**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ 4

1 АНАЛИТИЧЕСКАЯ ЧАСТЬ 5

1.1 Анализ предметной области 5

1.2 Постановка задачи 5

2 РАЗРАБОТКА ЛОГИЧЕСКОЙ МОДЕЛИ БАЗЫ ДАННЫХ 7

2.1 Построение диаграммы «сущность-связь» в нотации П.Чена 7

2.2 Построение модели основанной на ключах 7

2.3 Построение полной атрибутивной модели 8

3 РАЗРАБОТКА ФИЗИЧЕСКОЙ МОДЕЛИ БАЗЫ ДАННЫХ 10

3.1 Выбор аппаратной и программной платформы для реализации БД 10

3.2 Реализация базы данных 10

3.3 Тестирование базы данных 11

3.4 Разграничение прав доступа 12

3.5 Расчет информационных параметров базы данных 13

4 РАЗРАБОТКА КЛИЕНТСКОГО ПРИЛОЖЕНИЯ 15

4.1 Обоснование выбора языка программирования 15

4.2 Разработка интерфейса пользователя 15

4.3 Алгоритм работы каждого из модулей 16

4.4 Тестирование работы приложения 17

ЗАКЛЮЧЕНИЕ 18

СПИСОК ЛИТЕРАТУРЫ И ИНФОРМАЦИОННЫХ РЕСУРСОВ 19

ПРИЛОЖЕНИЕ А Иллюстративный материал 20

ПРИЛОЖЕНИЕ Б SQL скрипт создания базы данных 31

ПРИЛОЖЕНИЕ В Код клиентского приложения 34

**ВВЕДЕНИЕ**

Концепция организации информации в виде баз данных является значительным фактором при создании различных систем автоматизированной обработки информации. Проектирование подобных программно-технических компонентов информационных систем является комплексной задачей, включающей широкий спектр вопросов, начиная от адекватного моделирования предметной области, до выбора необходимых технических и программных средств, написания эргономических интерфейсов и т.д.

В данной курсовой работе предлагается спроектировать базу данных, начиная от ее логического проектирования на бумаге, и до момента создания физической модели базы данных, которая сможет полностью реализовать принципы работы логической модели. Для достижения поставленной цели потребуется применить знания, полученные в области программирования, проектирования и администрирования баз данных.

Данная курсовая работа является актуальной, так как на данный момент появляется и существует большое количество агенств, отправляющих своих клиентов в путешествия, соответственно им необходима систематизация данных, в чем и может помочь результат курсового проекта.

**1 АНАЛИТИЧЕСКАЯ ЧАСТЬ**

**1.1 Анализ предметной области**

Турфирма - это организация, позволяющая ее клиентам оформлять туры и путевки в разные страны для отдыха и приятного провождения времени.

Сотрудник, он же администратор, должен вести учет клиентов и туров, а также их составных элементов, таких как: отели, страны, скидки.

Администратор должен иметь возможность работать со всеми этими элементами и совершать такие действия:

* регистрировать туры и клиентов;
* обновлять информацию об отелях и направлениях отдыха;
* организация скидок на туры;
* при необходимости оформлять возврат денег в соответствии с причиной.

Алгоритм работы фирмы: клиент оформляет заявку, администратор регистрирует тур, опционально может быть совершен возврат.

**1.2 Постановка задачи**

Приложение создано для администраторов турфирмы для ведения учета информации о клиентах, турах, сотрудниках, скидках, возвратах, отелях.

Также с помощью этого приложения клиент может совершить заявку для заказа тура, которую обработает администратор. Для этого мы должны разграничить права доступа.

Для реализации проекта выделим сущности:

* Тур (ID тура, Вид тура, Стоимость, Дата отправления, Длительность, ID отеля, ID клиента, ID скидки)
* Отель (ID отеля, Название, Адрес, Класс отеля)
* Скидки (ID скидки, Размер скидки)
* Клиент (ID клиента, Количество персон, Количество детей, Паспорт, ФИО клиента, Номер телефона)
* Сотрудник (ID сотрудника, ФИО сотрудника, Телефон сотрудника)
* Возврат (ID возврата, Дата возврата, Причина возврата, Сумма возврата, Длительность, ID тура, ID клиента, ID сотрудника)
* Сводка продаж (ID сводки, Количество продаж, ID сотрудника)

Приложение должно включать в себя такие возможности как создание, редактирование и удаление записей, поддержание актуальной информации об отелях и скидках для сотрудников, создание заявки на оформление для клиентов, а также иметь понятный и удобный интерфейс.

**2 РАЗРАБОТКА ЛОГИЧЕСКОЙ МОДЕЛИ БАЗЫ ДАННЫХ**

**2.1 Построение диаграммы «сущность-связь» в нотации П.Чена**

Для построения диаграммы “сущность-связь” в нотации Чена, нужно выделить связи между сущностями. Для этого была сформирована сложная сетевая структура, основанная на основных сущностях предметной области. Разработанная структура представлена на рисунке 1 Приложения А.

На основе сложной сетевой структуры была создана простая сетевая структура путём добавления промежуточных таблиц для того, чтобы связь многие ко многим трансформировать в связь один ко многим (рисунок 2).

Нормализация не может быть произведена в простой сетевой структуре, поэтому необходимо привести сетевую модель к иерархической. Иерархическая (древовидная) структура не допускает наличия нескольких родителей у одного потомка. Полученная древовидная структура представлена на рисунке 3 Приложения А.

Далее на основе прошлых таблиц была синтезирована ER-диаграмма (рисунок 4 Приложения А), отображающая взаимосвязь объектов, а также основные элементы взаимодействующих объектов.

**2.2 Построение модели основанной на ключах**

Основной целью модели, основанной на ключах, является широкий обзор структур данных и ключей, нужных для поддержки определенной области. Эта модель определяет контекст, в котором могут быть созданы подробные модели, пригодные для конкретного воплощения.

Модель, основанная на ключах строится на основе ERD, но отображает больше деталей: ключевые поля, внешние ключи.

Разработанная модель представлена на рисунке 5 Приложения А.

**2.3 Построение полной атрибутивной модели**

Полная атрибутивная модель достигается нормализацией отношений до третьей или четвёртой нормальной формы.

Целью нормализации базы данных является сокращение избыточности. Нормализация предполагает последовательное приведение схемы базы данных к так называемым нормальным формам, каждая последующая из которых предъявляет более строгие требования по сравнению с предыдущей.

Рассмотрим все выделенные нами сущности, и проверим их на атомарность. В каждой из сущности соблюдается атомарность, следовательно, делаем вывод о том, что наша база данных находится в первой нормальной форме.

Так как наша база данных, находится в первой нормальной форме, рассмотрим её на удовлетворение второй нормальной форме (отношение R находится во второй нормальной форме в том и только в том случае, когда находится в 1НФ и каждый не ключевой атрибут полностью зависит от первичного ключа). Каждая сущность содержит уникальный, искусственно введенный, первичный ключ, не обладающий смысловой нагрузкой, от которого полностью зависят все не ключевые атрибуты, это удовлетворяет условиям второй нормальной форме, следовательно, делаем вывод о том, что база данных находится во второй нормальной форме.

Так как наша база данных находится во второй нормальной форме, рассмотрим на её удовлетворение третьей нормальной форме (отношение R находится в третьей нормальной форме в том и только в том случае, если находится в 2НФ и каждый не ключевой атрибут не транзитивно зависит от первичного ключа). Во всех созданных сущностях транзитивные зависимости отсутствуют, что говорит о том, что наша база данных находится в третьей нормальной форме.

В ходе нормализации была построена полная атрибутивная модель в нотации IDEF1X, которая представлена на рисунке 6 Приложения А.

