**MIEIC 2º ANO**

**Bases de Dados**

**Abril 2019**

**Base de Dados de**

**Site de Reservas de Alojamento Local**

Turma 6 - Grupo 4

João Praça - up201704748

Lucas Ribeiro - up201705227

Sílvia Rocha - up201704684

**Índice**

[Descrição do Tema e da sua Modelação 3](#_2bn6wsx)

[Modelo Conceptual 4](#_qsh70q)

[Esquema Relacional 5](#_1pxezwc)

[Análise de Dependências Funcionais e Formas Normais 6](#_147n2zr)

[Restrições da Base de Dados 9](#_23ckvvd)

[Anexos 14](#_32hioqz)

# Descrição do Tema e da sua Modelação

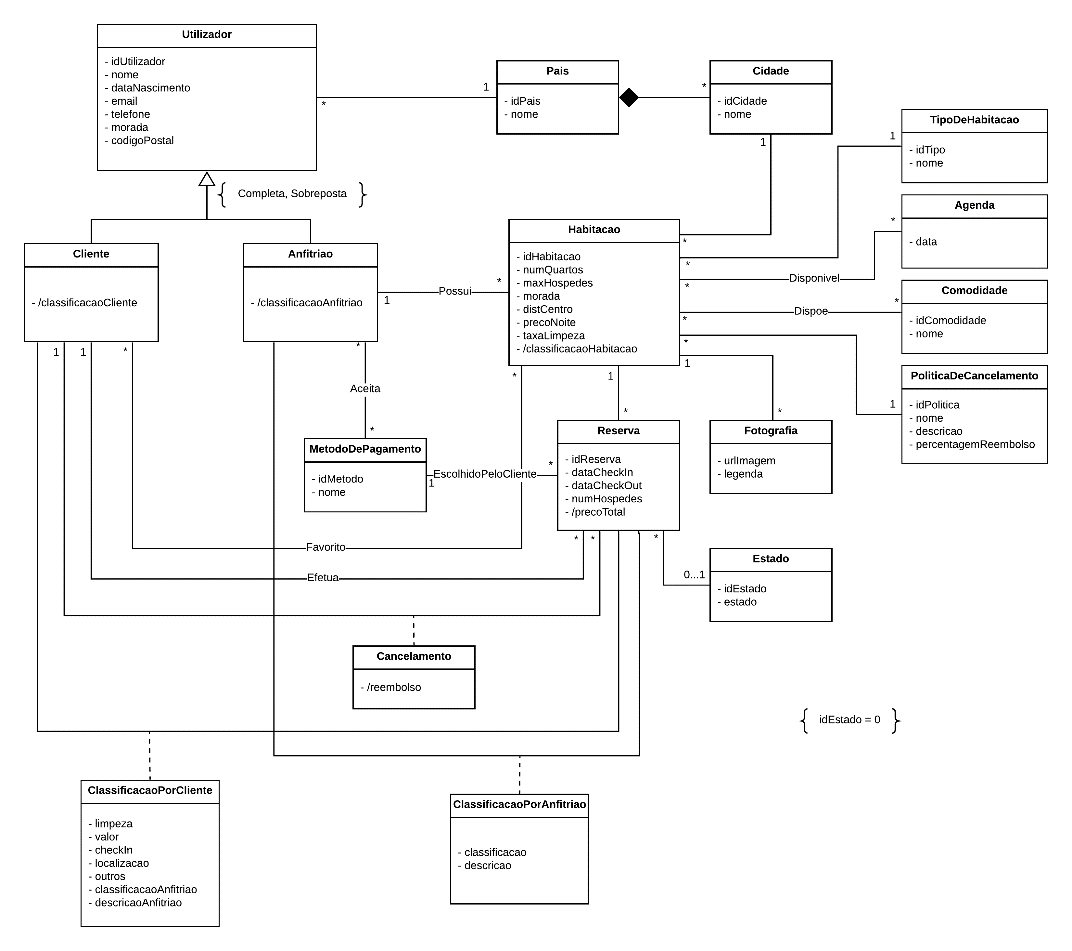
Baseado no sistema e organização da *Airbnb*, esta plataforma *online* permite aos seus clientes reservarem habitações de alojamento local fornecidas por anfitriões espalhados por todo o mundo. Aquando do registo, cada utilizador deve fornecer o seu nome, data de nascimento, e-mail, telefone, nacionalidade, morada e código postal. Após estar registado, um cliente poderá efetuar reservas e marcar habitações como favoritas. Caso o utilizador seja um anfitrião, tem ainda de escolher os métodos de pagamento que aceita. Assim que esteja registado, poderá adicionar as suas habitações. Para além disso, um utilizador pode ser um cliente e anfitrião simultaneamente.

