Universidade Federal São João del-Rei
Curso de Ciência da Computação

Trabalho Prático - Parte 3
Implementação do Sistema

Banco de Dados

Discentes: Júlio César Ferreira,
Elias De Paula Pereira,
Patrick Leandro Magalhães,
Juan Victor Costa Silva Aleixo e
Julio Cesar da Silva Rodrigues

Docente: Leonardo Chaves Dutra da Rocha

Dezembro
2022

Conteúdo

1 Introdução

Neste documento, serão apresentados de forma detalhada, os aspectos envolvidos nas modificações e correções realizadas nos modelos conceituais para modelagem de um banco de dados para uma rede de hotéis, cujo desenvolvimento foi iniciado na primeira parte do trabalho prático.

Serão abordadas as moficações mais cruciais e relevantes exploradas nos modelos de Entidade-Relacionamento Estendido e Relacional, a implementação atual do banco de dados de forma geral, funcionalidades presentes no sistema, limitações e também outros aspectos relacionados a requisitos não funcionais, como dependências funcionais e normalização.

2 Modificações nos Modelos Conceituais

Nesta seção, serão abordadas entidades, relacionamentos e seus respectivos atributos que sofreram quaisquer modificações, assim como as respectivas restrições de domínio que foram analisadas na construção dos modelos conceituais do banco de dados.

2.1 Modelo Entidade-Relacionamento Estendido

À seguir é apresentada por extenso a composição geral do modelo EER, contendo suas entidades, respectivos atributos e cada restrição de domínio correspondente:

- *Usuário*
  - **Login**: Atributo do tipo *String* com 45 caracteres variável;
  - **Senha**: Atributo do tipo *String* com 45 caracteres variável;
  - **CPF**: Atributo do tipo *Integer*, único com 11 caracteres;
  - **Primeiro_nome**: Atributo do tipo *String* com 45 caracteres variável;
  - **Sobrenome**: Atributo do tipo *String* com 45 caracteres variável;
  - **Datanascimento**: Atributo do tipo *Date* (AAAA-MM-DD);
  - **Telefone**: Atributo do tipo *Integer* com 11 caracteres;
  - **Endereço**: Atributo do tipo *String* com 45 caracteres variável;
  - **Email**: Atributo do tipo *String* com 45 caracteres variável;
  - **Diárias**: Atributo do tipo *Integer* variável.

- *Dependente* (Entidade Fraca)
  - **CPF**: Atributo do tipo *Integer*, único com 11 caracteres;
  - **Primeiro_nome**: Atributo do tipo *String* com 45 caracteres variável;
  - **Sobrenome**: Atributo do tipo *String* com 45 caracteres variável;
  - **Datanascimento**: Atributo do tipo *Date* (AAAA-MM-DD).
- *Hotel*
  - **Cod_hotel**: Atributo do tipo *Integer*, único e de tamanho variável;
  - **Localizacao**: Atributo do tipo *String* multivalorado;
  - **Nome**: Atributo do tipo *String* com 45 caracteres variável.
- *Quarto*
  - **Número**: Atributo do tipo *Integer*, único e de tamanho variável;
  - **Descrição**: Atributo do tipo *String* com 45 caracteres variável;
  - **Tipo**: Atributo do tipo *String* com 45 caracteres variável (Standard, Luxo, Premium, ...);
  - **Capacidade**: Atributo do tipo *Integer* variável;
  - **Status**: Atributo do tipo *Boolean* que indica se o quarto está ou não ocupado.
- *Reserva*
  - **Cod_reserva**: Atributo do tipo *Integer*, único com 11 caracteres;
  - **Valor_total**: Atributo do tipo *Double* variável;
  - **Valor_desconto**: Atributo do tipo *Integer* variável;
  - **Feriado**: Atributo do tipo *Boolean* para indicar reservas realizadas em período de feriado;
  - **Estado**: Atributo do tipo *String* com 9 caracteres {'reservado', 'hospedado', 'cancelado'};
  - **Data_inicial**: Atributo do tipo *Date* (AAAA-MM-DD);
  - **Data_final**: Atributo do tipo *Date* (AAAA-MM-DD);
  - **Num_ocupantes**: Atributo do tipo *Integer* variável.

