**1.1.3 Описание связей между объектами**

1. В одной гостинице имеется несколько номеров, также может и не оказаться номеров в ней. Из этого возникает следующая связь:

гостиница – номер (1:n);

2. Гостинице выдана пара: пароль и логин, поэтому возникает следующая связь:

гостиница – пароль и логин гостиницы (1:1);

3. Гостиница и номер предоставляет услуги, которых может быть не-сколько или не быть вообще. Образуются следующие связи через сущность «Оказание услуг гостиницей (номеров)»:

гостиница – оказание услуг гостиницей – услуги гостиницы (n:n);

номер – оказание услуг гостиницей – услуги номера (n:n);

4. Клиенты могут оставить отзывы о гостинице или не отправить ре-цензию. Отсюда возникает связь:

гостиница – отзыв (1:n);

5. Клиенты могут оформить одну бронь или несколько для одного определенного номера в гостинице и оплатить ее один раз:

бронирование – клиент (n:1);

бронирование – номер (1:1);

бронирование – платеж (1:1).

**1.1.4 Представление сущностей с атрибутами и доменами**

**Объект «Гостиница»:**

Гостиница = {

Код: Число (Первичный ключ)

Название: Строка(50)

Адрес: Строка(30)

Телефон: Строка(15)

Email: Строка(30)

Имя директора: Строка(30)

Категория: Строка(30)

Тип: Строка(30)}

**Объект «Услуги гостиницы»:**

Услуги гостиницы = {

Код: Число (Первичный ключ)

Услуги: Строка(40)}

**Объект «Оказание услуг гостиницей»:**

Оказание услуг гостиницей = {

Гостиница: Значение «Код» отношения «Гостиница» (Внешний ключ)

Услуги: Значение «Код» отношения «Услуги гостиницы» (Внешний ключ)}

**Объект «Номер»:**

Номер = {

Код: Число (Первичный ключ)

Цена: Денежное значение

Статус: Логическое значение

Расположение: Строка(30)

Уровень: Строка(30)

Тип: Строка(30)

Гостиница: Значение «Код» отношения «Гостиница» (Внешний ключ)}

**Объект «Услуги номера»:**

Услуги номера = {

Код: Число (Первичный ключ)

Услуги: Строка(40)}

**Объект «Оказание услуг номером»:**

Оказание услуг номером = {

Номер: Значение «Код» отношения «Номер» (Внешний ключ)

Услуги: Значение «Код» отношения «Услуги номера» (Внешний ключ)}

**Объект «Отзыв»:**

Отзыв = {

Отзыв: Строка(100) (Первичный ключ)

Имя: Строка(30) (Первичный ключ)

Дата: Дата и время (Первичный ключ)

Гостиница: Значение «Код» отношения «Гостиница» (Внешний ключ)}

**Объект «Пароль и логин гостиницы»:**

Пароль и логин гостиницы = {

Логин: Строка(20) (Первичный ключ)

Пароль: Строка(20) (Первичный ключ)

Гостиница: Значение «Код» отношения «Гостиница» (Внешний ключ)}

**Объект «Клиент»:**

Клиент = {

Код: Число (Первичный ключ)

Имя: Строка(20)

Фамилия: Строка(20)

Отчество: Строка(20)

Телефон: Строка(15)

Гражданство: Строка(30)

Банковские данные: Строка(30)

Email: Строка(30)}

**Объект «Бронирование»:**

Бронирование = {

Код: Число (Первичный ключ)

Дата въезда: Дата и время

Дата выезда: Дата и время

Номер: Значение «Код» отношения «Номер» (Внешний ключ)

Клиент: Значение «Код» отношения «Клиент» (Внешний ключ)}

**Объект «Платеж»:**

Платеж = {

Код: Число (Первичный ключ)

Счет: Денежное значение

Дата: Дата и время

Тип платежа: Строка(30)

Бронирование: Значение «Код» отношения «Бронирование» (Внешний ключ)}

## **Построение концептуального уровня БД**

## **1.2.1 Описание функциональных зависимостей в предметной области**

Отношение **Гостиница** = {код, название, адрес, телефон, email, имя директора, категория, тип}

{код}  {название}

{код} {адрес, телефон}

{код}  {email}

{код} {имя директора}

{код} {категория, тип}

Отношение **Услуги гостиницы** = {код, услуги}

{код}  {услуги}

Отношение **Номер** = {код, цена, статус, расположение, уровень, тип, гостиница}

{код}  {уровень, тип, цена}

{код}  {расположение}

{код}  {статус}

{код}  {гостиница}

Отношение **Услуги номера** = {код, услуги}

{код}  {услуги}

Отношение **Отзыв** = {код, отзыв, имя, дата, гостиница}

{гостиница}  {имя, отзыв, дата}

Отношение **Пароль и логин гостиницы** = {логин, пароль, гостиница}

{гостиница}  {логин, пароль}

Отношение **Клиент** = {код, имя, фамилия, отчество, телефон, гражданство, банковские данные, email}

{код}  {имя, фамилия, отчество, телефон, гражданство, банковские данные, email}

Отношение **Бронирование** = {код, дата въезда, дата выезда, номер, клиент}

{код}  {дата въезда, дата выезда, номер, клиент }

Отношение **Платеж** = {код, счет, дата, тип платежа, бронирование}

{код}  {счет, дата, тип платежа, бронирование}

**1.2.2 Нормализация отношений**

**1.2.2.1 Первая нормальная форма**

Отношение находится в первой нормальной форме (1НФ), если каждый его атрибут атомарен, т.е. на пересечении строки и столбца содержится только одно значение.

Рассмотрим базовые отношения базы данных бронирования гостиниц:

Отношение **Гостиница**(1) = {код, название, адрес, телефон, email, имя директора, категория, тип};

Отношение **Услуги гостиницы**(1)= {код, услуги};

Отношение **Номер**(1) = {код, цена, статус, расположение, уровень, тип, гостиница}

Отношение **Услуги номера**(1)= {код, услуги};

Отношение **Отзыв**(1) = {отзыв, имя, дата, гостиница};

Отношение **Пароль и логин гостиницы**(1) = {логин, пароль, гостиница};

Отношение **Клиент**(1) = {код, имя, фамилия, отчество, телефон, гражданство, банковские данные, email};

Отношение **Бронирование**(1)= {код, дата въезда, дата выезда, номер, клиент};

Отношение **Платеж**(1) = {код, счет, дата, тип платежа, бронирование}.