**3 РАЗРАБОТКА ФИЗИЧЕСКОЙ МОДЕЛИ БАЗЫ ДАННЫХ**

**3.1 Выбор аппаратной и программной платформы для реализации базы данных**

В качестве СУБД была выбрана MySQL. MySQL – свободно распространяемая СУБД с открытым кодом. MySQL – кроссплатформенная система. Ее можно использовать практически во всех современных операционных системах, в том числе Windows, Linux, Mac OS, Solaris, HP-UX и др.

MySQL имеет отличные технические характеристики: многопоточность, многопользовательский доступ, быстродействие, масштабируемость (компания-разработчик приводит пример MySQL-сервера, который работает с 60 тыс. таблиц, содержащими приблизительно 5 млрд строк). Все эти особенности сделали MySQL универсальной СУБД, используемой на серверах хост-провайдеров и идеально подходит по нашу базу данных и web-сайт.

**3.2 Реализация базы данных**

Далее нам нужно реализовать спроектированную базу данных с помощью выбранной СУБД. Построим физическую модель базы данных (рисунок 7 Приложения А).

**3.3 Тестирование базы данных**

После того, как база данных была реализована, ее необходимо протестировать. Для этого применим к созданной базе данных запросы и проверим корректность их выполнения.

Добавим в таблицу “Отели” новую запись. Результат представлен на рисунке 1.:

INSERT INTO отель VALUES (NULL, 'Santhiya Resort', 'Тайланд, Surat Thani', '5 звезд');



Рисунок 1 - Результат добавления записи

Обновим поле “фио\_сотрудника” в таблице “Сотрудник”. Результат представлен на рисунке 2.:

UPDATE сотрудник SET `фио\_сотрудника` = 'Каширин Н.Г.' WHERE сотрудник.`id\_сотрудника` = 2;



Рисунок 2 - Результат обновления записи

Проверим ссылочную целостность базы данных, например, удаление атрибута, который является внешним ключом.

Удалим запись в таблице “Клиент” и выполним поиск по id в таблице “Тур”, чтобы удостовериться, что каскадные ограничения ссылочной целостности работают корректно.

SQL запрос удаления в родительской таблице:

DELETE FROM клиент WHERE клиент.id\_клиента = 8;

SQL запрос поиска по id удаленному клиенту в таблице туров, результат которого изображен на рисунке 3:

SELECT \* FROM тур WHERE `id\_клиента` = 8;

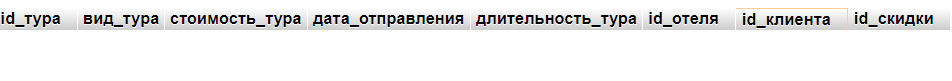


Рисунок 3 – Результат поиска выдачи по удаленному клиенту

Исходя их этого теста наблюдаем, что при изменении первичных ключей в родительских таблицах, их внешние ключи меняются в соответствии с установленными каскадными ограничениями.

**3.4 Разграничения прав доступа**

Для проекта были выделены две учетные записи – Сотрудник и Клиент.

Сотрудник: позволяет просматривать всю имеющуюся информацию в таблицах, вносить изменения в записи, добавлять и удалять записи.

Клиент: позволяет отправить заявку, которая в виде записи добавится в таблицу “Клиент”. Просмотр информации клиенту недоступен.

**3.5 Расчет информационных параметров базы данных**

Длина логической записи j-ого файла определяется как сумма длин полей, её составляющих и указателей, определяется по формуле:

|  |  |
| --- | --- |
| [байт], | (1) |

где – число групп полей в записях; – длина группы [байт].

Для переменных длин полей возьмем максимальную длину поля, следовательно, размер типов данных для хранения в памяти будет составлять:

1. INTEGER – 4 байта
2. VARCHAR(x) – x байт
3. DATE – 3 байта

Вычислим:

Объем памяти, необходимый для размещения информационного фонда без учёта системных данных и указателей определяется по формуле:

|  |  |
| --- | --- |
| [байт], | (2) |

где – число типов записей в информационном фонде; – количество записей -го файла.

Предположим, что в базе содержится:

7 туров, 7 клиентов, 5 сотрудников, 2 возврата, 5 скидок, 5 отелей и 5 сводок продаж и 7 записей клиент\_тур, тогда

Приращение информационного фонда вычисляется как:

|  |  |
| --- | --- |
| [байт-1], | (3) |

где – число добавленных типов записей; – интенсивность добавления записей в файл -го типа.

Получаем:

= (байт-1)

Время заполнения информационного фонда определяется как:

|  |  |
| --- | --- |
| [время], | (4) |

Получаем:

(дней)

Время резервного копирования определяется интенсивностью отказов, сопровождающихся потерей данных:

(дней)

**4 РАЗРАБОТКА КЛИЕНТСКОГО ПРИЛОЖЕНИЯ**

**4.1 Обоснования выбора языка программирования**

В качестве языка программирования был выбран PHP. Это язык программирования, используемый на стороне WEB-сервера для динамической генерации HTML-страниц.

Очень важное преимущество PHP заключается в его «движке». «Движок» PHP не является ни компилятором, ни интерпретатором. Он является транслирующим интерпретатором. Такое устройство «движка» PHP позволяет обрабатывать сценарии с достаточно высокой скоростью.

Сценарий РНР может состоять из 10 000 строк или из одной строки -- все зависит от специфики задачи. Нет необходимости подгружать библиотеки или указывать специальные параметры. Механизм РНР начинает выполнять код после первой экранирующей последовательности (<?) и продолжает выполнение до того момента, когда он встретит парную экранирующую последовательность (?>). Если код имеет правильный синтаксис, он исполняется в точности так, как указал программист.

Одними из его преимуществ является доступность и бесплатность, а также богатый инструментарий и естественная поддержка многими популярными Web-серверами.

**4.2 Разработка интерфейса пользователя**

В разработанном web-приложении были спроектированы страницы для работы с каждой из таблиц базы данных. Для сотрудника турфирмы в общем виде – это навигационная панель для переключения между таблицами и сама таблица с полями для ввода для редактирования, создания и удаления записей.

Для клиента – это страница с выбором типа тура, приводящая к форме для заполнения заявки.

При запуске приложения, пользователя встречает начальная страница, изображенная на рисунке 7 Приложения А, где можно выбрать учетную запись, согласно разграниченным правам доступа.

При нажатии на иконку сотрудника, пользователь переходит на первую страницу с таблицей “Туры”. В ней администратор ведет учет записей, их создание, редактирование и удаление. Страница “Туры” изображена на рисунке 8 Приложения А.

Тот же самый функционал ожидает пользователя на всех страницах. Их скриншоты показаны на рисунках 9-15 Приложения А.

Для пользователя все обстоит немного проще, при нажатии кнопки учетной записи пользователя, выполняется переход на страницу, изображенную на рисунке 16 Приложения А, которая содержит информацию о турах.

При нажатии кнопки забронировать, приложение открывает страницу с формой для заполнения данных клиента, которые отправляются в таблицу “Клиент” базы данных. Страница изображена на рисунке 17 Приложения А.

**4.3 Алгоритм работы каждого из модулей**

При загрузке страницы приложение подключается к базе данных и с помощью php-скрипта записи из таблицы выводятся на экран.

При нажатии на кнопку редактирования, нижние формы заполняются актуальными данными и пользователь получает возможность их изменять. Для этого используется метод POST, отправляющий данный из заполненных форм на сервер. Так же работает и добавление записей.

Удаление производится похожим образом, формируется форма для кнопки удаления и при нажатии SQL запрос отправляется на сервер.

**4.4 Тестирование работы приложения**

Для выявления дефектов работы приложения произведем все доступные операции с данными.

Для начала добавим новую запись в таблицу “Клиент”. Данный процесс показан на рисунке 18 Приложения А.

Вместе с этим добавим и тур для нового клиента, что изображено на рисунке 19 Приложения А.

Если сотрудник допустил ошибку или обновилась информация, то он может изменить запись, нажав на кнопку с изображением карандаша. Изменение записи и результат показаны на рисунках 20 и 21 Приложения А.