- *Cupom*
  - **Cod_cupom**: Atributo do tipo *Integer* variável;
  - **Status**: Atributo do tipo *Boolean* para identificar se o cupom é válido ou não;
  - **Valor_desconto**: Atributo do tipo *Integer* variável.
- *Feriado*
  - **Cod_feriado**: Atributo do tipo *Integer*, único com 11 caracteres;
  - **Nome**: Atributo do tipo *String* com 45 caracteres (variável);
  - **Data_inicial**: Atributo do tipo *Date* (AAAA-MM-DD);
  - **Data_final**: Atributo do tipo *Date* (AAAA-MM-DD).

Os relacionamentos presentes neste modelo EER são sumarizados à seguir:

- *DEPENDENTES_DE*: Relação entre Dependente e Usuário. Um usuário pode possuir múltiplos dependentes ($1 : N$). Dependente possui participação total, pois todo dependente deve estar obrigatoriamente relacionado a um usuário. Usuário possui participação parcial, já que este pode ou não ter dependentes associados. É um relacionamento de identificação, visto que Dependente é uma entidade fraca.
- *TRABALHA_EM*: Relação entre Administrador e Hotel. Cada hotel possui um administrador ($1 : 1$). Participação total de ambas as entidades, pois todo administrador deve estar associado à um hotel e todo hotel deve possuir um administrador.
- *CATALOGA*: Relação entre Administrador e Feriado. Um administrador pode cadastrar múltiplos feriados ($1 : N$). Feriado possui participação total, pois todo feriado deve ser catalogado por um administrador, enquanto administrador tem participação parcial, já que este pode ou não catalogar feriados.
- *GANHA*: Relação entre Cupom e Cliente. Um cliente pode ganhar múltiplos cupons ($1 : N$). Participação parcial de ambas as entidades, pois nem todo cliente terá cupons e o cadastro de um cupom não depende da existência de um cliente.
- *REALIZA*: Relação entre Cliente e Reserva. Um cliente pode realizar múltiplas reservas ($1 : N$). Reserva possui participação total, pois toda reserva deve obrigatorimente estar relacionada à um cliente para

existir. Cliente possui participação parcial, já que este pode ou não possuir reservas associadas ao seu CPF.

- *VINCULA*: Relação entre Cupom e Reserva. Uma reserva pode ter múltiplos cupons associados ($1 : N$). Reserva possui participação parcial, pois não é necessário que toda reserva esteja associada à um cupom, enquanto cupom também possui participação parcial, já que este pode existir sem que esteja vinculado à uma reserva (cupom ainda não utilizado).

- *POSSUI*: Relação entre Hotel e Quarto. Um quarto deve estar associado à um único hotel até por questões físicas, e não deve existir sem que esteja associado ao mesmo. Hotel também possui participação total, pois todo hotel deve estar associado à pelo menos um quarto em sua existência ($1 : N$).

2.1.1 Observações

Usuário pode assumir papéis de cliente e/ou administrador de forma conjunta (*overlapped*) se aplicável. Por exemplo, um administrador pode atuar como cliente em ocasiões não habituais (um administrador que está de folga e realizou uma reserva como cliente no hotel em que trabalha), com ambos papéis herdando da entidade *Usuário*.

Além disto, existem algumas entidades e relacionamentos presentes (assim como seus atributos) no modelo inicial que optamos por remover do modelo conceitual final para o banco de dados. Estas são exibidas na Tabela 1:

Tabela 1: Entidades e Relacionamentos Removidos

Entidade	**Relacionamento**
Fornecedor	FF
Produto	FP
Fornece	FH

Os obstáculos encontrados ao lidar com estas entidades foram de que, primeiramente, um *fornecedor* era tratado obrigatoriamente como um usuário previamente cadastrado no sistema do *hotel* correspondente, o que não necessariamente é aplicável na prática. Também foram encontradas dificuldades quanto ao gerenciamento da complexidade de representar tais entidades e relacionamentos como tabelas no banco de dados, no que diz respeito a implementação das funcionalidades básicas e geração de relatórios.

Algumas outras modificações também foram realizadas como a adição de um relacionamento entre *Reserva* e *Cupom* (*VINCULA*), para que um usuário seja capaz de utilizar seus cupons. *Quarto* agora possui relacionamento com *Hotel* diretamente em vez de *Reserva* por questões de consistência, já que a existência de quartos não depende da existência de reservas, e sim de hotel, transferindo a lógica de cadastro de reservas para a aplicação (e.g., valor conforme o tipo de quarto).