Все базовые отношения находятся в 1НФ; производные отношения «Оказание услуг гостиницей» и «Оказание услуг номером» введены для того, чтобы нормализовать отношения «Гостиница» и «Номер». Эти отношения

были декомпозированы, а поля «Услуги гостиницы» и «Услуги номера» были вынесены в отдельные отношения с введением ключей, которые представляют собой списки значений.

**3.3 Результаты реализации клиентского интерфейса**

Представлены реализованные интерфейсы для выделенных конечных пользователей в галаве 1.1.1. Для каждого пользователя выдана информация, соответствующая его представлению.

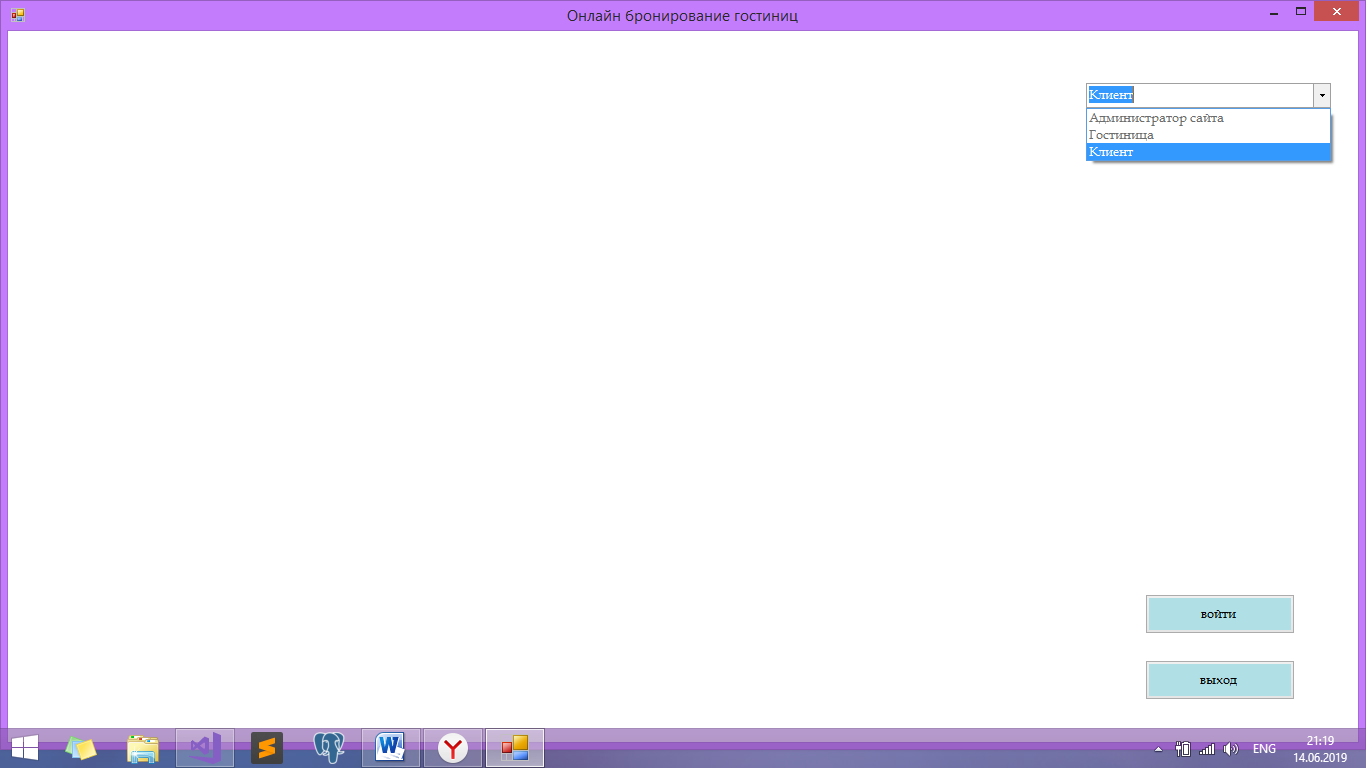


Рисунок 1. Скриншот интерфейса приложения при выборе пользователя «клиент»