И наконец, проверим функцию удаления записи. Произведем удаление последней записи в таблице “Тур”. Результат приведен на рисунке 22 Приложения А.

В результате тестирования приложения не было выявлено дефектов, вызывающих нарушение структуры базы данных, приложение выполняет необходимые действия по добавлению, изменению и удалению данных в таблицах.

**ЗАКЛЮЧЕНИЕ**

В данном курсовом проекте была спроектирована и реализована реляционная база данных на тему «База данных турфирмы». Было написано web-приложение для работы и администрирования базы данных и имеющее простой и удобный интерфейс. Также были выделены учетные записи, организующие разграничение прав доступа.

В ходе проектирования база данных была приведена к третьей нормальной форме. Была реализована ее физическая модель.

Были закреплены навыки проектирования логической модели базы данных, проведения их нормализации, построения физической модели и разработки программного обеспечения.

В итоге, было получено web-приложение, позволяющее вести учет данных о клиентах, турах и сотрудниках, а также дающее администратору возможность создавать, удалять и редактировать записи, а клиенту отправлять заявку на оформление тура. Приложение подходит для использования в небольших турфирмах и турагенствах.

**СПИСОК ЛИТЕРАТУРЫ И ИНФОРМАЦИОННЫХ РЕСУРСОВ**

1. Дейт К.Дж. Введение в системы баз данных, 7-е издание.:Пер. с англ./ К.Дж.Дейт.– М.: Издательский дом «Вильямс», 2002. – 1072с.

2. Ульман Д. Основы систем баз данных/Д.Ульман.– М.: Финансы и статистика, 1983. – 334 с.

2. Мартин Дж. Организация баз данных в вычислительных системах/ Дж.Мартин.– М.: «Мир», 1978.–615с.

4. Голицына О.Л., Партыка Т.Л., Попов И.И. Основы проектирования баз данных; Форум – Москва, 2012. – 416 с.

