеджера необходима для управления существующими турами. Помимо просмотра и выбора существующих туров, менеджер может добавлять, изменять и удалять тур/отель/питание/трансфер. На рисунке 6 представлен сценарий пользователя в роли менеджера.

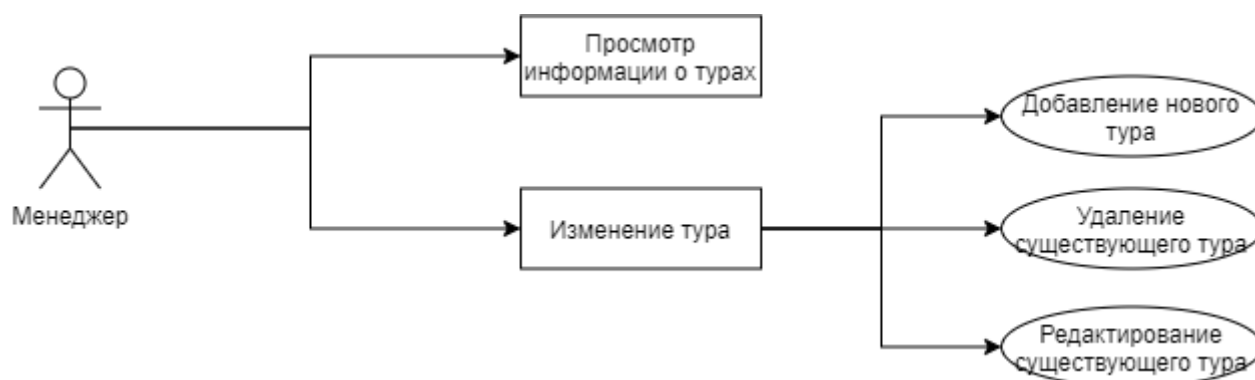


Рисунок 6. Сценарий пользователя в роли менеджера.

## 2.2. Формализация сущностей системы

На рисунке 7 представлена диаграмма сущностей системы.