Por sua vez, uma habitação pode ser caracterizada segundo o seu tipo (moradia, apartamento, entre outros), número de quartos, número máximo de hóspedes, morada, cidade, distância ao centro da cidade, preço por noite, disponibilidade, comodidades (ar-condicionado, Wi-Fi, entre outros), taxa de limpeza, algumas fotografias e deverá estar associada a uma política de cancelamento de reservas.

Ao realizar uma reserva, o cliente deve especificar a data de check-in, data de check-out, o método de pagamento dentro dos disponíveis e o número de hóspedes, sendo-lhe apresentado o preço total. Após concluída, a reserva fica associada a um estado da estadia (pendente, a decorrer ou concluída) e pode também ser cancelada (no caso de a estadia estar pendente), originando um reembolso de valor variável, de acordo com a política de cancelamento escolhida pelo anfitrião. No final de uma estadia, o cliente classifica a habitação em vários parâmetros (limpeza, valor, localização, entre outros) com uma pontuação de 0 a 5, e o seu anfitrião com uma pontuação geral e uma pequena descrição. O anfitrião também deverá avaliar o hóspede, mais uma vez com uma pontuação geral e uma descrição. Desta forma, o cliente, anfitrião e habitação são também caracterizados por uma classificação, que será a média aritmética das classificações que lhes foram atribuídas.

**Modelo Conceptual**

Com base na estrutura apresentada no ponto anterior do presente documento, obtivemos o modelo conceptual visível no diagrama UML seguinte (nos Anexos é possível ver o diagrama com mais detalhe).



Como é visível na imagem, este modelo conceptual é composto por:

* dezassete classes.
* dezanove associações de entre as quais uma é composta, duas são derivadas e uma contém três classes de associação.

**Esquema Relacional**

Utilizador (id, nome, dataNascimento, email, telefone, morada, codigoPostal, pais->Pais)

Cliente (id->Utilizador, classificacaoCliente)

Anfitriao (id->Utilizador, classificacaoAnfitriao)

Habitacao (id, numQuartos, maxHospedes, morada, distCentro, precoNoite, taxaLimpeza, classificacaoMedia,cidade->Cidade,tipo->TipoDeHabitacao, politica->PoliticaDeCancelamento)

Reserva (id, dataCheckIn, dataCheckOut, numHospedes, precoTotal, habitacao->Habitacao, estado->Estado)

Estado (id, estado)

MetodoDePagamento (id, nome)

Pais (id, nome)

Cidade (id, nome, pais->Pais)

TipoDeHabitacao (id, nome)

Agenda (data)

Comodidade (id, nome)

PoliticaDeCancelamento (id, nome, descricao, percentagemReembolso)

Fotografia (urlImagem, legenda, habitacao->Habitacao)

ClassificacaoPorCliente (limpeza, valor, checkIn, localizacao, outros, classificacaoAnfitriao, descricaoAnfitriao, anfitriao->Anfitriao, reserva->Reserva)

ClassificacaoPorAnfitriao (classificacao, descricao, cliente->Cliente, reserva->Reserva)

Possui (anfitriao->Anfitriao, habitacao->Habitacao)

Favorito (cliente->Cliente, habitacao->Habitacao)

Efetua (cliente->Cliente, reserva->Reserva)

Cancelamento (cliente->Cliente, reserva->Reserva, reembolso)

Aceita (anfitriao->Anfitriao, metodo->MetodoDePagamento)

EscolhidoPeloCliente (metodo->MetodoDePagamento, reserva->Reserva)

Disponivel (data->Agenda, habitacao->Habitacao)

Dispoe (comodidade->Comodidade, habitacao->Habitacao)

**Análise de Dependências Funcionais e Formas Normais**

**Utilizador:**

Utilizador (id, nome, dataNascimento, email, telefone, morada, codigoPostal, pais->Pais)

* id -> nome, dataNascimento, email, telefone, morada, codigoPostal, pais
* email -> id, nome, dataNascimento, telefone, morada, codigoPostal, pais
* telefone -> id, nome, dataNascimento, email, morada, codigoPostal, pais

**Justificação**: id, email e telefone são chave de Utilizador. Sendo que todos os atributos do lado esquerdo das dependências funcionais são chaves, a relação está na BCNF e consequentemente na 3rd NF.