Por fim, foram adicionados os atributos correspondentes à capacidade dos quartos e número de ocupantes da reserva, para manter a consistência com as capacidades máximas de cada quarto na realização de reservas. O atributo *Salario* foi removido da entidade *Administrador*, assim como seu relacionamento com *Cupom* (O administrador ainda é capaz de inserir cupons, apenas não existe referência a qual destes fez o cadastro).

A representação ilustrativa do modelo Entidade-Relacionamento Estendido é exibida na página 12 na Figura 2.1.1.

2.2 Modelo Relacional

Nesta seção, será discutida a modelagem do modelo relacional para refletir o modelo EER recém explanado. O diagrama foi formulado utilizando o mesmo algoritmo já aplicado na primeira etapa, mas como algumas nuances relacionadas à adição e modificação de relacionamentos, além da criação de tabelas "extras"na identificação de dependências funcionais e normalização do banco de dados.

2.2.1 Desenvolvimento

Como base, partimos do passo 1 do algoritmo, mapeamento todas as entidades regulares (excetuando a especialização, que será explanada adiante neste relatório) em relações, incluindo todos seus atributos correspondentes. As entidades mapeadas no passo 1 foram as seguintes: Cupom, Feriado, Quarto, Reserva e Hotel.

Na próxima etapa, mapeamos as entidades fracas presente no modelo EER, incluindo todos seus atributos como no passo anterior, mas incluindo como atributo de chave estrangeira, a chave primária da entidade com que esta se relaciona. *Dependente* foi a entidade fraca mapeada no passo 2, enquanto a relação de identificação mapeada foi *DEPENDENTES_DE*.

Em seguida, aplicamos o passo 3 do algoritmo, utilizando a técnica da chave estrangeira para mapear o relacionamento *TRABALHA_EM* entre Administrador e Hotel (FK em Hotel que aponta para a chave primária do Administrador - CPF do Usuário). Como ambas entidades possuem participação

total, poderia ser utilizada a técnica de mesclagem de relações. Optamos por não explorá-la devido ao mapeamento de especialização em relação única aplicada às subclasses e superclasse existentes no modelo, o que julgamos que prejudicaria a interpretação geral da modelagem do banco.

No passo 4, para o mapeamento de todos os relacionamentos binários $(1 : N)$, utilizamos a técnica da chave estrangeira, colocando-a no lado N da relação. Os relacionamentos mapeados desta forma foram os seguintes: *CATALOGA*, *POSSUI*, *VINCULA*, *REALIZA* e *GANHA*.

No passo 6, para o mapeamento do único atributo multivalorado presente no modelo EER, criamos uma nova relação (*Localização_Hotel*), combinando como chave primária o par: código de hotel (também assume o papel de chave estrangeira) e seu respectivo endereço.

Por fim, mas de grande relevância, utilizamos o passo *8D* (classes sobrepostas - *overlapping*) no mapeamento de especialização da entidade Usuário em Cliente e Administrador. Com isso, mesclamos a superclasse e as subclasses em uma única relação contendo os atributos de todas as mesmas, que possui como chave primária, a chave primária da superclasse (CPF de Usuário). Além disso, foram incluídos dois atributos de tipo de relação: *tipo_administrador* e *tipo_cliente*. São atributos do tipo *Boolean* que terão como propósito central, identificar a especialização da relação.

Finalizada a aplicação do algoritmo, foram criadas duas tabelas, a primeira destas sendo *Reserva_Cupom*. A mesma foi adicionada com o intuito de possibilitar a criação de reservas que não possuam cupons vinculados, possuindo como chave primária o par: {cod_reserva, cod_cupom}, que também atuam como chaves estrangeiras individualmente referenciando cupons e reservas.

Já a segunda tabela foi criada a partir da verificação da dependência funcional do atributo *senha* em *Usuário* em relação as chaves candidatas. Este não depende funcionalmente da chave primária *cpf*, então foi necessário criar a tabela *Login* para atingir a 2ª forma normal na tabela *Usuário*. Os detalhes sobre as soluções aplicadas e as dependências funcionais envolvidas no restante do esquema serão detalhadas posteriormente neste documento, assim como a forma normal em que o banco de dados como um todo se encontra.