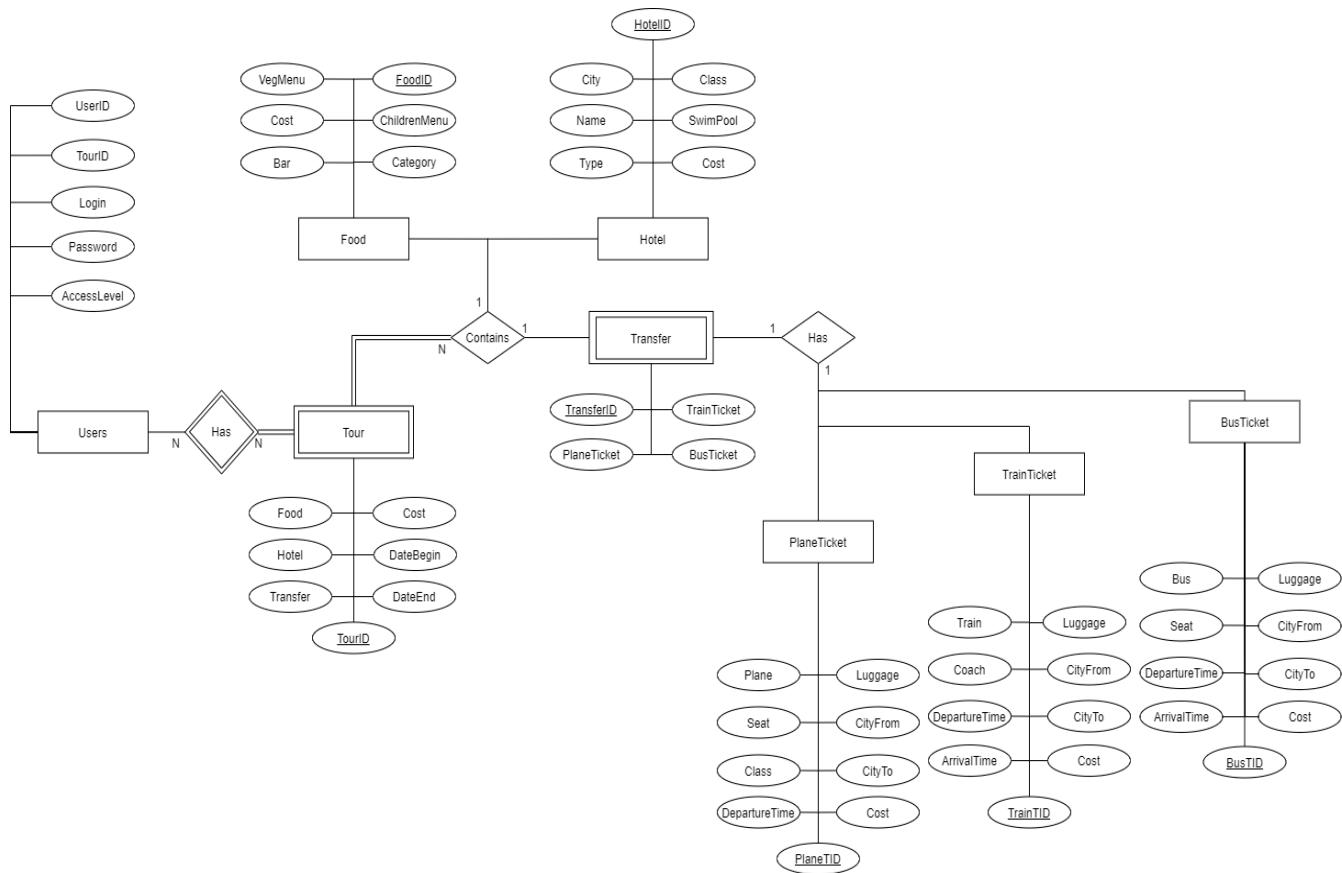


Рисунок 7. Диаграмма сущностей.

В соответствии с диаграммой были выделены следующие сущности:

1. Users – содержит информацию о зарегистрированных пользователях.
2. Tour – содержит информацию о представленных турах.
3. Food – содержит информацию о представленном питании.
4. Hotel – содержит информацию о представленных отелях.
5. Transfer – содержит информацию о представленных трансферах (автобус/самолет/поезд).
6. PlaneTicket – содержит информацию о представленных самолетах.
7. TrainTicket – содержит информацию о представленных поездах.
8. BusTicket – содержит информацию о представленных автобусах.

### 2.3. Ролевая модель

На уровне базы данных выделена ролевая модель, состоящая из 3 ролей. Такой подход позволяет обеспечить дополнительную защиту объектов базы данных.

Роли:

- 1) гость (неавторизированный пользователь);
- 2) турист (авторизированный пользователь);
- 3) менеджер.

Пользователь, использующий сервис в качестве *гостя* имеет следующие полномочия:

- SELECT Tour;
- SELECT Hotel;
- SELECT Food;
- SELECT BusTicket;
- SELECT PlaneTicket;
- SELECT TrainTicket;

Пользователь, прошедший авторизацию под ролью *туриста* обладает всеми полномочиями гостя, а также:

- SELECT/UPDATE Users;
- SELECT Transfer;

Пользователь, использующий сервис в роли *менеджера* обладает следующими правами:



- SELECT/INSERT/UPDATE/DELETE над таблицами Tour, Hotel, Food, Transfer, BusTicket, PlaneTicket, TrainTicket;
- SELECT/UPDATE Users;

## **2.4. Функции**

Поскольку информация о туре содержит данные для работы менеджера, требуется представление этих данных в форме, понятных гостю или туристу. С этой целью была написана функция, выполняющая преобразование внутренних данных, связанных с туром, в формат, понятный остальным пользователям.

На вход функции подается число (TourID), содержащее идентификатор тура. В результате применения JOIN к таблицам Tour, Food, Hotel на выходе формируется пользовательская таблица, содержащая ИД тура, город, название отеля, его тип, категорию питания, трансфер, стоимость и даты.

## **2.5. Вывод**

В данном разделе была сформирована база данных путем рассмотрения сценариев пользователей, формализации сущности системы, рассмотрения ролевой модели и описания функции.

### **3. Технологический раздел**

В данном разделе будут выбраны СУБД и технологический стек, описаны средства реализации поставленной задачи, представлено описание презентационного уровня приложения.

#### **3.1. Анализ СУБД**

СУБД — это набор программ, позволяющий организовывать, контролировать и администрировать базы данных. Самыми популярными на данный момент являются Oracle, MySQL, Microsoft SQL Server, PostgreSQL.

##### **3.1.1. Oracle**

Первая версия этой объектно-реляционной СУБД появилась в конце 70-х, и с тех пор постоянно развивается и дорабатывается, упрощая работу с ней и расширяя функционал. Главными достоинствами Oracle является высокая производительность (возможность группировки транзакций в один пакет) и совместимость с различными ОС.