**ПРИЛОЖЕНИЕ А**

**Иллюстративный материал**

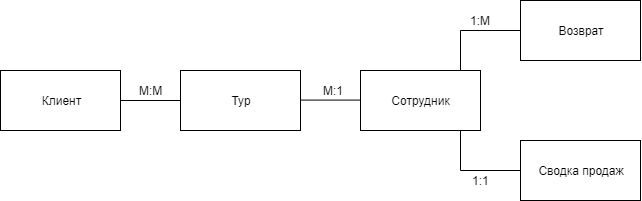


Рисунок A 1 – Сложная сетевая структура

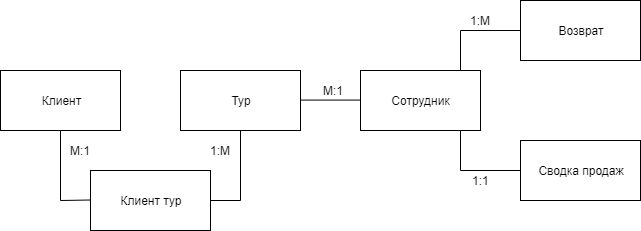


Рисунок A 2 – Простая сетевая структура

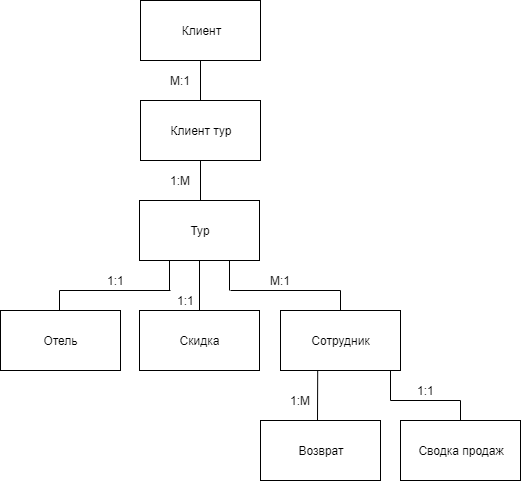


Рисунок A 3 – Древовидная структура

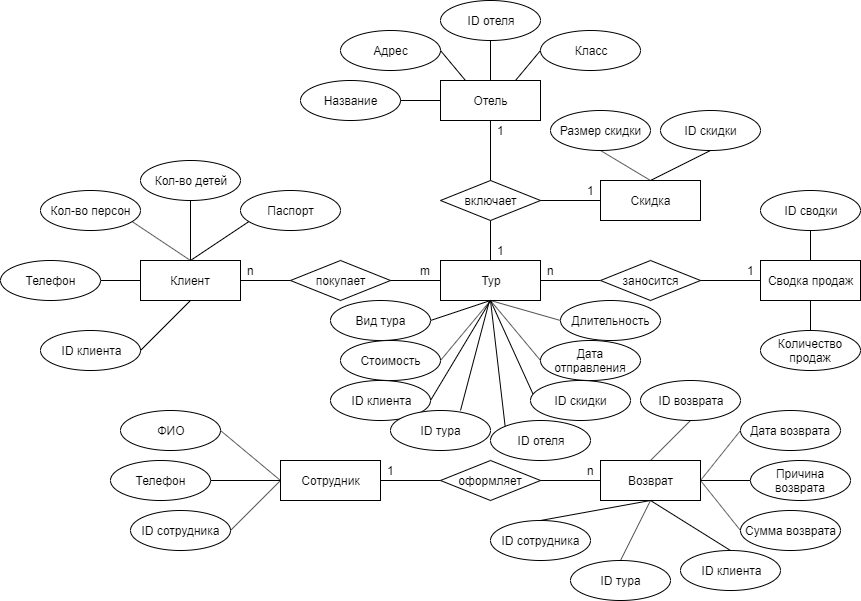


Рисунок A 4 – ER диаграмма в нотации П.Чена

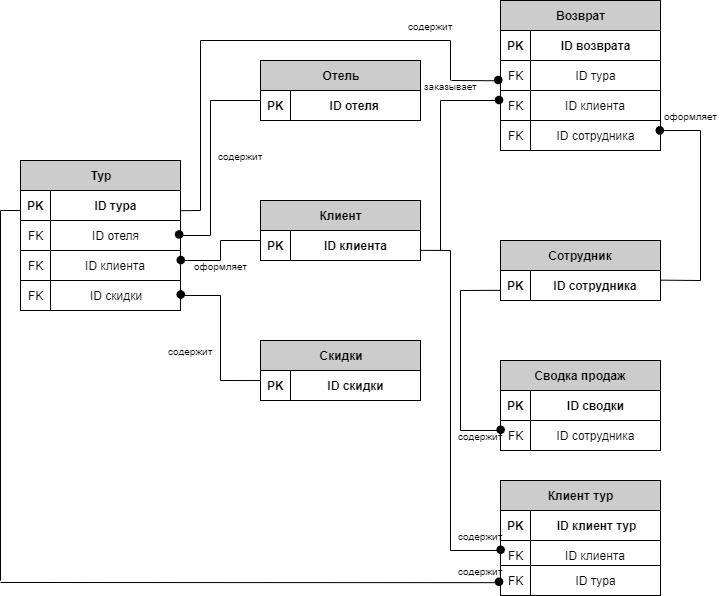


Рисунок A 5 – Модель, основанная на ключах

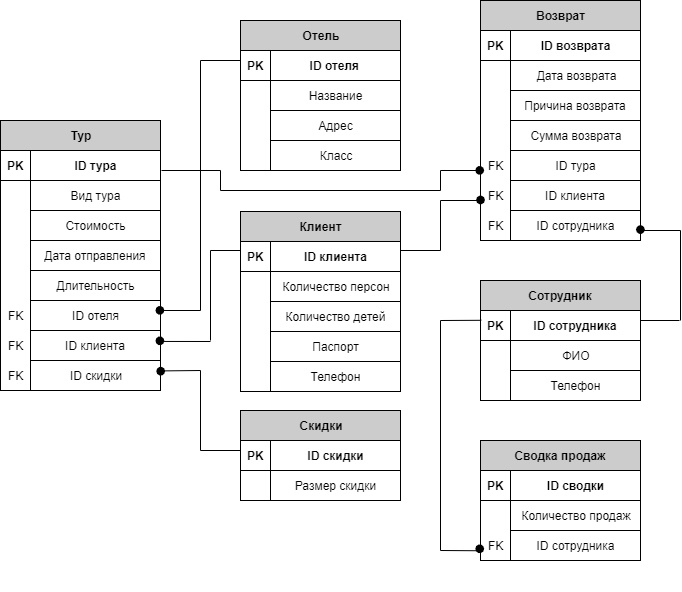


Рисунок A 6 – Полная атрибутивная модель

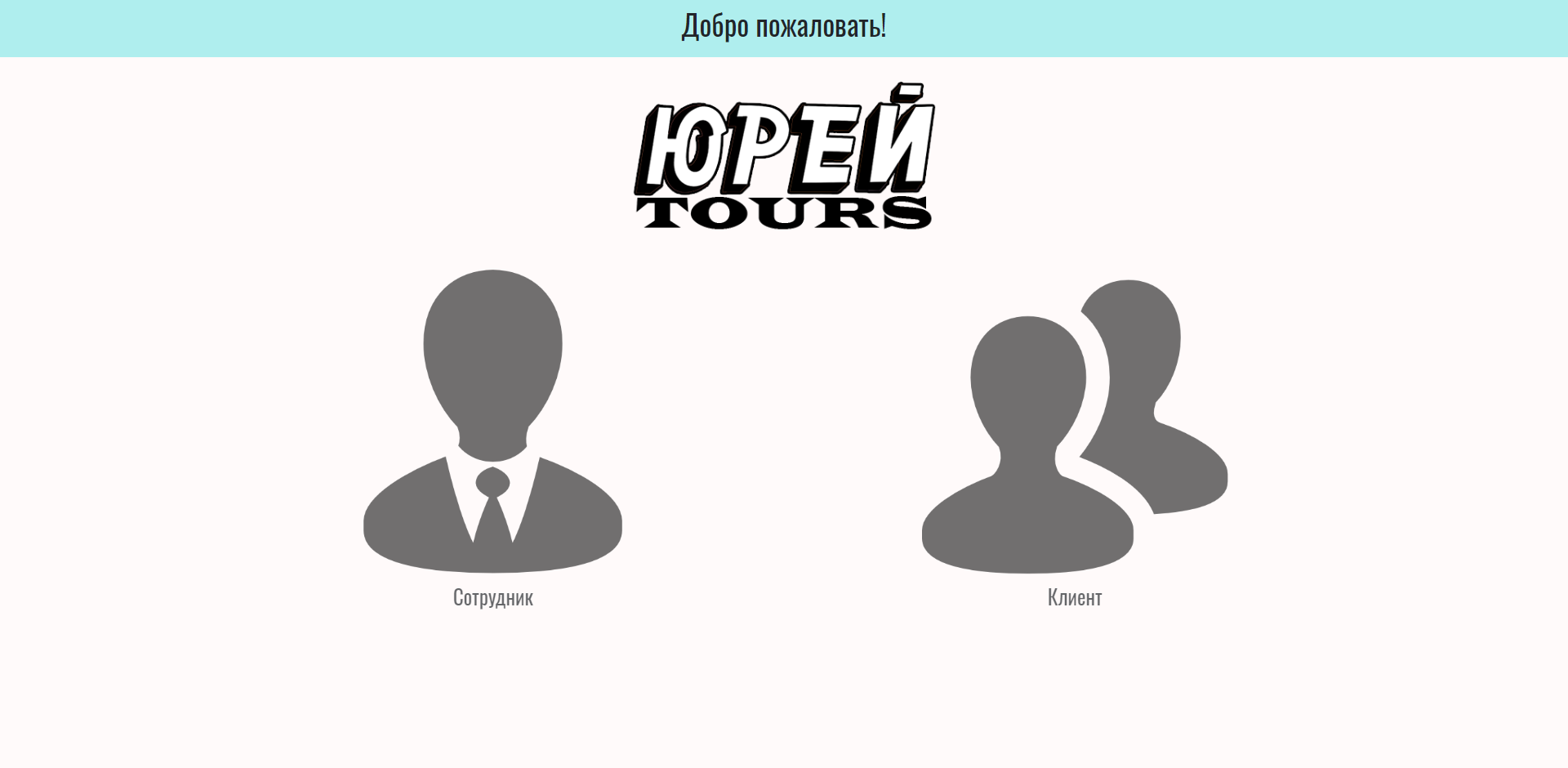


Рисунок А 7 – Выбор учетной записи на начальной странице

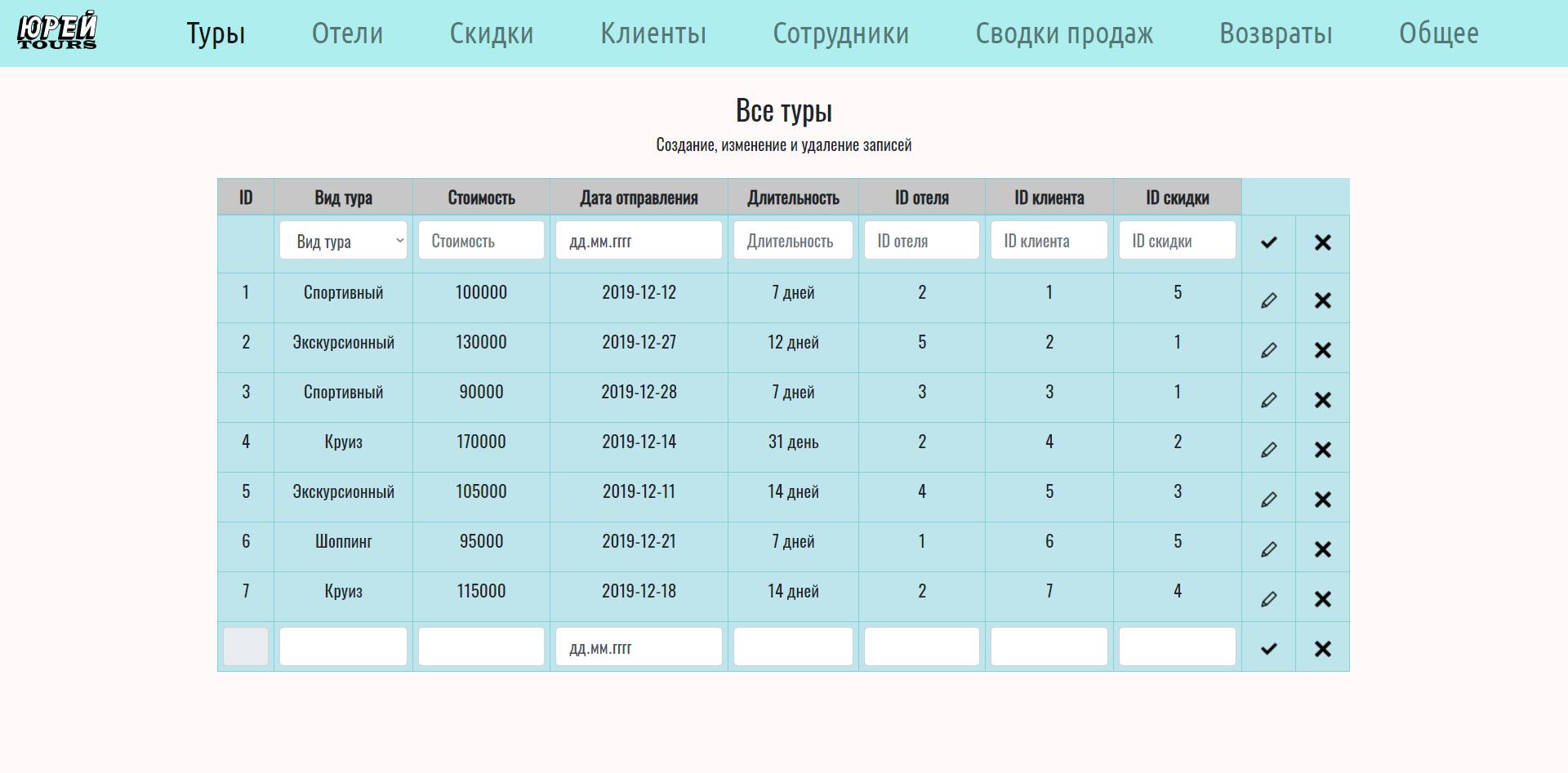


Рисунок А 8 – Страница Туры



Рисунок А 9 – Страница Отели

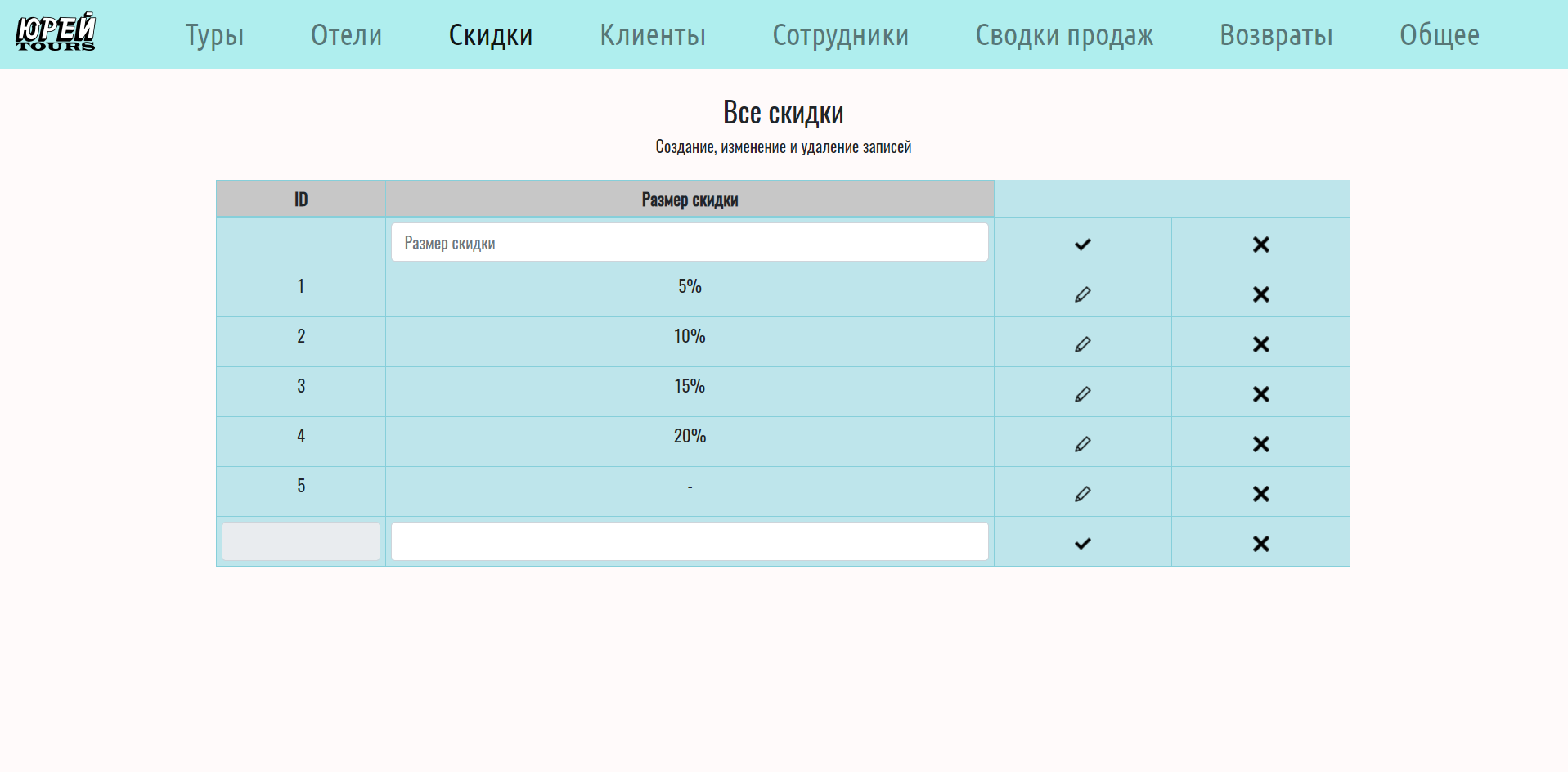


Рисунок А 10 – Страница Скидки



Рисунок А 11 – Страница Клиенты



Рисунок А 12 – Страница Сотрудники



Рисунок А 13 – Страница Сводки продаж



Рисунок А 14 – Страница Возвраты



Рисунок А 15 – Страница Общее



Рисунок А 16 – Страница Информация