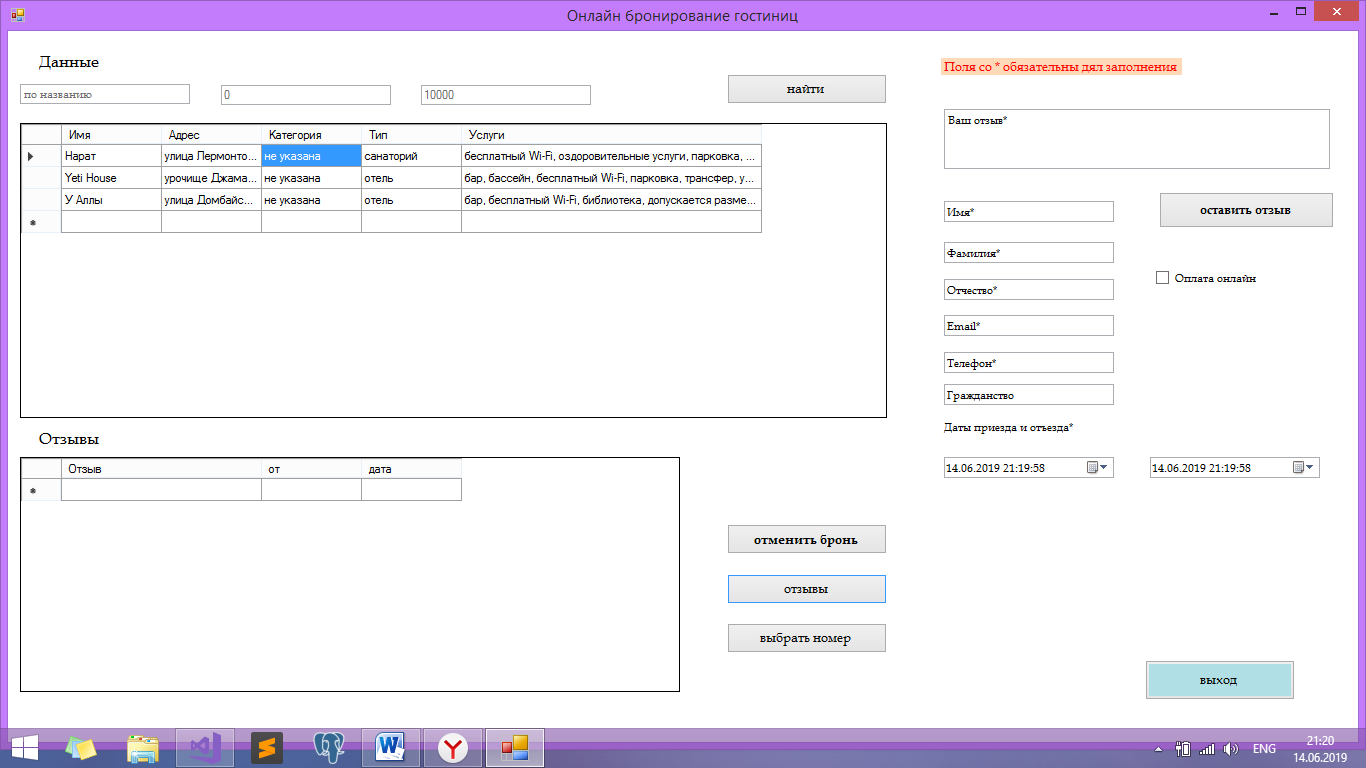


Рисунок 2. Диалоговое окно пользователя «клиент»

На стороне «клиента» доступны список гостиниц с предоставляемыми услугаи с возможностью поиска по названию и ценам номером, отзывы оставленные другими клиентами, перечень свободных номеров гостиницы и их услугах. Полномочия клиента: забронировать номер с возможностью оплаты онлайн или отмены брони, также написание отзыва о гостинице (Рисунок 2).

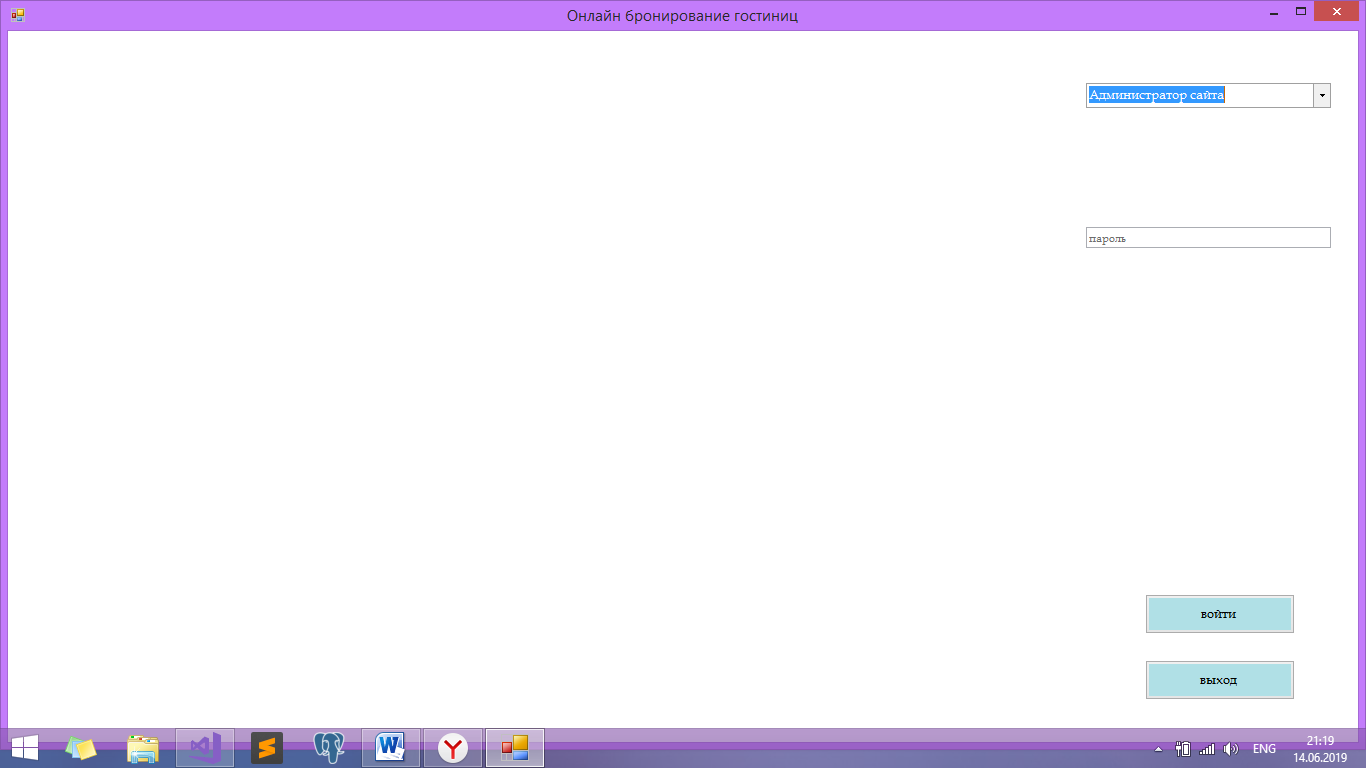


Рисунок 3. Скриншот интерфейса приложения при выборе пользователя «администратор»