Однако существенным минусом данной СУБД является высокая стоимость лицензии, поэтому она используется в основном крупными компаниями и корпорациями.

##### **3.1.2. MySQL**

MySQL является одной из самых распространенных СУБД. Главными преимуществами является доступность (распространяется бесплатно), подробная документация. Также пакет MySQL включен в стандартные репозитории наиболее распространённых дистрибутивов операционной системы Linux, что позволяет устанавливать её элементарно. К недостаткам относится невозможность использовать вложенные выборки, триггеры, хранимые процедуры.

### **3.1.3. PostgreSQL**

PostgreSQL — еще одна популярная и бесплатная система. Она универсальна, то есть подойдет для работы с большинством популярных платформ. При этом PostgreSQL — объектно-реляционная СУБД, что дает ей некоторые преимущества над другими бесплатными СУБД, в большинстве являющимися реляционными. Также PostgreSQL предоставляет средства для арифметики с временными величинами и поддерживает схемы, подзапросы, правила, курсоры, триггеры, наследование таблиц [4].

В связи с указанными преимуществами было принято решение об использовании PostgreSQL для разработки системы.

### **3.2. Технологический стек**

Для выполнения проекта был выбран язык C#, так как этот язык объектно-ориентирован и содержит пакет для работы с PostgreSQL.

В качестве среды разработки была выбрана среда Microsoft Visual Studio, поскольку она бесплатная, обеспечивает работу с WinForms, позволяет облегчать процесс написания и отладки кода [5].

### **3.3. Реализация функции базы данных**

В листинге 1 представлена реализация функции FullTour(), принимающей на вход число – ID тура, и возвращающая пользовательскую таблицу с полями ID тура, город, название отеля, его тип, категория питания, трансфер, стоимость и даты.

### Листинг 1 – Реализация функции FullTour.

```
DROP FUNCTION IF EXISTS FullTour;
CREATE FUNCTION FullTour(TID int)
RETURNS TABLE
(
    TourID int,
    City VARCHAR(30),
    Name VARCHAR(30),
    Type VARCHAR (30),
    Category VARCHAR (30),
    Transfer INT,
    Cost INT,
    DateBegin DATE,
    DateEnd DATE
)
AS $$
    SELECT TourID, City, Name, Type, Category, Transfer, Tour.Cost, DateBegin, DateEnd
    FROM Tour JOIN Food ON Tour.Food = Food.FoodID
        JOIN Hotel ON Tour.Hotel = Hotel.HotelID
    WHERE Tour.TourID = TID;
$$ LANGUAGE SQL;
```

### 3.4. Диаграмма базы данных

На рисунке 8 приведена диаграмма базы данных. В приложении А приведен листинг создания таблиц базы данных.