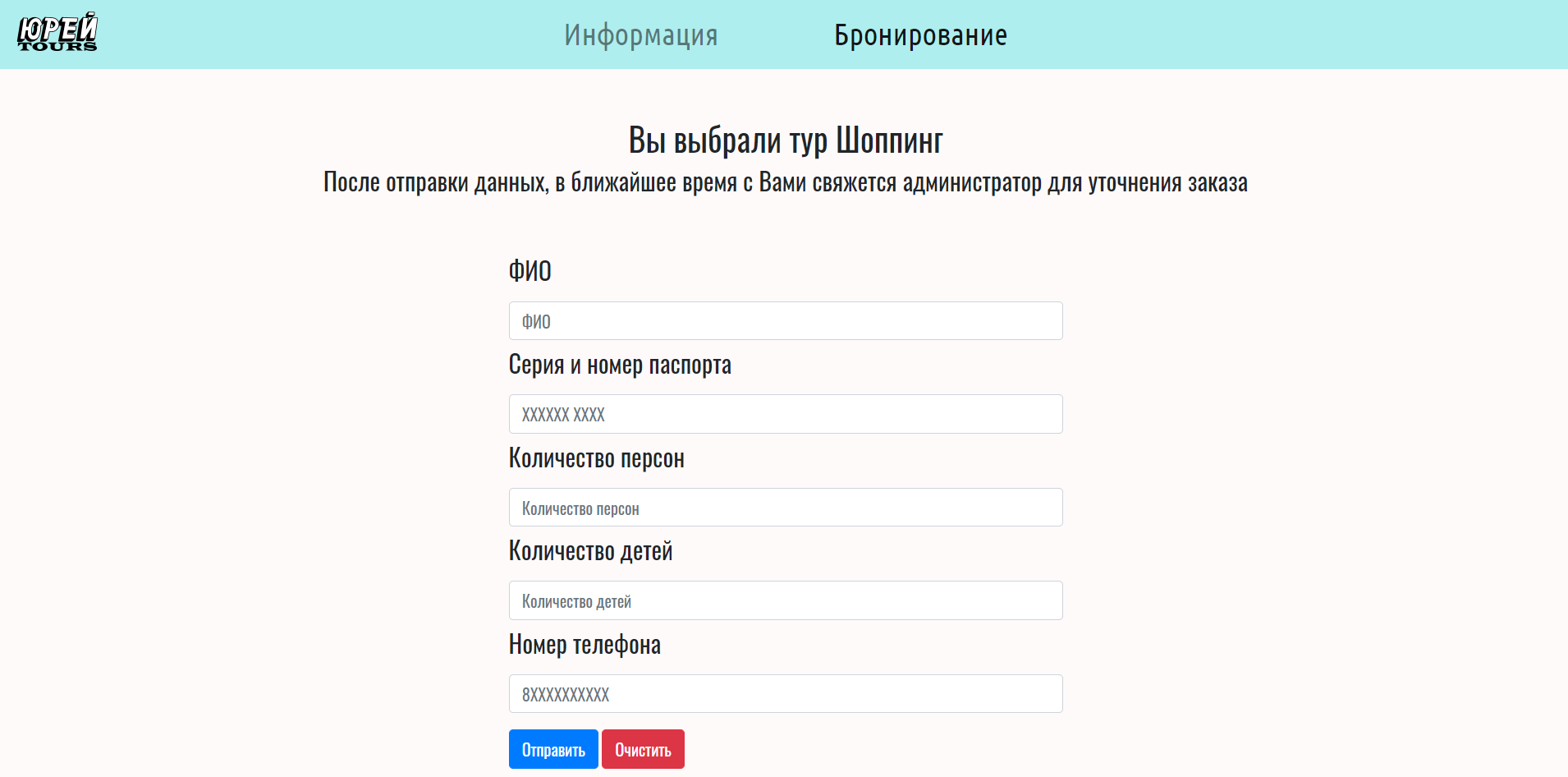


Рисунок А 17 – Страница Бронирование



Рисунок А 18 – Добавление клиента



Рисунок А 19 – Добавление тура



Рисунок А 20 – Изменение записи



Рисунок А 21 – Результат редактирования записи

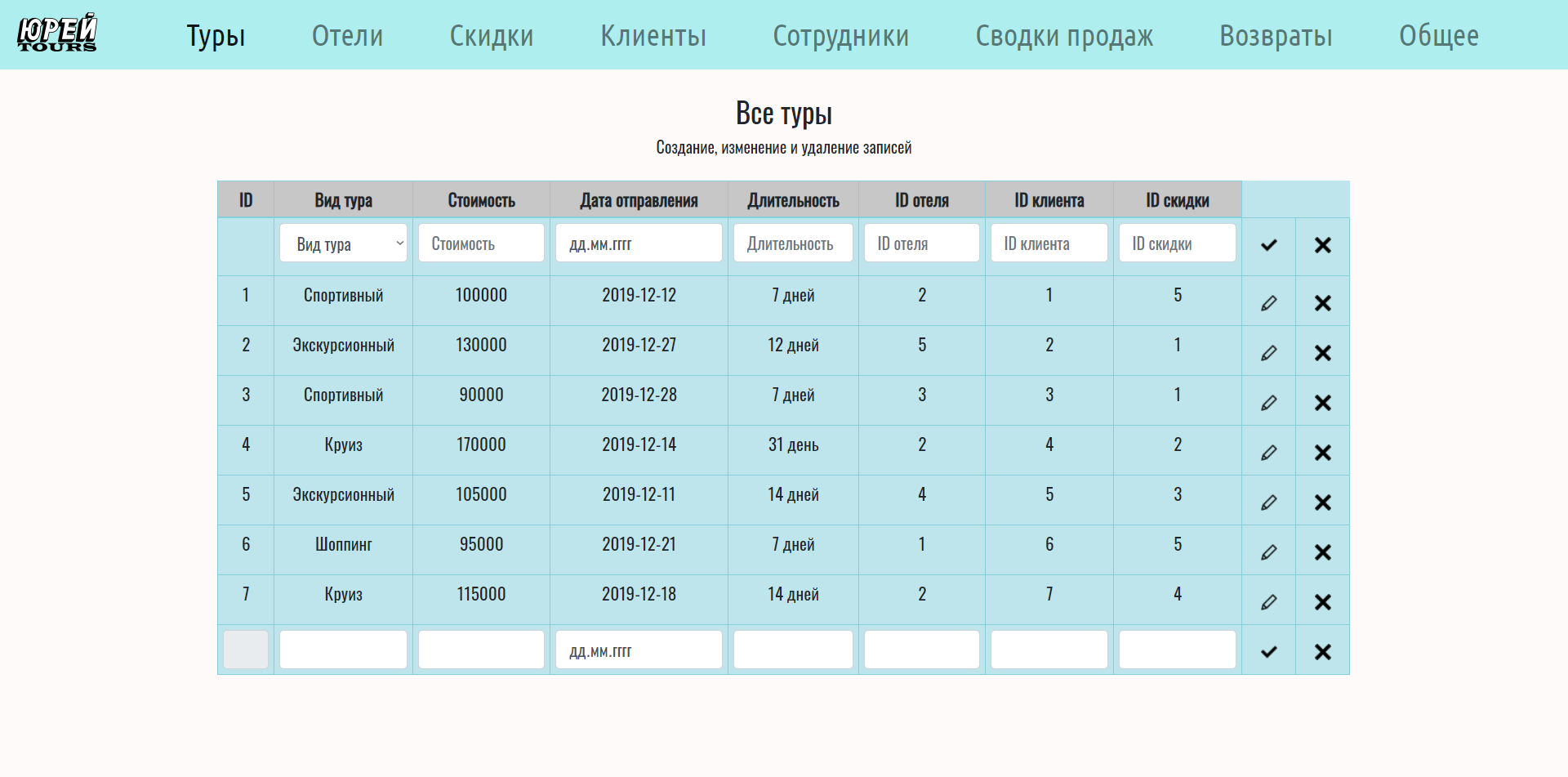


Рисунок А 22 – Результат удаления записи

**ПРИЛОЖЕНИЕ Б**

**SQL скрипт создания базы данных**

CREATE TABLE клиент (

id\_клиента int NOT NULL AUTO\_INCREMENT,

количество\_персон int NOT NULL,

количество\_детей int NOT NULL,

паспорт varchar(30) NOT NULL,

фио\_клиента varchar(30) NOT NULL,

телефон\_клиента varchar(30) NOT NULL,

PRIMARY KEY(id\_клиента) )

CREATE TABLE возврат (

id\_возврата int NOT NULL AUTO\_INCREMENT,

дата\_возврвта varchar(15) NOT NULL,

причина\_возврата varchar(30) NOT NULL,

сумма\_возврвта varchar(30) NOT NULL,

id\_тура int NOT NULL,

id\_клиента int NOT NULL,

id\_сотрудника int NOT NULL,

PRIMARY KEY(id\_возврата) )