**Habitacao:**

Habitacao (id, numQuartos, maxHospedes, morada, distCentro, precoNoite, taxaLimpeza, classificacaoMedia, cidade->Cidade, tipo->TipoDeHabitacao, politica->PoliticaDeCancelamento)

* id -> numQuartos, maxHospedes, morada, distCentro, precoNoite, taxaLimpeza, classificacaoMedia, cidade, tipo, politica
* morada -> id, numQuartos, maxHospedes, distCentro, precoNoite, taxaLimpeza, classificacaoMedia, cidade, tipo, politica

**Justificação**: id e morada são chaves de Habitação. Sendo que todos os atributos do lado esquerdo das dependências funcionais são chaves, a relação está na BCNF e consequentemente na 3rd NF.

**Reserva:**

Reserva (id, dataCheckIn, dataCheckOut, numHospedes, precoTotal, habitacao->Habitacao, estado->Estado)

* id -> dataCheckIn, dataCheckOut, numHospedes, precoTotal, habitação, estado->Estado
* dataCheckIn, habitação -> id, dataCheckOut, numHospedes, precoTotal, estado->Estado

**Justificação**: id e o tuplo dataCheckIn e habitação são chaves de Reserva. Sendo que todos os atributos do lado esquerdo das dependências funcionais são chaves, a relação está na BCNF e consequentemente na 3rd NF.

**Estado:**

Estado (id, estado)

* id -> estado
* estado -> id

**Justificação**: id e estado são chaves de Estado. Sendo que todos os atributos do lado esquerdo das dependências funcionais são chaves, a relação está na BCNF e consequentemente na 3rd NF.

**MetodoDePagamento:**

MetodoDePagamento (id, nome)

* id -> nome
* nome -> id

**Justificação**: id e nome são chaves de MetodoDePagamento. Sendo que todos os atributos do lado esquerdo das dependências funcionais são chaves, a relação está na BCNF e consequentemente na 3rd NF.

**Pais:**

Pais (id, nome)

* id -> nome
* nome -> id

**Justificação**: id e nome são chaves de Pais. Sendo que todos os atributos do lado esquerdo das dependências funcionais são chaves, a relação está na BCNF e consequentemente na 3rd NF.

**Cidade:**

Cidade (id, nome, pais->Pais)

* id -> nome, pais
* nome -> id, pais

**Justificação**: id e nome são chaves de Cidade. Sendo que todos os atributos do lado esquerdo das dependências funcionais são chaves, a relação está na BCNF e consequentemente na 3rd NF.

**TipoDeHabitacao:**

TipoDeHabitacao (id, nome)

* id -> nome
* nome -> id

**Justificação**: id e nome são chaves de TipoDeHabitacao. Sendo que todos os atributos do lado esquerdo das dependências funcionais são chaves, a relação está na BCNF e consequentemente na 3rd NF.

**Comodidade:**

Comodidade (id, nome)

* id -> nome
* nome -> id

**Justificação**: id e nome são chaves de Comodidade. Sendo que todos os atributos do lado esquerdo das dependências funcionais são chaves, a relação está na BCNF e consequentemente na 3rd NF.

**PoliticaDeCancelamento:**

PoliticaDeCancelamento (id, nome, descricao, percentagemReembolso)

* id -> nome, descricao, percentagemReembolso
* nome -> id, descricao, percentagemReembolso

**Justificação**: id e nome são chaves de PoliticaDeCancelamento. Sendo que todos os atributos do lado esquerdo das dependências funcionais são chaves, a relação está na BCNF e consequentemente na 3rd NF.

**Fotografia:**

Fotografia (urlImagem, legenda, habitacao->Habitacao)

* urlImagem -> legenda, habitacao

**Justificação**: urlImagem é chave de Fotografia. Sendo que todos os atributos do lado esquerdo das dependências funcionais são chaves, a relação está na BCNF e consequentemente na 3rd NF.