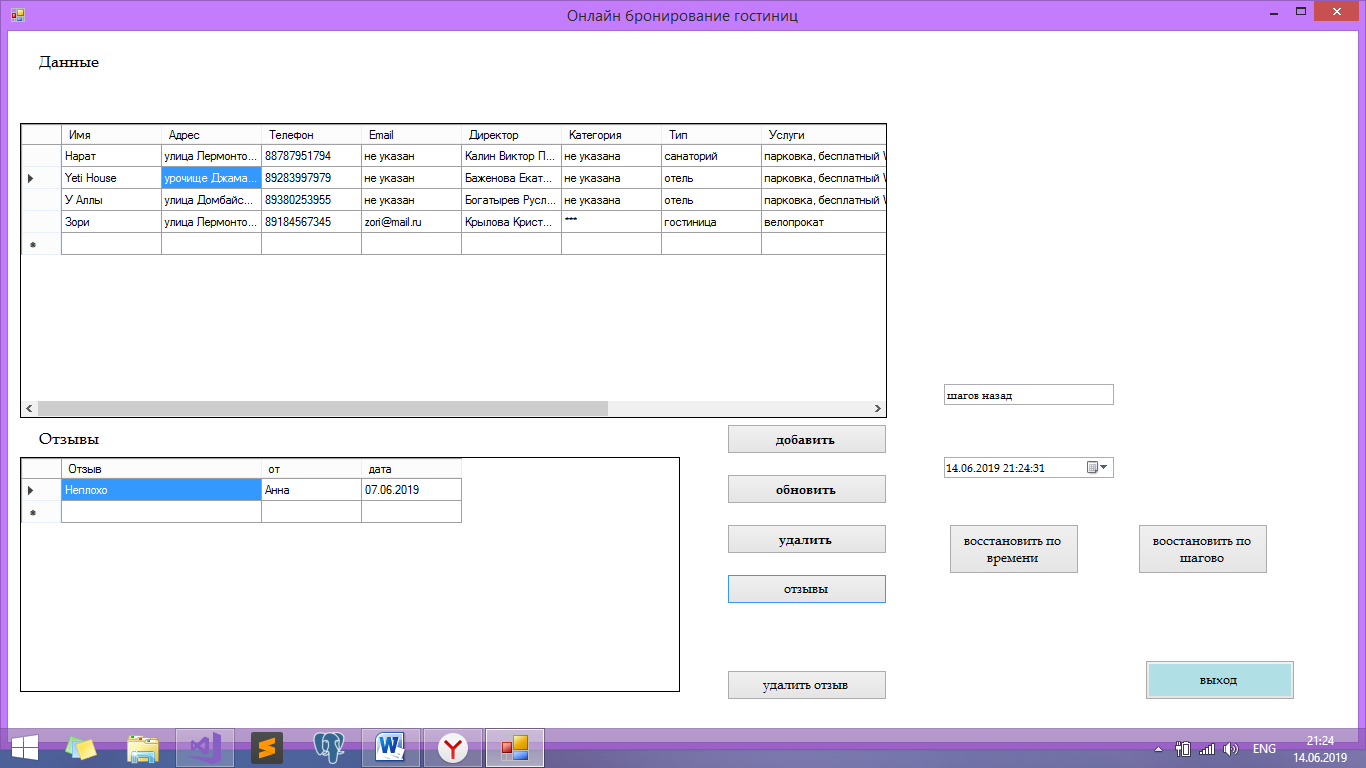


Рисунок 4. Диалоговое окно пользователя «администратор»

Администратору сайта по бронированию номеров доступна полная информация о гостиницах с паролем и логином, также об отзвах. Администратор вправе вносить новые гостиницы в список, убрать из списка или обновить сведения о них. Также только на стороне администратора существует возможность уаление отзывов (Рисунок 4).

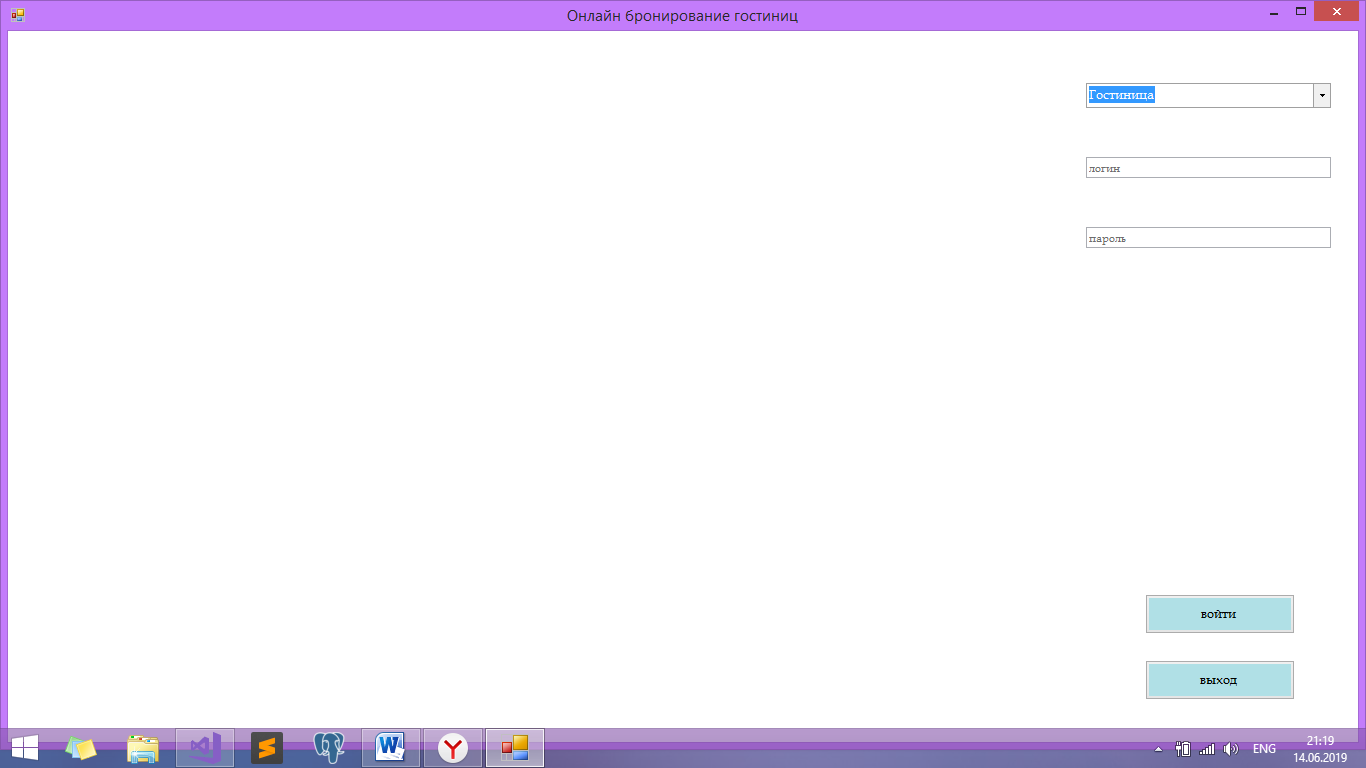


Рисунок 5. Скриншот интерфейса приложения при выборе пользователя «гостиница»

