UFMS
Engenharia de Software

Laboratório de Banco de Dados T1 - 2023/02 - Vanessa Araujo Borges

Trabalho Prático - Sistema hoteleiro - Reserv

Matheus Nantes Rezende da Silva

11/11/2023

Índice

1. Especificação do Problema

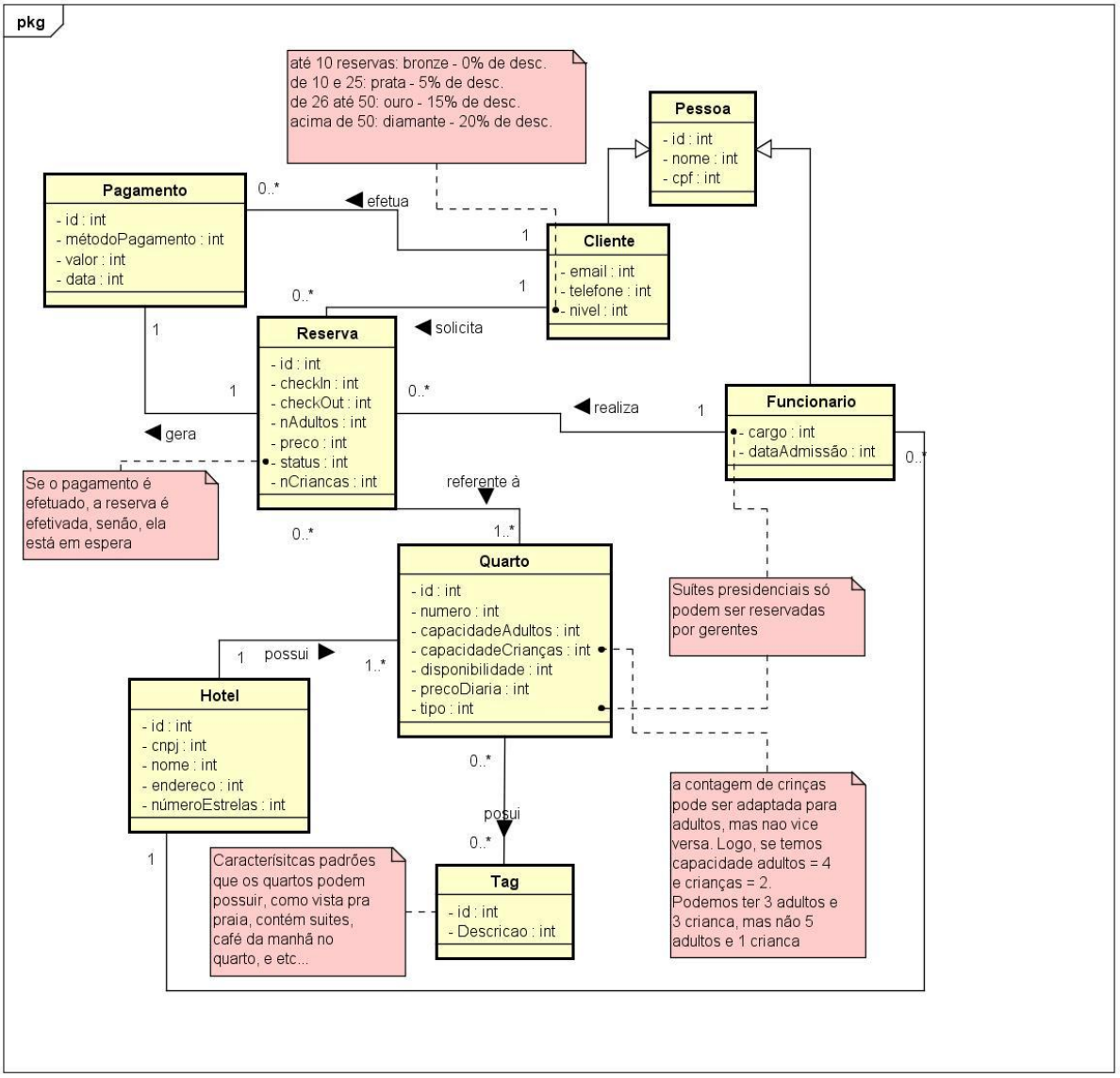
Em um sistema hoteleiro, é preciso cadastrar funcionários, clientes, quartos e hotéis, bem como realizar o gerenciamento de cada uma dessas entidades, bem como os seus relacionamentos.