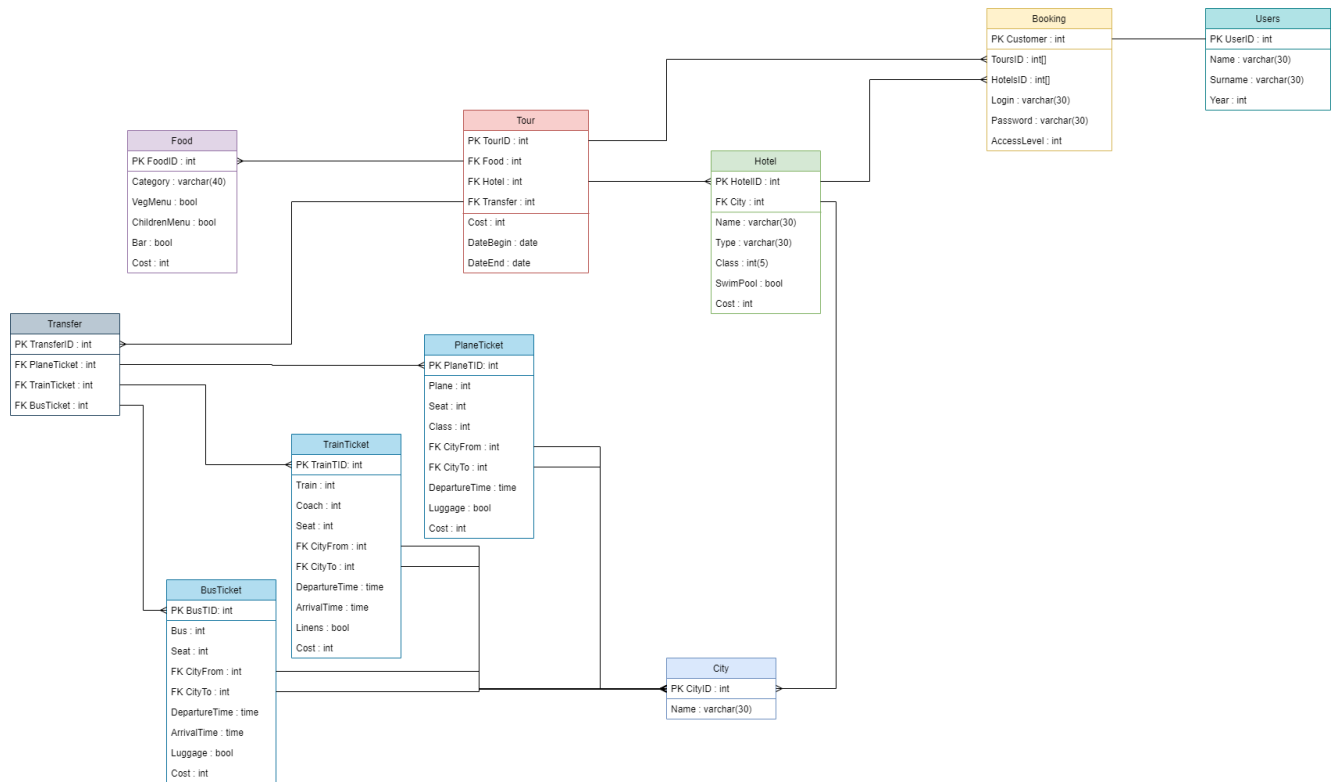


Рисунок 8. Диаграмма базы данных.

### 3.5. Презентационный уровень приложения

В данном пункте будет продемонстрирован презентационный уровень приложения.

На рисунке 9 приведена стартовая форма авторизации.

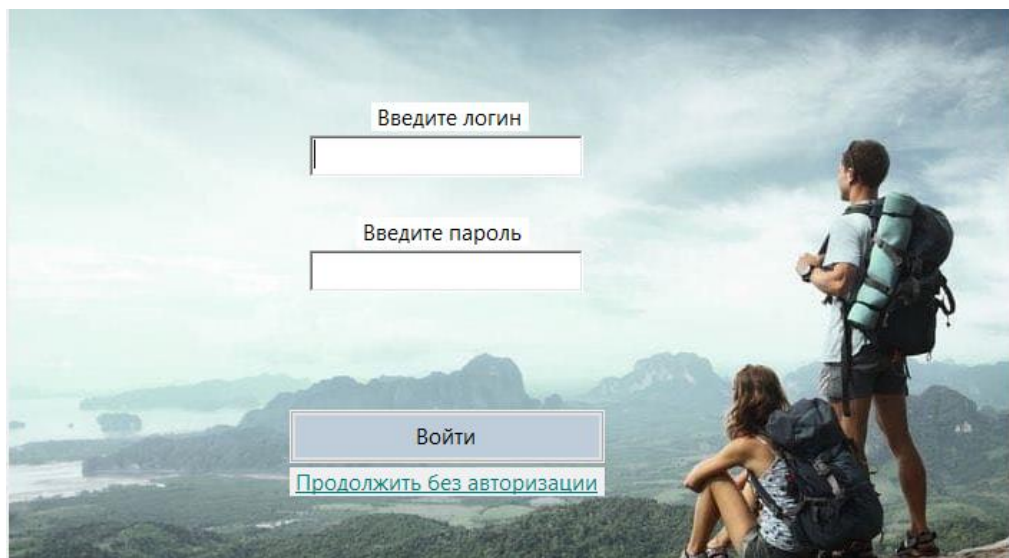


Рисунок 9. Форма авторизации.

В представленной форме пользователь может продолжить без авторизации. Тогда он попадет на форму подбора тура, представленную на рисунке 10.



Рисунок 10. Форма подбора тура.

В результате переключения между вкладками, пользователь может выбрать необходимую. На рисунке 11 представлен внешний вид одной из вкладок.