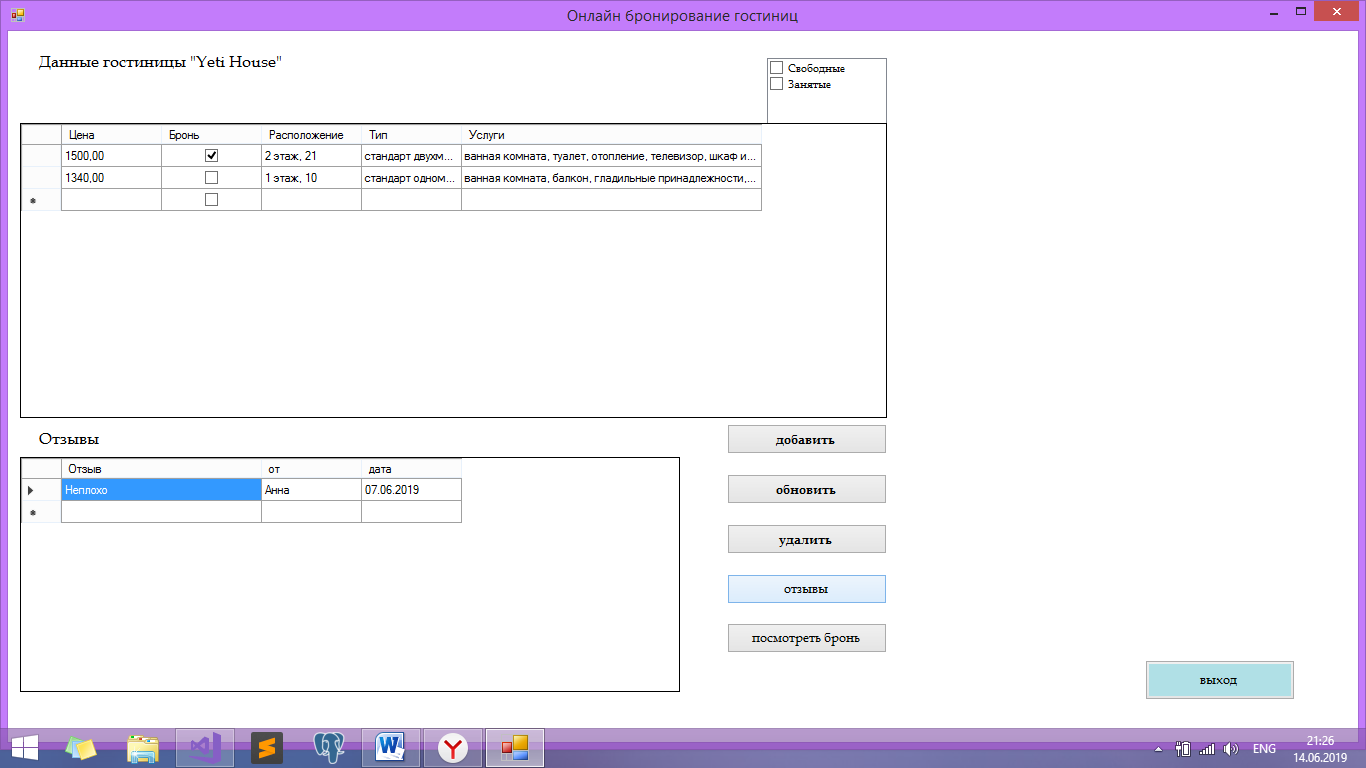


Рисунок 6. Диалоговое окно пользователя «гостиница «Yeti House». Две таблицы с перечнем всех номеров и отзывов о гостинице соотвественно

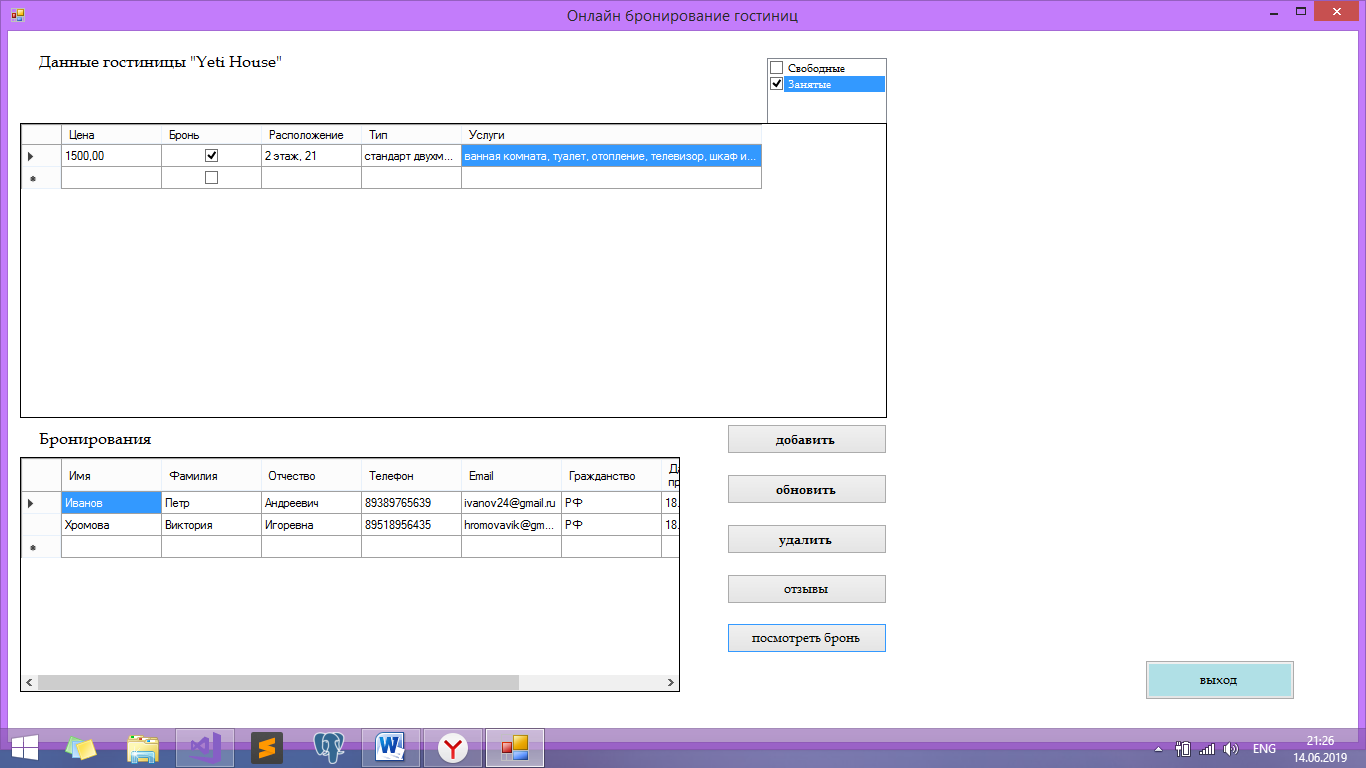


Рисунок 7. Диалоговое окно пользователя «гостиница «Yeti House». Две таблицы с перечнем занятых номеров и информации о бронировании номера соотвественно

Гостинице доступна информация обо всех своих номерах, бронированиях и отзывах при ввдении логина и пароля. На стороне гостиницы введены возможности внесения новых номеров, удаления или обновления сведений о них (Рисунок 6,7).