**ClassificacaoPorCliente:**

ClassificacaoPorCliente (limpeza, valor, checkIn, localizacao, outros, classificacaoAnfitriao, descricaoAnfitriao, anfitriao->Anfitriao, reserva->Reserva)

* reserva -> limpeza, valor, checkIn, localizacao, outros, classificacaoAnfitriao, descricaoAnfitriao, anfitriao

**Justificação**: reserva é chave de ClassificacaoPorCliente. Sendo que todos os atributos do lado esquerdo das dependências funcionais são chaves, a relação está na BCNF e consequentemente na 3rd NF.

**ClassificacaoPorAnfitriao:**

ClassificacaoPorAnfitriao (classificacao, descricao, cliente->Cliente, reserva->Reserva)

* reserva -> descricao, classificacao, cliente

**Justificação**: reserva é chave de ClassificacaoPorAnfitriao. Sendo que todos os atributos do lado esquerdo das dependências funcionais são chaves, a relação está na BCNF e consequentemente na 3rd NF.

**Possui:**

Possui (anfitriao->Anfitriao, habitacao->Habitacao)

* habitação -> anfitriao

**Justificação**: habitação é chave de Possui. Sendo que todos os atributos do lado esquerdo das dependências funcionais são chaves, a relação está na BCNF e consequentemente na 3rd NF.

**Efetua:**

Efetua (cliente->Cliente, reserva->Reserva)

* reserva -> cliente

**Justificação**: reserva é chave de Efetua. Sendo que todos os atributos do lado esquerdo das dependências funcionais são chaves, a relação está na BCNF e consequentemente na 3rd NF.

**Cancelamento:**

Cancelamento (cliente->Cliente, reserva->Reserva, reembolso)

* reserva -> reembolso, cliente

**Justificação**: reserva é chave de Cancelamento. Sendo que todos os atributos do lado esquerdo das dependências funcionais são chaves, a relação está na BCNF e consequentemente na 3rd NF.

**EscolhidoPeloCliente:**

EscolhidoPeloCliente (metodo->MetodoDePagamento, reserva->Reserva)

* reserva -> metodo

**Justificação**: reserva é chave de EscolhidoPeloCliente. Sendo que todos os atributos do lado esquerdo das dependências funcionais são chaves, a relação está na BCNF e consequentemente na 3rd NF.

**Outras Relações:**

As restantes relações não apresentam dependências funcionais não triviais.

**Restrições da Base de Dados**

**Agenda:**

Não pode haver dois dias repetidos (restrição PRIMARY KEY).

**País:**

Não pode haver dois países com o mesmo nome (restrição PRIMARY KEY).

Não pode haver países com nome nulo (restrição NOT NULL).

Não pode haver dois países com o mesmo nome (restrição UNIQUE).

**Cidade:**

Não pode haver duas cidades com o mesmo nome (restrição PRIMARY KEY).

Não pode haver cidades com nome nulo (restrição NOT NULL).

Não pode haver cidades associadas a um país inexistente (restrição de integridade referencial).

**Utilizador:**

Não pode haver dois utilizadores com o mesmo id (restrição PRIMARY KEY).

Não pode haver utilizadores com nome nulo (restrição NOT NULL).

Não pode haver utilizadores com data de nascimento nula (restrição NOT NULL).

Não pode haver utilizadores com e-mail nulo (restrição NOT NULL).

Não pode haver dois utilizadores com o mesmo e-mail (restrição UNIQUE).

Não pode haver utilizadores com telefone nulo (restrição NOT NULL).

Não pode haver dois utilizadores com o mesmo telefone (restrição UNIQUE).

Não pode haver utilizadores com morada nula (restrição NOT NULL).

Não pode haver utilizadores com código postal nulo (restrição NOT NULL).

Não pode haver utilizadores com classificação média menor que 1 ou maior que 5 (restrição CHECK).

Não pode haver utilizadores associados a um país inexistente (restrição de integridade referencial).

**Cliente:**

Não pode haver clientes associados a um utilizador inexistente (restrição de integridade referencial).

**Anfitrião:**

Não pode haver anfitriões associados a um utilizador inexistente (restrição de integridade referencial).

**Método de Pagamento:**

Não pode haver dois métodos de pagamento com o mesmo id (restrição PRIMARY KEY).

Não pode haver dois métodos de pagamento com o mesmo nome (restrição UNIQUE).

Não pode haver métodos de pagamento com nome nulo (restrição NOT NULL).