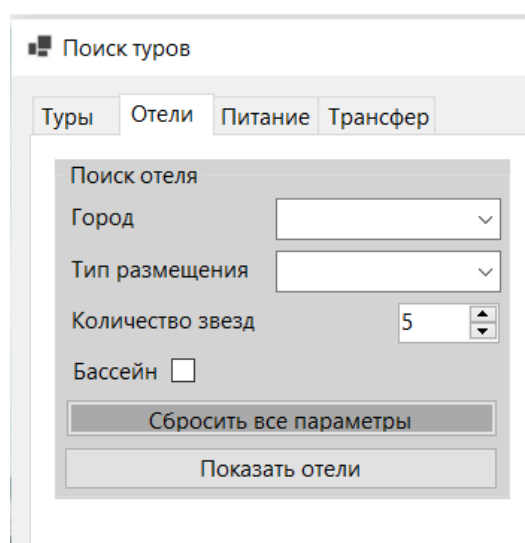


Рисунок 11. Поиск отеля неавторизованным пользователем.

В случае авторизации пользователя в начальной форме в качестве туриста интерфейс поиска туров будет выглядеть следующим образом (рисунки 14-15):

Поиск туров

Туры   Отели   Питание   Трансфер

Поиск туров

Город

Дата начала 1 декабря 2019 г.

Дата конца 1 декабря 2022 г.

Показать туры

Забронированные туры

Показать забронированные туры

Забронировать тур

Отменить бронирование

Рисунок 14. Поиск тура туристом.

Поиск туров

Туры   Отели   Питание   Трансфер

Поиск трансфера

Откуда

Куда

Дата 18 июня 2021 г.

☐ Автобус

☐ Самолет

☐ Поезд

Показать трансфер

Забронированный трансфер

☐ Автобус

☐ Самолет

☐ Поезд

Показать забронированный трансфер

Рисунок 15. Поиск трансфера туристом.

В случае использования сервиса менеджером, интерфейс будет выглядеть так, как показано на рисунке 15. Рисунок 16 отображает форму для добавления/изменения/удаления тура/отеля/питания/трансфера.

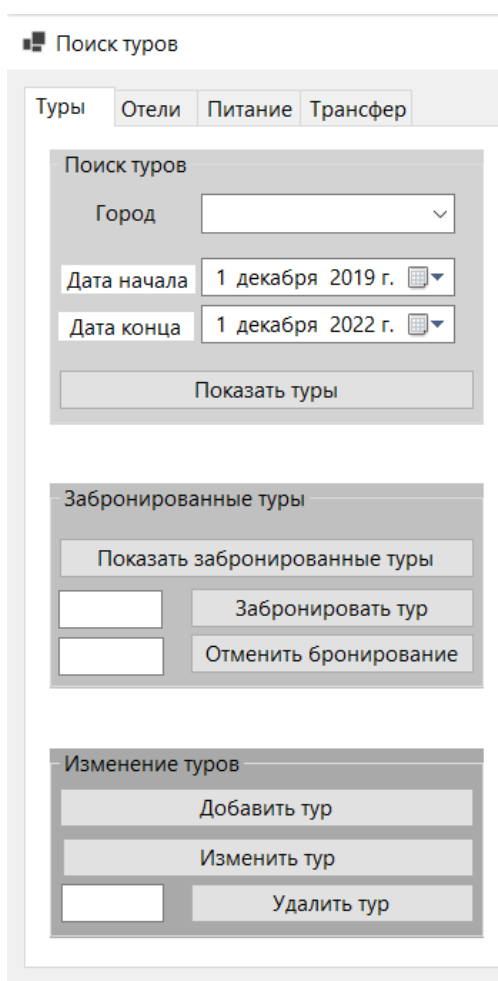


Рисунок 15. Управление туром менеджером.