ПРИЛОЖЕНИЕ А

ER – диаграмма

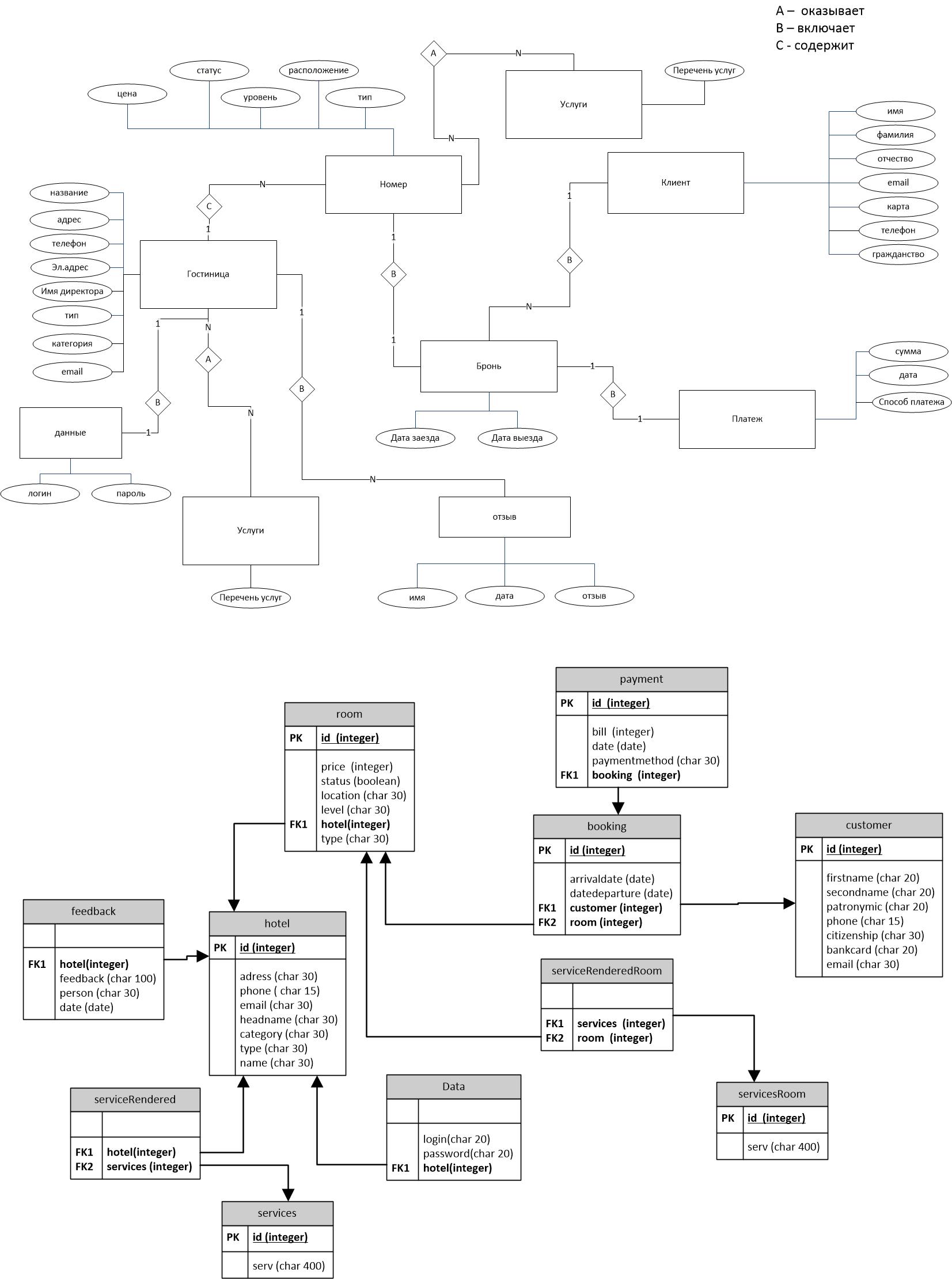


Рисунок А.1. ER – диаграмма БД услуг поиска и бронирования гостиничных номеров

Даталогическая схема

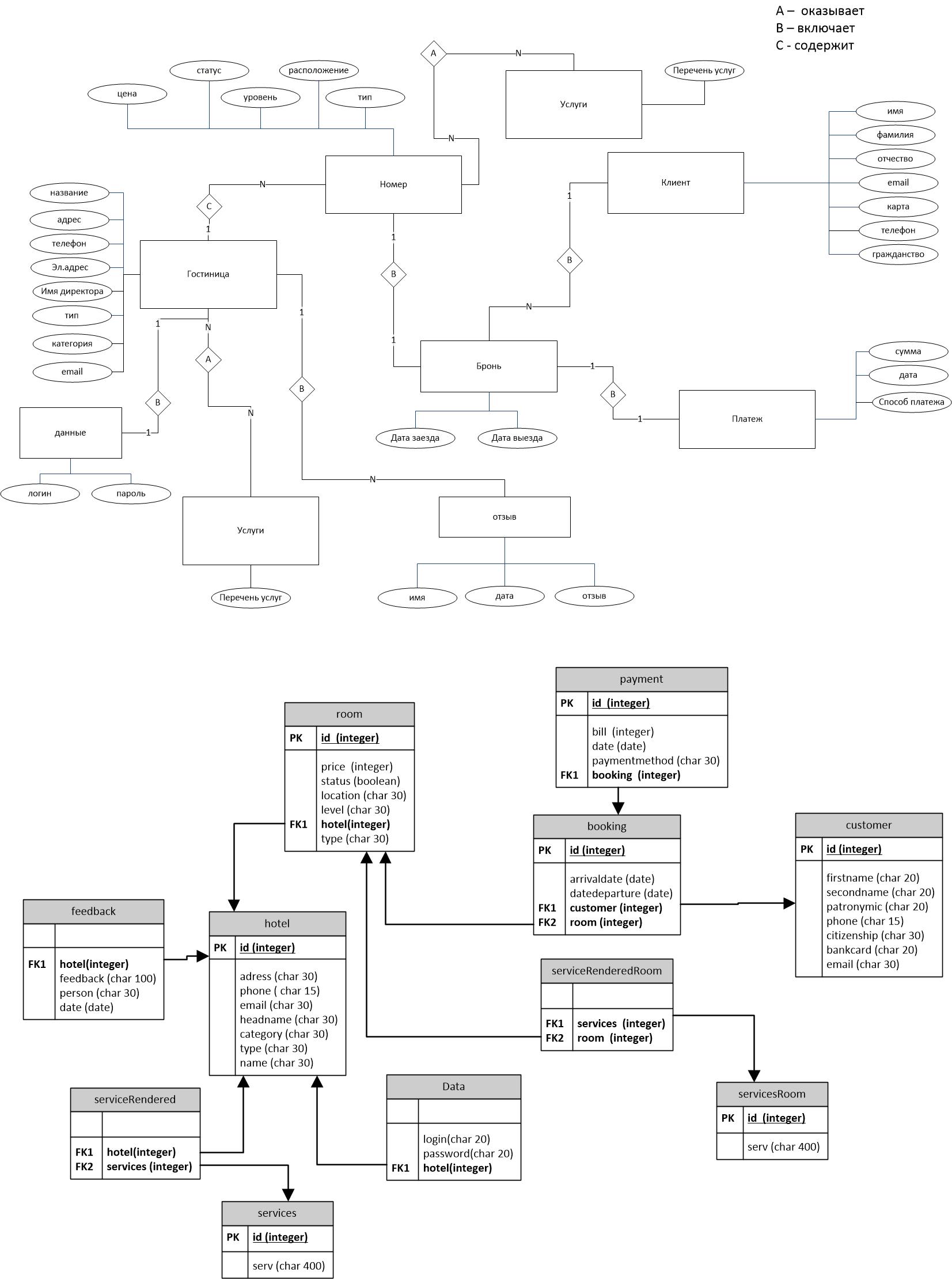


Рисунок А.2. Даталогическая схема БД услуг поиска и бронирования гостиничных номеров

ПРИЛОЖЕНИЕ Б

Листинг Б.1 - Код базы данных в PostgreSQL

CREATE TABLE if not exists hotel( –- отношение «Гостиница»

id\_hotel serial PRIMARY KEY,

name\_hotel character varying(50) NOT NULL,

adress character varying(50) NOT NULL,

phone\_hotel character varying(20) NOT NULL,

email\_hotel character varying(30) DEFAULT 'не указан',

headname character varying(30) NOT NULL,

category character varying(30) DEFAULT 'не указана',

type\_hotel character varying(30) DEFAULT 'не указан');

CREATE TABLE if not exists feedback( –- отношение «Отзыв»

hotel integer not null,

text\_feedback character varying(100) not null,

person character varying(30) not null,

date\_feedback date not null,

FOREIGN KEY (hotel) REFERENCES hotel (id\_hotel) ON DELETE CASCADE);

CREATE TABLE if not exists service\_hotel( –- отношение «Услуги гостиницы»

id\_service\_hotel serial PRIMARY KEY,

services\_hotel character varying(40));

CREATE TABLE if not exists service\_rendered\_hotel( /\* отношение «Оказание услуг гостиницей»\*/

service\_hotel integer,

hotel integer,

FOREIGN KEY (hotel) REFERENCES hotel (id\_hotel) ON DELETE CASCADE,

FOREIGN KEY (service\_hotel) REFERENCES service\_hotel (id\_service\_hotel) ON DELETE CASCADE);

CREATE TABLE if not exists room( –- отношение «Номер»

id\_room serial PRIMARY KEY,

price money not null,

status boolean not null,

location character varying(30) NOT NULL,

level character varying(30) DEFAULT 'не указан',

hotel integer,

FOREIGN KEY (hotel) REFERENCES hotel (id\_hotel) ON DELETE CASCADE);

CREATE TABLE if not exists service\_room( –- отношение «Услуги номера»

id\_service\_room serial PRIMARY KEY,

services\_room character varying(40));

CREATE TABLE if not exists service\_rendered\_room( /\* отношение «Оказание услуг номером»\*/

service\_room integer,

room integer,

FOREIGN KEY (room) REFERENCES room (id\_room) ON DELETE CASCADE,