A representação ilustrativa do modelo relacional é exibida na página 13 na Figura 2.2.1.

3 Normalização

Verificou-se que o atributo *senha* pertencente a tabela usuário, não era dependente funcionalmente de todas as chaves candidatas da relação (e.g., em um cenário hipotético, vários usuários (cpf) podem possuir a mesma senha). Por isto, foi criada uma nova tabela (*Login*), cuja chave estrangeira foi posicionada em usuário. Desta forma, nas duas tabelas em questão, quaisquer atributos são totalmente dependentes funcionalmente de todas as chaves candidatas.

Esta modificação permitiu o alcance da segunda forma normal, com todas as tabelas satisfazendo as mesmas definições formais. Embora tenhamos analisado brevemente níveis superiores de normalização, concluímos que esta é a forma normal máxima do banco no momento. Isto acontece porque existem tabelas cujos atributos não principais podem depender funcionalmente de atributos não principais de outras tabelas (e.g., o valor total de uma reserva depende do tipo de quarto), levando-nos a acreditar que o banco não está na terceira forma normal ou superior.

4 Implementação

A implementação do projeto foi realizada utilizando banco de dados *MySQL*, juntamente com a ferramenta de interface gráfica cliente **MySQL Workbench 8.0** para efetuar as consultas SQL, administrar o sistema, modelar, criar e manter a base de dados por meio de um ambiente integrado.

Para interagir com o banco de dados, foi escolhida a linguagem de programação *PHP*, que possui como característica a facilidade para conexões com banco de dados. Para visualização no ambiente do cliente, utilizou-se o *HTML*, linguagem de marcação utilizada na construção de páginas na *Web*, e o *CSS*, que possibilita estilização de um documento web, melhorando sua formatação e a visualização das informações.

Para configurar o banco de dados, em um primeiro momento foi utilizado o modelos conceituais, desenvolvendo assim, tabelas, relacionamentos, atributos, observando suas características e restrições. Após a primeira implementação, verificou-se a necessidade de uma revisão das tabelas e suas relações, para atender a ferramenta de normalização de banco de dados e as melhores práticas para seu desenvolvimento. Através da normalização e identificação das restrições e regras de negócio já estabelecidas, passou-se a desenvolver as páginas *Web* para utilização dos recursos criados.

4.1 Triggers

Inicialmente, definimos uma *Trigger* para realizar a checagem relacionada à usuários que atingem 20 diárias. Quando tal usuário satisfaz tal condição, este automaticamente ganharia 1 cupom de desconto de 10%.

Uma *Trigger* que atua como *Assertion* relacionada a limitar o uso de cupons de 30% ou mais em feriados prolongados também foi criada. Feriados com dois dias ou mais não permitem tal modo de utilização dos cupons.

A última *Trigger* é reponsável pela checagem de cancelamento de um reserva realizada pelo usuário, o que automaticamente dispara uma ação para remover um cupom associado a tal usuário.

5 Funcionalidades e Limitações

As funcionalidades presentes resumem-se a cadastros, possibilitando a inserção de dados no banco para as tabelas correspondentes a usuário, hotel, reserva, quarto, feriado e cupom.

Foram implementados também algumas consultas para geração de relatórios, cuja principais funções são a apresentação dos resultados de consultas de interesse da administração do sistema. É importante citar que tais consultas só operam a nível de banco, ou seja, estas não estão integradas à interface do sistema. São exibidas algumas destas consultas nas Figuras 5 e 5.