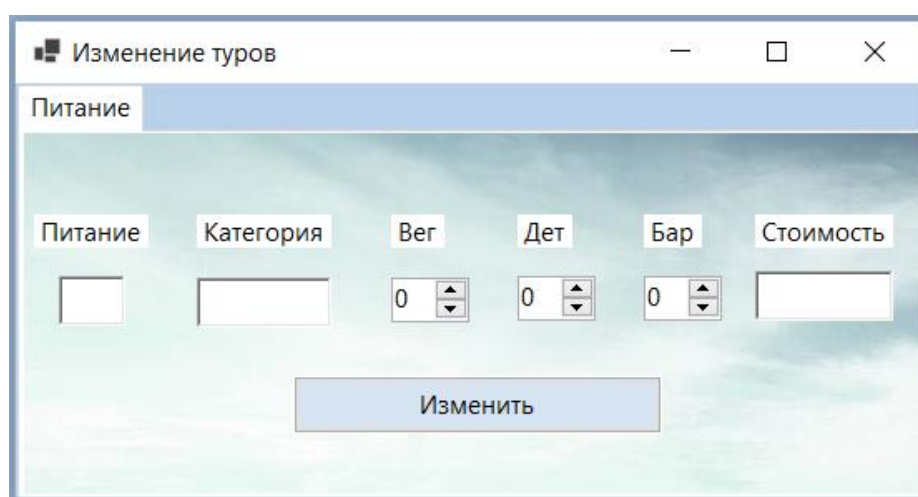


Рисунок 16. Изменение тура менеджером.



### **3.6. Вывод**

В данном разделе в результате рассмотрения и анализа были выбраны СУБД и технологический стек. Также были представлены средства реализации поставленной задачи и представлен презентационный уровень приложения.

## **Заключение**

Цель курсового проекта была достигнута.

Была спроектирована база данных и разработано десктоп-приложение поиска туров по указанным параметрам с возможностью бронирования и изменения туров.

В ходе работы были исследованы существующие решения и подходы к хранению данных. Были формализованы сущности, описана спроектированная база данных, описана ролевая модель.

Был получен опыт разработки базы данных и десктоп-приложений с использованием PostgreSQL и C#.

В дальнейшей разработке приложение и база данных могут быть масштабированы. Так, могут быть добавлены покупка туров, предложение пользователю более выгодных туров, дополнение данных личной информацией пользователя (номер паспорта, любимое направление, количество членов семьи и т.д.).

## Литература

1. Дейт К. Дж. Введение в системы баз данных. – 8-е изд. – М.: «Вильямс», 2006
2. Когаловский М. Р. Перспективные технологии информационных систем. — М.: ДМК Пресс; Компания АйТи, 2003
3. Кодд Е. Реляционная модель данных для больших совместно используемых банков данных [Электронный ресурс]: <http://citforum.ru/database/> (дата обращения 14.06.2021).
4. Документация PostgreSQL [Электронный ресурс] URL: <https://postgrespro.ru/docs/postgresql> (дата обращения 16.06.2021).
5. Microsoft Visual Studio Code [Электронный ресурс] URL: <https://code.visualstudio.com> (дата обращения 15.06.2021).