**Aceita:**

Não pode haver anfitriões associados a um anfitrião inexistente (restrição de integridade referencial).

Não pode haver métodos associados a um método inexistente (restrição de integridade referencial).

Não pode haver duas combinações de anfitriões e métodos iguais (restrição PRIMARY KEY).

**Reserva:**

Não pode haver duas reservas com o mesmo id (restrição PRIMARY KEY).

Não pode haver reservas com data de check-in nula (restrição NOT NULL).

Não pode haver reservas com data de check-out nula (restrição NOT NULL).

Não pode haver reservas com número de hóspedes menor que 1 (restrição CHECK).

Não pode haver reservas com preço total menor ou igual a 0 (restrição CHECK).

Não pode haver reservas associadas a uma habitação inexistente (restrição de integridade referencial).

Não pode haver duas combinações de data de check-in e habitação iguais (restrição UNIQUE).

**Estado:**

Não pode haver dois estados com o mesmo id (restrição PRIMARY KEY).

Não pode haver estados com estado nulo (restrição NOT NULL).

Não pode haver dois estados com o mesmo estado (restrição UNIQUE).

Não pode haver estados associados a uma reserva inexistente (restrição de integridade referencial).

**Cancelamento:**

Não pode haver cancelamentos com reembolso nulo (restrição NOT NULL).

Não pode haver cancelamentos associados a um cliente inexistente (restrição de integridade referencial).

Não pode haver cancelamentos associados a uma reserva inexistente (restrição de integridade referencial).

Não pode haver dois cancelamentos com a mesma reserva (restrição PRIMARY KEY).

**Classificação por Anfitrião:**

Não pode haver classificações menor que 1 e maior que 5 (restrição CHECK).

Não pode haver classificações associadas a um anfitrião inexistente (restrição de integridade referencial).

Não pode haver classificações associadas a uma reserva inexistente (restrição de integridade referencial).

Não pode haver duas classificações para a mesma reserva (restrição PRIMARY KEY).

**Classificação por Cliente:**

Não pode haver limpeza menor que 1 e maior que 5 (restrição CHECK).

Não pode haver valor menor que 1 e maior que 5 (restrição CHECK).

Não pode haver checkIn menor que 1 e maior que 5 (restrição CHECK).

Não pode haver localização menor que 1 e maior que 5 (restrição CHECK).

Não pode haver classificacaoAnfitriao menor que 1 e maior que 5 (restrição CHECK).

Não pode haver classificações associadas a um cliente inexistente (restrição de integridade referencial).

Não pode haver classificações associadas a uma reserva inexistente (restrição de integridade referencial).

Não pode haver duas classificações para a mesma reserva (restrição PRIMARY KEY).

**Comodidade:**

Não pode haver duas comodidades com o mesmo id (restrição PRIMARY KEY).

Não pode haver comodidades com nome nulo (restrição NOT NULL).

Não pode haver duas comodidades com nome igual (restrição UNIQUE).

**Efetua:**

Não pode haver clientes associados a um cliente inexistente (restrição de integridade referencial).

Não pode haver reservas associados a uma reserva inexistente (restrição de integridade referencial).

Não pode haver duas combinações de clientes e reservas iguais (restrição PRIMARY KEY).

**Escolhido Pelo Cliente:**

Não pode haver métodos associados a um método inexistente (restrição de integridade referencial).

Não pode haver reservas associados a uma reserva inexistente (restrição de integridade referencial).

Não pode haver duas combinações de métodos e reservas iguais (restrição PRIMARY KEY).

**Tipo De Habitação:**

Não pode haver dois tipos de habitação com o mesmo id (restrição PRIMARY KEY).

Não pode haver um tipo de habitação com nome nulo (restrição NOT NULL).

Não pode haver dois tipos de habitação com nome igual (restrição UNIQUE).

**Política De Cancelamento:**

Não pode haver duas políticas com o mesmo id (restrição PRIMARY KEY).

Não pode haver uma política com nome nulo (restrição NOT NULL).

Não pode haver duas políticas com nome igual (restrição UNIQUE).

Não pode haver uma política com descrição nula (restrição NOT NULL).

Não pode haver duas políticas com descrição igual (restrição UNIQUE).

A percentagem de reembolso não pode ser inferior a 0 ou superior a 1 (restrição CHECK).

**Habitação:**

Não pode haver duas habitações com o mesmo id (restrição PRIMARY KEY).