Como principais limitações, o sistema não implementa a navegação em ofertas sem que exista um *login* ativo. Também não foram implementadas exclusões e as interações são realizadas de forma manual, além da ausência de um conjunto extenso de regras de negócio, principalmente relacionados as reservas.

```sql
-- Receita total de cada franquia de hotel
SELECT H.nome, SUM(R.valor_total) AS Receita
FROM reserva AS R INNER JOIN hotel AS H ON R.hotel_cod_hotel = H.cod_hotel
WHERE (R.data_inicial BETWEEN <variable_data_x> AND <variable_data_y>) AND R.estado <> 'Cancelado';
GROUP BY H.nome
ORDER BY H.nome ASC;

-- Receita total "perdida" em descontos
SELECT H.nome, SUM(((R.valor_total * 100) / (100 - valor_desconto)) - R.valor_total) AS Desconto
FROM reserva AS R INNER JOIN hotel AS H ON R.hotel_cod_hotel = H.cod_hotel
WHERE (R.data_inicial BETWEEN <variable_data_x> AND <variable_data_y>) AND R.estado <> 'Cancelado';
GROUP BY H.nome
ORDER BY H.nome ASC;

-- Total de reservas "em uso" ou reservadas (por franquia de hotel)
SELECT H.nome, COUNT(R.cod_reserva) AS Pending
FROM reserva AS R INNER JOIN hotel AS H ON R.hotel_cod_hotel = H.cod_hotel
WHERE (R.estado = 'Hospedado' OR R.estado = 'Reservado') AND (R.data_inicial BETWEEN <variable_data_x> AND <variable_data_y>)
GROUP BY H.nome
ORDER BY H.nome ASC;

-- 5 datas mais atrativas (Entrada - Todos)
SELECT data_inicial, COUNT(data_inicial) AS Occurrences
FROM reserva
GROUP BY data_inicial
ORDER BY Occurrences DESC
LIMIT 5;

-- 3 datas mais atrativas (Entrada -  Cada hotel)
SELECT R.data_inicial, COUNT(R.data_inicial) AS Occurrences
FROM reserva AS R INNER JOIN hotel AS H ON R.hotel_cod_hotel = H.cod_hotel
GROUP BY H.nome
ORDER BY Occurrences DESC
LIMIT 3;
```

```sql
-- Total de reservas canceladas (por hotel)
SELECT H.nome, COUNT(R.estado) AS Cancelled_Reserves_
FROM reserva AS R INNER JOIN hotel AS H ON R.hotel_cod_hotel = H.cod_hotel
WHERE R.estado = 'Cancelado' AND (R.data_inicial BETWEEN <variable_data_x> AND <variable_data_y>)
GROUP BY H.nome
ORDER BY H.nome ASC;

-- Total de clientes (por hotel)
SELECT H.nome, L.local_hotel, COUNT(U.cpf) AS Total_clients
FROM ((hotel AS H INNER JOIN localizacao_hotel AS L ON L.hotel_cod_hotel = cod_hotel)
      INNER JOIN reserva AS R ON R.hotel_cod_hotel = H.cod_hotel) INNER JOIN usuario AS U ON R.usuario_cpf = U.cpf
GROUP BY H.nome
ORDER BY H.nome ASC;

-- Informações sobre reservas entre determinadas datas especificadas pelo cliente
SELECT data_inicial, data_final, valor_total AS Valor_pago
FROM reserva
WHERE (data_inicial BETWEEN <variable_data_x> AND <variable_data_y>) AND usuario_cpf = <variable_cpf>

-- Retorna o tipo de quarto mais reservado em cada hotel
SELECT H.nome, L.local_hotel, Q.tipo, COUNT(R.tipo) AS Occurrences
FROM ((quarto AS Q INNER JOIN hotel AS H ON Q.hotel_cod_hotel = H.cod_hotel)
      INNER JOIN locacalizacao_hotel AS L ON L.hotel_cod_hotel = H.cod_hotel) INNER JOIN reserva AS R ON R.hotel_cod_hotel = H.cod_hotel
GROUP BY H.nome, L.local_hotel
ORDER BY Occurrences DESC
LIMIT 1;

-- Retorna a proporção de quantos usuários cadastrados moram fora de MG
SELECT (100 - ((SELECT COUNT(cpf) FROM usuario WHERE endereco LIKE '%Minas Gerais%') / (SELECT COUNT(cpf) FROM usuario)) * 100) AS Residem_fora;
```