## Приложение А

Создание таблиц базы данных.

```
CREATE TABLE IF NOT EXISTS Food
```

```
(  
    FoodID INT NOT NULL PRIMARY KEY,  
    Category VARCHAR(40) NOT NULL,  
    VegMenu BOOL NULL,  
    ChildrenMenu BOOL NULL,  
    Bar BOOL NULL,  
    Cost INT NOT NULL  
);
```

```
ALTER TABLE Food
```

```
ADD CONSTRAINT FCost_CHK CHECK (Cost >= 0),  
ADD CONSTRAINT FCat_CHK CHECK (Category = 'Завтрак' OR Category = 'Полупансион' OR  
    Category = 'Полный пансион' OR Category = 'Полный пансион+' OR  
    Category = 'Все включено' OR Category = 'Континентальный завтрак' OR  
    Category = 'Английский завтрак' OR Category = 'Американский завтрак');
```

```
CREATE TABLE IF NOT EXISTS Tour
```

```
(  
    TourID INT NOT NULL PRIMARY KEY,  
    Food INT NOT NULL,  
    Hotel INT NOT NULL,  
    Transfer INT NOT NULL,  
    Cost INT NOT NULL,  
    DateBegin DATE NOT NULL,  
    DateEnd DATE NOT NULL  
);
```

```
ALTER TABLE Tour
```

```
ADD CONSTRAINT FK_TF FOREIGN KEY (Food) REFERENCES Food(FoodID),  
ADD CONSTRAINT FK_TH FOREIGN KEY (Hotel) REFERENCES Hotel(HotelID),  
ADD CONSTRAINT FK_TT FOREIGN KEY (Transfer) REFERENCES Transfer(TransferID);
```

```
ALTER TABLE Tour
ADD CONSTRAINT TCost_CHK CHECK (Cost >= 0),
ADD CONSTRAINT TDate_CHK CHECK (DateBegin <= DateEnd);
```

```
CREATE TABLE IF NOT EXISTS Hotel
(
    HotelID INT NOT NULL PRIMARY KEY,
    Name VARCHAR(50) NOT NULL,
    Type VARCHAR(30) NULL,
    Class INT NULL,
    SwimPool BOOL NULL,
    City VARCHAR(30) NULL,
    Cost INT NOT NULL
);
```

```
ALTER TABLE Hotel
ADD CONSTRAINT UK_H UNIQUE (Name);
```

```
ALTER TABLE Hotel
ADD CONSTRAINT HCost_CHK CHECK (Cost >= 0),
ADD CONSTRAINT HType_CHK CHECK (Type = 'Отель' OR Type = 'Апартамент' OR
    Type = 'Хостел' OR Type = 'Гостевой дом' OR
    Type = 'Мотель' OR Type = 'Вила' OR
    Type = 'Курортный отель' OR Type = 'Кемпинг' OR Type = 'Постель и завтрак'),
ADD CONSTRAINT HClass_CHK CHECK (Class >= 0 AND Class <= 5);
```

```
CREATE TABLE IF NOT EXISTS Users
(
    UserID INT NOT NULL PRIMARY KEY,
    ToursID INT[],
    Login VARCHAR(30),
    Password VARCHAR(30),
    AccessLevel INT NULL
);
```

```
ALTER TABLE Users
```

```
ADD CONSTRAINT UAL_CHK CHECK (AccessLevel >= 0 AND AccessLevel <= 2);
```

```
CREATE TABLE IF NOT EXISTS Transfer
```

```
(  
    TransferID INT NOT NULL PRIMARY KEY,  
    PlaneTicket INT NULL,  
    TrainTicket INT NULL,  
    BusTicket INT NULL  
);
```

```
ALTER TABLE Transfer
```

```
ADD CONSTRAINT FK_TP FOREIGN KEY (PlaneTicket) REFERENCES PlaneTicket(PlaneTID),
```

```
ADD CONSTRAINT FK_TT FOREIGN KEY (TrainTicket) REFERENCES TrainTicket(TrainTID),
```

```
ADD CONSTRAINT FK_TB FOREIGN KEY (BusTicket) REFERENCES BusTicket(BusTID);
```

```
CREATE TABLE IF NOT EXISTS PlaneTicket
```

```
(  
    PlaneTID INT NOT NULL PRIMARY KEY,  
    Plane INT NULL,  
    Seat INT NULL,  
    Class INT NULL,  
    CityFrom VARCHAR(30) NULL,  
    CityTo VARCHAR(30) NULL,  
    DepartureTime TIMESTAMP NULL,  
    Luggage BOOL NULL,  
    Cost INT NULL  
);
```

```
ALTER TABLE PlaneTicket
```

```
ADD CONSTRAINT PTCost_CHK CHECK (Cost >= 0),
```

```
ADD CONSTRAINT PTClass_CHK CHECK (Class = 1 OR Class = 2);
```

```
CREATE TABLE IF NOT EXISTS TrainTicket
```

```
(
```

```

TrainTID INT NOT NULL PRIMARY KEY,
Train INT NULL,
Coach INT NULL,
Seat INT NULL,
CityFrom VARCHAR(30) NULL,
CityTo VARCHAR(30) NULL,
DepartureTime TIMESTAMP NULL,
ArrivalTime TIMESTAMP NULL,
Linens BOOL NULL,
Cost INT NULL
);

ALTER TABLE TrainTicket
ADD CONSTRAINT TTCost_CHK CHECK (Cost >= 0),
ADD CONSTRAINT TTCoach_CHK CHECK (Coach >= 1 AND Coach <= 20),
ADD CONSTRAINT TTTime_CHK CHECK (DepartureTime < ArrivalTime);

CREATE TABLE IF NOT EXISTS BusTicket
(
    BusTID INT NOT NULL PRIMARY KEY,
    Bus INT NULL,
    Seat INT NULL,
    CityFrom VARCHAR(30) NULL,
    CityTo VARCHAR(30) NULL,
    DepartureTime TIMESTAMP NULL,
    ArrivalTime TIMESTAMP NULL,
    Luggage BOOL NULL,
    Cost INT NULL
);

ALTER TABLE BusTicket
ADD CONSTRAINT BTCost_CHK CHECK (Cost >= 0),
ADD CONSTRAINT TTTime_CHK CHECK (DepartureTime < ArrivalTime);

```

## **Приложение Б**

Презентация.