FOREIGN KEY (service\_room) REFERENCES service\_room (id\_service\_room) ON DELETE CASCADE);

CREATE TABLE if not exists log\_in( –- отношение «Логин и пароль гостиницы»

hotel integer not null,

login\_hotel character varying(20) not null,

password\_hotel character varying(20) not null,

FOREIGN KEY (hotel) REFERENCES hotel (id\_hotel) ON DELETE CASCADE);

CREATE TABLE if not exists customer( –- отношение «Клиент»

id\_customer serial PRIMARY KEY,

firstname character varying(20) not null,

lastname character varying(20) not null,

patronymic character varying(20) DEFAULT 'не указан',

phone\_customer character varying(20) not null,

email\_customer character varying(30) not null,

citizenship character varying(30),

bankcard character varying(30));

CREATE TABLE if not exists booking( –- отношение «Бронирование»

id\_booking serial PRIMARY KEY,

arrivaldate date not null,

datedeparture date not null,

customer integer,

room integer,

FOREIGN KEY (room) REFERENCES room (id\_room) ON DELETE CASCADE,

FOREIGN KEY (customer) REFERENCES customer (id\_customer) ON DELETE CASCADE);

CREATE TABLE if not exists payment( –- отношение «Платеж»

id\_payment serial PRIMARY KEY,

paymentmethod character varying(30),

bill money not null,

date\_payment date not null,

booking integer,

FOREIGN KEY (booking) REFERENCES booking (id\_booking) ON DELETE CASCADE);

CREATE TABLE if not exists temper( –- отношение «Восстановление»

Id SERIAL PRIMARY KEY,

tim timestamp,

tablename CHARACTER VARYING(30),

operation CHARACTER VARYING(6),

idtb integer,

name character varying(50),

adress character varying(50),

phone character varying(20),

email character varying(30),

headname character varying(30),

category character varying(30),

type character varying(30),

services character varying(40),

refer integer,

price money,

status boolean,

location character varying(30),

level character varying(30));

/\* функция для сохранений старых записей в отношение «Восстановление» при изменениях в таблицах: «Гостиница», «Номер»,«Услуги гостиницы», «Услуги номера», «Оказание услуг гостиницей»,«Оказание услуг номером» \*/

CREATE OR REPLACE FUNCTION process\_with\_tables()