ALTER TABLE возврат ADD FOREIGN KEY(id\_тура) REFERENCES тур(id\_тура) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE возврат ADD FOREIGN KEY(id\_клиента) REFERENCES клиент(id\_клиента) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE возврат ADD FOREIGN KEY(id\_сотрудника) REFERENCES сотрудник(id\_сотрудника) ON DELETE CASCADE ON UPDATE CASCADE;

CREATE TABLE отель (

id\_отеля int NOT NULL AUTO\_INCREMENT,

название\_отеля varchar(15) NOT NULL,

адрес\_отеля varchar(30) NOT NULL,

класс\_отеля varchar(15) NOT NULL,

PRIMARY KEY(id\_отеля) )

CREATE TABLE скидка (

id\_скидки int NOT NULL AUTO\_INCREMENT,

размер\_скидки varchar(10) NOT NULL,

PRIMARY KEY(id\_скидки) )

CREATE TABLE сводка\_продаж (

id\_сводки int NOT NULL AUTO\_INCREMENT,

количество\_продаж varchar(15) NOT NULL,

id\_сотрудника int NOT NULL,

PRIMARY KEY(id\_сводки) )

ALTER TABLE сводка\_продаж ADD FOREIGN KEY(id\_сотрудника) REFERENCES сотрудник(id\_сотрудника) ON DELETE CASCADE ON UPDATE CASCADE;

CREATE TABLE сотрудник (

id\_сотрудника int NOT NULL AUTO\_INCREMENT,

фио\_сотрудника varchar(35) NOT NULL,

телефон\_сотрудника int NOT NULL,

PRIMARY KEY(id\_сотрудника) )