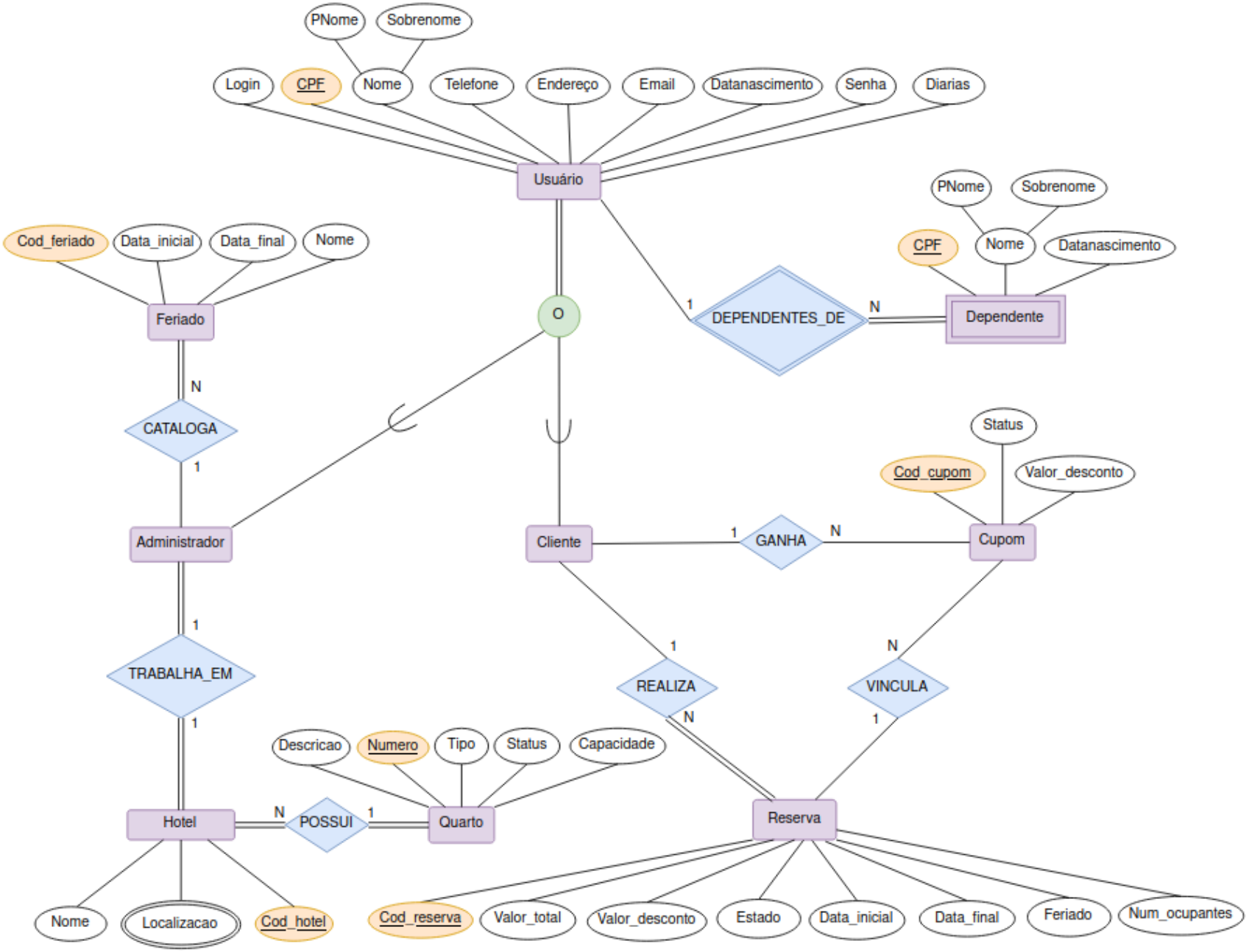
6 Conclusão

Neste trabalho, experienciamos na prática, como é lidar com as dificuldades de um sistema completo envolvendo Banco de Dados, desde o *Front-end* até o *Back end*, utilizando diferentes ferramentas relacionadas. Os estudos realizados foram muito agregadores, uma vez que ajudaram no desenvolvimento e na compreensão dos principais conceitos vistos na teoria ministrada em aula. Além disto, foi possível vivenciar as principais dificuldades e decisões para a construção e manutenção de um sistema como um todo, e compreender de forma superficial as atividades envolvidas no trabalho realizado por administradores de Banco de Dados.

Referências

- Sistemas de Banco de Dados - Navathe - 6ª edição
- Material disponível no portal didático da disciplina de Banco de Dados

Modelo Entidade-Relacionamento Estendido



Modelo Relacional