RETURNS TRIGGER AS

$select\_to$

BEGIN

if ( TG\_TABLE\_NAME = 'hotel' ) THEN

begin

if ( TG\_OP = 'INSERT' ) THEN

INSERT INTO temper(tim, tablename, operation, idtb, name, adress,phone,email,headname,category, type )

VALUES (now(),TG\_TABLE\_NAME,TG\_OP, New.id\_hotel,New.name\_hotel,New.adress,New.phone\_hotel,New.email\_hotel,New.headname,New.category,New.type\_hotel);

end if;

if ( TG\_OP = 'UPDATE' ) THEN

INSERT INTO temper(tim, tablename, operation, idtb, name, adress,phone,email,headname,category, type )

VALUES (now(),TG\_TABLE\_NAME,TG\_OP, Old.id\_hotel,Old.name\_hotel,Old.adress,Old.phone\_hotel,Old.email\_hotel,Old.headname,Old.category,Old.type\_hotel);

end if;

if ( TG\_OP = 'DELETE' ) THEN

INSERT INTO temper(tim, tablename, operation, idtb, name, adress,phone,email,headname,category, type )

VALUES (now(),TG\_TABLE\_NAME,TG\_OP, Old.id\_hotel,Old.name\_hotel,Old.adress,Old.phone\_hotel,Old.email\_hotel,Old.headname,Old.category,Old.type\_hotel);

end if;

end;

end if;

if ( TG\_TABLE\_NAME = 'service\_hotel' ) THEN

begin

if ( TG\_OP = 'INSERT' ) THEN

INSERT INTO temper(tim, tablename, operation,idtb,services)

VALUES (now(),TG\_TABLE\_NAME,TG\_OP, New.id\_service\_hotel,New.services\_hotel);

end if;

if ( TG\_OP = 'UPDATE' ) THEN

INSERT INTO temper(tim, tablename, operation,idtb, services)

VALUES (now(),TG\_TABLE\_NAME,TG\_OP,Old.id\_service\_hotel,Old.services\_hotel);

end if;

if ( TG\_OP = 'DELETE' ) THEN

INSERT INTO temper(tim, tablename, operation,idtb, services)

VALUES (now(),TG\_TABLE\_NAME,TG\_OP,Old.id\_service\_hotel,Old.services\_hotel);

end if;

end;

end if;

if ( TG\_TABLE\_NAME = 'service\_rendered\_hotel' ) THEN

begin

if ( TG\_OP = 'INSERT' ) THEN

INSERT INTO temper(tim, tablename, operation,idtb,refer)

VALUES (now(),TG\_TABLE\_NAME,TG\_OP, New.service\_hotel,New.hotel);

end if;

if ( TG\_OP = 'UPDATE' ) THEN

INSERT INTO temper(tim, tablename, operation,idtb, refer)

VALUES (now(),TG\_TABLE\_NAME,TG\_OP,Old.service\_hotel,Old.hotel);

end if;

if ( TG\_OP = 'DELETE' ) THEN

INSERT INTO temper(tim, tablename, operation,idtb, refer)

VALUES (now(),TG\_TABLE\_NAME,TG\_OP,Old.service\_hotel,Old.hotel);

end if;

end;

end if;

if ( TG\_TABLE\_NAME = 'room' ) THEN

begin

if ( TG\_OP = 'INSERT' ) THEN

INSERT INTO temper(tim, tablename, operation, idtb, price,status,location,level,refer)

VALUES (now(),TG\_TABLE\_NAME,TG\_OP, New.id\_room,New.price,New.status,New.location ,New.level ,New.hotel);

end if;

if ( TG\_OP = 'UPDATE' ) THEN

INSERT INTO temper(tim, tablename, operation, idtb, price,status,location,level,refer)

VALUES (now(),TG\_TABLE\_NAME,TG\_OP, Old.id\_room,Old.price,Old.status,Old.location,Old.level ,Old.hotel);

end if;

if ( TG\_OP = 'DELETE' ) THEN

INSERT INTO temper(tim, tablename, operation, idtb, price,status,location,level,refer)

VALUES (now(),TG\_TABLE\_NAME,TG\_OP, Old.id\_room,Old.price,Old.status,Old.location,Old.level ,Old.hotel);

end if;

end;

end if;

if ( TG\_TABLE\_NAME = 'service\_room' ) THEN

begin

if ( TG\_OP = 'INSERT' ) THEN

INSERT INTO temper(tim, tablename, operation,idtb,services)

VALUES (now(),TG\_TABLE\_NAME,TG\_OP, New.id\_service\_room,New.services\_room);

end if;

if ( TG\_OP = 'UPDATE' ) THEN

INSERT INTO temper(tim, tablename, operation,idtb, services)

VALUES (now(),TG\_TABLE\_NAME,TG\_OP,Old.id\_service\_room,Old.services\_room);

end if;

if ( TG\_OP = 'DELETE' ) THEN

INSERT INTO temper(tim, tablename, operation,idtb, services)

VALUES (now(),TG\_TABLE\_NAME,TG\_OP,Old.id\_service\_room,Old.services\_room);

end if;

end;

end if;

if ( TG\_TABLE\_NAME = 'service\_rendered\_room' ) THEN

begin

if ( TG\_OP = 'INSERT' ) THEN

INSERT INTO temper(tim, tablename, operation,idtb,refer)

VALUES (now(),TG\_TABLE\_NAME,TG\_OP, New.service\_room,New.room);

end if;

if ( TG\_OP = 'UPDATE' ) THEN

INSERT INTO temper(tim, tablename, operation,idtb, refer)

VALUES (now(),TG\_TABLE\_NAME,TG\_OP,Old.service\_room,Old.room);

end if;

if ( TG\_OP = 'DELETE' ) THEN

INSERT INTO temper(tim, tablename, operation,idtb, refer)

VALUES (now(),TG\_TABLE\_NAME,TG\_OP,Old.service\_room,Old.room);

end if;

end;

end if;

RETURN Null;

END;

$select\_to$

LANGUAGE plpgsql;

/\* триггер, срабатывающий при внесений изменей в таблицы: «Гостиница», «Номер»,«Услуги гостиницы», «Услуги номера», «Оказание услуг гостиницей», «Оказание услуг номером» в виде добавления, удаления или обновления записи \*/

CREATE TRIGGER trigger\_for\_tbs

AFTER INSERT OR UPDATE OR DELETE

ON hotel

FOR EACH ROW

EXECUTE PROCEDURE process\_with\_tables();