Alguns regras de negócio foram definidas, como: validação de cpf, validação de perído de tempo de uma reserva, pois se um quarto já está ocupado naquele período, não deve ser permitido realizar uma reserva em tal período, o valor total de uma reserva é um atributo derivado, pois é resultante do cálculo dos preços dos quartos reservados multiplicado pelos dias que tais quartos serão reservados,também deverá ser feito o login de um usuário, que no nosso caso é um funcionário do hotel, e sua senha e cpf deverá ser validado, entre outras especificidades.

Neste trabalho foram implementados para todas as entidades os métodos get, e post. Algumas entidades, possuem alguns casos de uso extras, como funcionário que tem a parte da validação de login. A entidade que teve o CRUD implementado completamente foi o Cliente, e tal CRUD pode ser feito completamente na página :"http://localhost:3000/cliente", onde é possível ver os clientes cadastrados, editar e excluir um cliente específico, ou então adicionar um novo cliente ao banco de dados.

2. Esquema Relacional e trigger

Para mostrar as relações entre as entidades deste trabalho foi desenvolvido um modelo relacional, com entidades, atributos e multiplicidades. Tal diagrama está em .astah e em .jpg no zip enviado.



O trigger implementado foi o cliente_log, pois a única entidade que possui um CRUD completo neste sistema, editável através do frontend desenvolvido, foi a entidade Cliente, logo que é a classe que possui mais variação de operações. tail trigger, e sua respectiva função pode ser visualizado na imagem a seguir e também no arquivo “trigger.sql” enviado no zip, bem como na última migration executada no backend. O trigger implementado é de caráter básico, pois as operações realizadas em cliente são básicas também, então apenas há um identificação de qual operação foi executado, em qual cliente, e em qual momento.

```sql
create table cliente_log(
    data date,
    id TEXT,
    tipo char);

create or replace function clientelog()
returns trigger as $$
declare
agora timestamp := current_date;
begin
    if TG_OP = 'UPDATE' then
        insert into cliente_log values(agora, old.id, 'U');
        return new;
    elsif TG_OP='DELETE' then
        insert into cliente_log values(agora, old.id, 'D');
        return new;
    elsif TG_OP='INSERT' then
        insert into cliente_log values(agora, old.id, 'I');
        return new;
    end if;
    return new;
end;
$$ language plpgsql;

create or replace trigger clientelog
before update or delete on cliente for each row
execute function clientelog();
```

3. Tecnologias utilizadas

O SGBD utilizado foi o postgreSQL, devido à familiaridade que tenho com ele.

Para o backend foi utilizado o ORM Prisma, com a linguagem typesscript.

Para o frontend foi utilizado o Framework Nextjs, um “framework do framework” Reactjs, com a linguagem javascript.

4. Tutorial de instalação

Primeiramente será preciso abrir dois terminais, onde um deverá estar no diretório “T1-LBD\frontend\t1_lbd_reserv”, para o frontend, e outro em “T1-LBD\backend”.

Dentro de cada um desses diretórios, deverão ser executados os comandos “npm install”, ou então “yarn add *” para que todas as dependências sejam instaladas, e elas são muitas, antes do envio cerca de 500 mb de dependências tiveram que ser removidos para poder ser realizado o envio, pois o limite do Ava é de 100mb.

Após realizar a instalação das dependências, deverá ser executado nos dois terminais o comando “yarn dev”, e é importante utilizar o yarn para que funcione corretamente.

Após isso, o servidor backend estará rodando em localhost:3333, e o frontend estará rodando na porta 3000.

Também é possível acessar o servidor backend com o postgresql, sendo possível visualizar as tabelas, constraints, triggers e funções.