O número de quartos não pode ser inferior a 0 (restrição CHECK).

O número máximo de hóspedes não pode ser inferior a 0 (restrição CHECK).

Não pode haver uma habitação com morada nula (restrição NOT NULL).

Não pode haver duas habitações com morada igual (restrição UNIQUE).

A distância ao centro não pode ser inferior a 0 (restrição CHECK).

O preço por noite não pode ser inferior ou igual a 0 (restrição CHECK).

A taxa de limpeza não pode ser inferior a 0 (restrição CHECK).

O valor da classificação média não pode ser inferior a 1 nem superior a 5 (restrição CHECK).

Não pode haver cidades associadas a uma cidade inexistente (restrição de integridade referencial).

Não pode haver tipos associados a um tipo de habitação inexistente (restrição de integridade referencial).

Não pode haver políticas associadas a uma política de cancelamento inexistente (restrição de integridade referencial).

**Disponível:**

Não pode haver datas associadas a uma data inexistente (restrição de integridade referencial).

Não pode haver habitações associadas a uma habitação inexistente (restrição de integridade referencial).

Não pode haver duas combinações de datas e habitações iguais (restrição PRIMARY KEY).

**Dispõe:**

Não pode haver comodidades associadas a uma comodidade inexistente (restrição de integridade referencial).

Não pode haver habitações associadas a uma habitação inexistente (restrição de integridade referencial).

Não pode haver duas combinações de comodidades e habitações iguais (restrição PRIMARY KEY).

**Favorito:**

Não pode haver clientes associados a um cliente inexistente (restrição de integridade referencial).

Não pode haver habitações associadas a uma habitação inexistente (restrição de integridade referencial).

Não pode haver duas combinações de clientes e habitações iguais (restrição PRIMARY KEY).

**Fotografia:**

Não pode haver duas fotografias com o mesmo urlImagem(restrição PRIMARY KEY).

Não pode haver habitações associadas a uma habitação inexistente (restrição de integridade referencial).

**Possui:**

Não pode haver anfitriões associados a um anfitrião inexistente (restrição de integridade referencial).

Não pode haver habitações associadas a uma habitação inexistente (restrição de integridade referencial).

Não pode haver duas combinações de anfitriões e habitações iguais (restrição PRIMARY KEY).

**Descrição sucinta dos triggers**

⦁ **Trigger número 1 -** Tem como objetivo impedir que um cliente classifique um Anfitrião., antes que a reserva esteja concluída. O trigger atinge este objectivo ao ignorar qualquer operação de insert na classificação de anfitrião que seja efetuada, antes que o idEstado esteja igual a 0.

⦁ **Trigger número 2 -** Tem como objetivo atualizar a classificação dos anfitriões com a média das classificações ao anfitrião pela totalidade dos clientes, tal como atualizar a classificação das habitações com a média das médias das classificações atribuídas pelos clientes a cada critério da classificação de uma habitação. O trigger atinge este objetivo ao fazer as atualizações referidas depois de cada tentativa de inserção de novos dados na relação de classificação por cliente.

⦁ **Trigger número 3 -** Tem como objetivo tornar a casa indisponível em datas para as quais já se efetuaram reservas. Para atingir este objetivo, o trigger apaga todas as relações de disponibilidade com id igual ao da habitação à qual se tentou efetuar uma operação de insert na reserva.

Para que o funcionamento devido da base de dados fosse atingido seria também necessária implementação de pelo menos mais 5 triggers, sendo o propósito destes:

⦁ Impedir que um cliente classifique uma habitação antes que a reserva esteja concluída.

⦁ Atualizar a classificação de um cliente sempre que um anfitrião o classifica.

⦁ Atualizar as datas de disponibilidade sempre que se cancelar uma reserva.

⦁ Atualizar o valor do reembolso aquando o cancelamento de uma reserva.

⦁ Atualizar o preço total aquando a criação de uma reserva.

Também seria necessário criar triggers semelhantes aos já criados mas para tipos de eventos diferentes (inserção, apagamento etc.).

Por exemplo, apesar de ja existir um trigger que atualiza a classificação dos anfitriões com a média das classificações ao anfitrião pela totalidade dos clientes, seria também necessário criar um que ajusta-se a classificação de um anfitrião caso um dos clientes que o classificou fosse eliminado.

**Anexos**