CREATE TABLE `тур`(

`id\_тура` int NOT NULL AUTO\_INCREMENT,

`вид\_тура` VARCHAR(30) NOT NULL,

`стоимость\_тура` VARCHAR(15) NOT NULL,

`дата\_отправления` VARCHAR(15) NOT NULL,

`длительность\_тура` VARCHAR(15) NOT NULL,

`id\_отеля` int NOT NULL,

`id\_клиента int NOT NULL,

`id\_скидки` int NOT NULL,

PRIMARY KEY (`id\_тура`)

ALTER TABLE тур ADD FOREIGN KEY(id\_отеля) REFERENCES отель(id\_отеля) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE тур ADD FOREIGN KEY(id\_клиента) REFERENCES клиент(id\_клиента) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE тур ADD FOREIGN KEY(id\_скидки) REFERENCES скидка(id\_скидки) ON DELETE CASCADE ON UPDATE CASCADE;

CREATE TABLE клиент\_тур (

id\_клиент\_тур int NOT NULL AUTO\_INCREMENT,

id\_тура int NOT NULL,

id\_клиента int NOT NULL,

PRIMARY KEY(id\_клиент\_тур) )

ALTER TABLE клиент\_тур ADD FOREIGN KEY(id\_тура) REFERENCES тур(id\_тура) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE клиент\_тур ADD FOREIGN KEY(id\_клиента) REFERENCES клиент(id\_клиента) ON DELETE CASCADE ON UPDATE CASCADE;

**ПРИЛОЖЕНИЕ В**

**Код приложения**

clienttouraction.php

<?php

function add()

{

$tid = $\_POST['tid'];

$cid = $\_POST['cid'];

require 'dbconnect.php';

$sql = 'INSERT INTO `клиент\_тур`(`id\_тура`, `id\_клиента`) VALUES(:tid, :cid)';

$query = $pdo->prepare($sql);

$query->execute(['tid' => $tid, 'cid' => $cid]);

header('Location: /clienttour.php');

};

function ddelete()

{

require 'dbconnect.php';

$id = $\_POST['deleteid'];

$sql = 'DELETE FROM `клиент\_тур` WHERE `id\_клиент\_тур` = ?';

$query = $pdo->prepare($sql);

$query->execute([$id]);

header('Location: /clienttour.php');

};

function edit()

{

$tid = $\_POST['tid'];

$cid = $\_POST['cid'];

$id = $\_POST['t\_id'];

require 'dbconnect.php';

$sql = 'UPDATE `клиент\_тур` SET `id\_клиент\_тур`=:id, `id\_тура`=:tid, `id\_клиента`=:cid WHERE `id\_клиент\_тур` = :id';

$query = $pdo->prepare($sql);

$query->execute(['tid' => $tid, 'cid' => $cid, 'id' => $id]);

header('Location: /clienttour.php');

}

if ( isset ($\_POST['add']) )

{

add();

}

if ( isset ($\_POST['delete']) )

{

ddelete();

}

if ( isset ($\_POST['update']) )

{

edit();

}

?>

briefsaction.php

<?php

function add()

{

$com = $\_POST['com'];

$id\_worker = $\_POST['id\_worker'];

require 'dbconnect.php';

$sql = 'INSERT INTO `сводка\_продаж`(`количество\_продаж`, `id\_сотрудника`) VALUES(:com, :id\_worker)';

$query = $pdo->prepare($sql);

$query->execute(['com' => $com, 'id\_worker' => $id\_worker]);

header('Location: /briefs.php');

};

function ddelete()

{

require 'dbconnect.php';

$id = $\_POST['deleteid'];

$sql = 'DELETE FROM `сводка\_продаж` WHERE `id\_сводки` = ?';

$query = $pdo->prepare($sql);

$query->execute([$id]);

header('Location: /briefs.php');

};

function edit()

{

$com = $\_POST['com'];

$id\_worker = $\_POST['id\_worker'];

$id = $\_POST['t\_id'];

require 'dbconnect.php';

$sql = 'UPDATE `сводка\_продаж` SET `id\_сводки`=:id, `количество\_продаж`=:com, `id\_сотрудника`=:id\_worker WHERE `id\_сводки` = :id';

$query = $pdo->prepare($sql);

$query->execute(['com' => $com, 'id\_worker' => $id\_worker, 'id' => $id]);

header('Location: /briefs.php');

}

if ( isset ($\_POST['add']) )

{

add();

}

if ( isset ($\_POST['delete']) )

{

ddelete();

}

if ( isset ($\_POST['update']) )

{

edit();

}

?>

clientsaction.php

<?php

function add()

{

$person = $\_POST['person'];

$kids = $\_POST['kids'];

$passport = $\_POST['passport'];

$name = $\_POST['name'];

$phone = $\_POST['phone'];

require 'dbconnect.php';

$sql = 'INSERT INTO `клиент`(`количество\_персон`, `количество\_детей`, `паспорт`, `фио\_клиента`, `телефон\_клиента`) VALUES(:person, :kids, :passport, :name, :phone)';

$query = $pdo->prepare($sql);

$query->execute(['person' => $person, 'kids' => $kids, 'passport' => $passport,

'name' => $name, 'phone' => $phone]);

header('Location: /clients.php');

};

function ddelete()

{

require 'dbconnect.php';

$id = $\_POST['deleteid'];

$sql = 'DELETE FROM `клиент` WHERE `id\_клиента` = ?';

$query = $pdo->prepare($sql);

$query->execute([$id]);

header('Location: /clients.php');

};

function edit()

{

$person = $\_POST['person'];

$kids = $\_POST['kids'];

$passport = $\_POST['passport'];

$name = $\_POST['name'];

$phone = $\_POST['phone'];

$id = $\_POST['t\_id'];

require 'dbconnect.php';

$sql = 'UPDATE `клиент` SET `id\_клиента`=:id, `количество\_персон`=:person, `количество\_детей`=:kids, `паспорт`=:passport, `фио\_клиента`=:name, `телефон\_клиента`=:phone WHERE `id\_клиента` = :id';

$query = $pdo->prepare($sql);

$query->execute(['person' => $person, 'kids' => $kids, 'passport' => $passport,

'name' => $name, 'phone' => $phone, 'id' => $id]);

header('Location: /clients.php');

}

if ( isset ($\_POST['add']) )

{

add();

}

if ( isset ($\_POST['delete']) )

{

ddelete();

}

if ( isset ($\_POST['update']) )

{

edit();

}

?>

hotelsaction.php

<?php

function add()

{

$name = $\_POST['name'];

$adress = $\_POST['adress'];

$class = $\_POST['class'];

require 'dbconnect.php';

$sql = 'INSERT INTO `отель`(`название\_отеля`, `адрес\_отеля`, `класс\_отеля`) VALUES(:name, :adress, :class)';

$query = $pdo->prepare($sql);

$query->execute(['name' => $name, 'adress' => $adress, 'class' => $class]);

header('Location: /hotels.php');

};

function ddelete()

{

require 'dbconnect.php';

$id = $\_POST['deleteid'];

$sql = 'DELETE FROM `отель` WHERE `id\_отеля` = ?';

$query = $pdo->prepare($sql);

$query->execute([$id]);

header('Location: /hotels.php');

};

function edit()

{

$name = $\_POST['name'];

$adress = $\_POST['adress'];

$class = $\_POST['class'];

$id = $\_POST['t\_id'];

require 'dbconnect.php';

$sql = 'UPDATE `отель` SET `id\_отеля`=:id, `название\_отеля`=:name, `адрес\_отеля`=:adress, `класс\_отеля`=:class WHERE `id\_отеля` = :id';

$query = $pdo->prepare($sql);

$query->execute(['name' => $name, 'adress' => $adress, 'class' => $class, 'id' => $id]);

header('Location: /hotels.php');

}

if ( isset ($\_POST['add']) )

{

add();

}

if ( isset ($\_POST['delete']) )

{

ddelete();

}

if ( isset ($\_POST['update']) )

{

edit();

}

?>

reservationaction.php

<?php

$person = $\_POST['person'];

$kids = $\_POST['kids'];

$passport = $\_POST['passport'];

$name = $\_POST['name'];

$phone = $\_POST['phone'];

require 'dbconnect.php';

$sql = 'INSERT INTO `клиент`(`количество\_персон`, `количество\_детей`, `паспорт`, `фио\_клиента`, `телефон\_клиента`) VALUES(:person, :kids, :passport, :name, :phone)';

$query = $pdo->prepare($sql);

$query->execute(['person' => $person, 'kids' => $kids, 'passport' => $passport,

'name' => $name, 'phone' => $phone]);

header('Location: /info.php');

?>

returnsaction.php

<?php

function add()

{

$ddate = $\_POST['date'];

$con = $\_POST['con'];

$price = $\_POST['price'];

$tourid = $\_POST['tourid'];

$clientid = $\_POST['clientid'];

$workerid = $\_POST['workerid'];

require 'dbconnect.php';

$sql = 'INSERT INTO `возврат`(`дата\_возврата`, `причина\_возврата`, `сумма\_возврата`, `id\_тура`, `id\_клиента`, `id\_сотрудника`) VALUES(:ddate, :con, :price, :tourid, :clientid, :workerid)';

$query = $pdo->prepare($sql);

$query->execute(['ddate' => $ddate, 'con' => $con, 'price' => $price,

'tourid' => $tourid, 'clientid' => $clientid, 'workerid' => $workerid]);

header('Location: /returns.php');

};

function ddelete()

{

require 'dbconnect.php';

$id = $\_POST['deleteid'];

$sql = 'DELETE FROM `возврат` WHERE `id\_возврата` = ?';

$query = $pdo->prepare($sql);

$query->execute([$id]);

header('Location: /returns.php');

};

function edit()

{

$ddate = $\_POST['date'];

$con = $\_POST['con'];

$price = $\_POST['price'];

$tourid = $\_POST['tourid'];

$clientid = $\_POST['clientid'];

$workerid = $\_POST['workerid'];

$id = $\_POST['t\_id'];

require 'dbconnect.php';

$sql = 'UPDATE `возврат` SET `id\_возврата`=:id, `дата\_возврата`=:kind, `причина\_возврата`=:price, `сумма\_возврата`=:ddate, `id\_тура`=:hotelid, `id\_клиента`=:clientid, `id\_сотрудника`=:saleid WHERE `id\_возврата` = :id';

$query = $pdo->prepare($sql);

$query->execute(['ddate' => $ddate, 'con' => $con, 'price' => $price,

'tourid' => $tourid, 'clientid' => $clientid, 'workerid' => $workerid,

'id' => $id]);

header('Location: /returns.php');

}

if ( isset ($\_POST['add']) )

{

add();

}

if ( isset ($\_POST['delete']) )

{

ddelete();

}

if ( isset ($\_POST['update']) )

{

edit();

}

?>

salesaction.php

<?php

function add()

{

$sale = $\_POST['sale'];

require 'dbconnect.php';

$sql = 'INSERT INTO `скидка`(`размер\_скидки`) VALUES(:sale)';

$query = $pdo->prepare($sql);

$query->execute(['sale' => $sale]);

header('Location: /sales.php');

};

function ddelete()

{

require 'dbconnect.php';

$id = $\_POST['deleteid'];

$sql = 'DELETE FROM `скидка` WHERE `id\_скидки` = ?';

$query = $pdo->prepare($sql);

$query->execute([$id]);

header('Location: /sales.php');

};

function edit()

{

$sale = $\_POST['sale'];

$id = $\_POST['t\_id'];

require 'dbconnect.php';

$sql = 'UPDATE `скидка` SET `id\_скидки`=:id, `размер\_скидки`=:sale WHERE `id\_скидки` = :id';

$query = $pdo->prepare($sql);

$query->execute(['sale' => $sale, 'id' => $id]);

header('Location: /sales.php');

}

if ( isset ($\_POST['add']) )

{

add();

}

if ( isset ($\_POST['delete']) )

{

ddelete();

}

if ( isset ($\_POST['update']) )

{

edit();

}

?>

toursaction.php

<?php

function add()

{

$kind = $\_POST['kind'];

$price = $\_POST['price'];

$ddate = $\_POST['date'];

$length = $\_POST['length'];

$hotelid = $\_POST['hotelid'];

$clientid = $\_POST['clientid'];

$saleid = $\_POST['saleid'];

require 'dbconnect.php';

$sql = 'INSERT INTO `тур`(`вид\_тура`, `стоимость\_тура`, `дата\_отправления`, `длительность\_тура`, `id\_отеля`, `id\_клиента`, `id\_скидки`) VALUES(:kind, :price, :ddate, :length, :hotelid, :clientid, :saleid)';

$query = $pdo->prepare($sql);

$query->execute(['kind' => $kind, 'price' => $price, 'ddate' => $ddate,

'length' => $length, 'hotelid' => $hotelid, 'clientid' => $clientid,

'saleid' => $saleid]);

header('Location: /tours.php');

};

function ddelete()

{

require 'dbconnect.php';

$id = $\_POST['deleteid'];

$sql = 'DELETE FROM `тур` WHERE `id\_тура` = ?';

$query = $pdo->prepare($sql);

$query->execute([$id]);

header('Location: /tours.php');

};

function edit()

{

$kind = $\_POST['kind'];

$price = $\_POST['price'];

$ddate = $\_POST['date'];

$length = $\_POST['length'];

$hotelid = $\_POST['hotelid'];

$clientid = $\_POST['clientid'];

$saleid = $\_POST['saleid'];

$id = $\_POST['t\_id'];

require 'dbconnect.php';

$sql = 'UPDATE `тур` SET `id\_тура`=:id, `вид\_тура`=:kind, `стоимость\_тура`=:price, `дата\_отправления`=:ddate, `длительность\_тура`=:length, `id\_отеля`=:hotelid, `id\_клиента`=:clientid, `id\_скидки`=:saleid WHERE `id\_тура` = :id';

$query = $pdo->prepare($sql);

$query->execute(['kind' => $kind, 'price' => $price, 'ddate' => $ddate,

'length' => $length, 'hotelid' => $hotelid, 'clientid' => $clientid,

'saleid' => $saleid, 'id' => $id]);

header('Location: /tours.php');

}

if ( isset ($\_POST['add']) )

{

add();

}

if ( isset ($\_POST['delete']) )

{

ddelete();

}

if ( isset ($\_POST['update']) )

{

edit();

}

?>

workersaction.php

<?php

function add()

{

$name = $\_POST['name'];

$phone = $\_POST['phone'];

require 'dbconnect.php';

$sql = 'INSERT INTO `сотрудник`(`фио\_сотрудника`, `телефон\_сотрудника`) VALUES(:name, :phone)';

$query = $pdo->prepare($sql);

$query->execute(['name' => $name, 'phone' => $phone]);

header('Location: /workers.php');

};

function ddelete()

{

require 'dbconnect.php';

$id = $\_POST['deleteid'];

$sql = 'DELETE FROM `сотрудник` WHERE `id\_сотрудника` = ?';

$query = $pdo->prepare($sql);

$query->execute([$id]);

header('Location: /workers.php');

};

function edit()

{

$name = $\_POST['name'];

$phone = $\_POST['phone'];

$id = $\_POST['t\_id'];

require 'dbconnect.php';

$sql = 'UPDATE `сотрудник` SET `id\_сотрудника`=:id, `фио\_сотрудника`=:name, `телефон\_сотрудника`=:phone WHERE `id\_сотрудника` = :id';

$query = $pdo->prepare($sql);

$query->execute(['name' => $name, 'phone' => $phone, 'id' => $id]);

header('Location: /workers.php');

}

if ( isset ($\_POST['add']) )

{

add();

}

if ( isset ($\_POST['delete']) )

{

ddelete();

}

if ( isset ($\_POST['update']) )

{

edit();

}

?>