



Universidade do estado do Rio de  
Janeiro

## **TRABALHO DE MODELAGEM & IMPLEMENTAÇÃO BANCO DE DADOS**

**NOMES:** Christian Xavier Brandão,  
Jonathan Neves Alencar,  
Luis Felipe da Silva Souza

**PERÍODO:** 2019.2

**Rio de Janeiro**

**2019**

# MINIMUNDO:

## SISTEMA DE VIAGEM EMPRESARIAL

O sistema da empresa possui 3 tipos de solicitações, aérea, hospedagem e locação de carro. O funcionário poderá escolher diversas passagem, e pertencerá a um funcionário apenas. Funcionário possui PRONTUÁRIO, E-MAIL, SENHA E CPF que serão único, além de NOME, CARGO, PERFIL e SEXO que não são exclusivo.

Todos os funcionários pertencem a um departamento apenas. Departamento possui NOME, ORÇAMENTO, CENTRO de CUSTO e ID DO GERENTE (ID do gerente receberá o número de identificação do chefe de departamento).

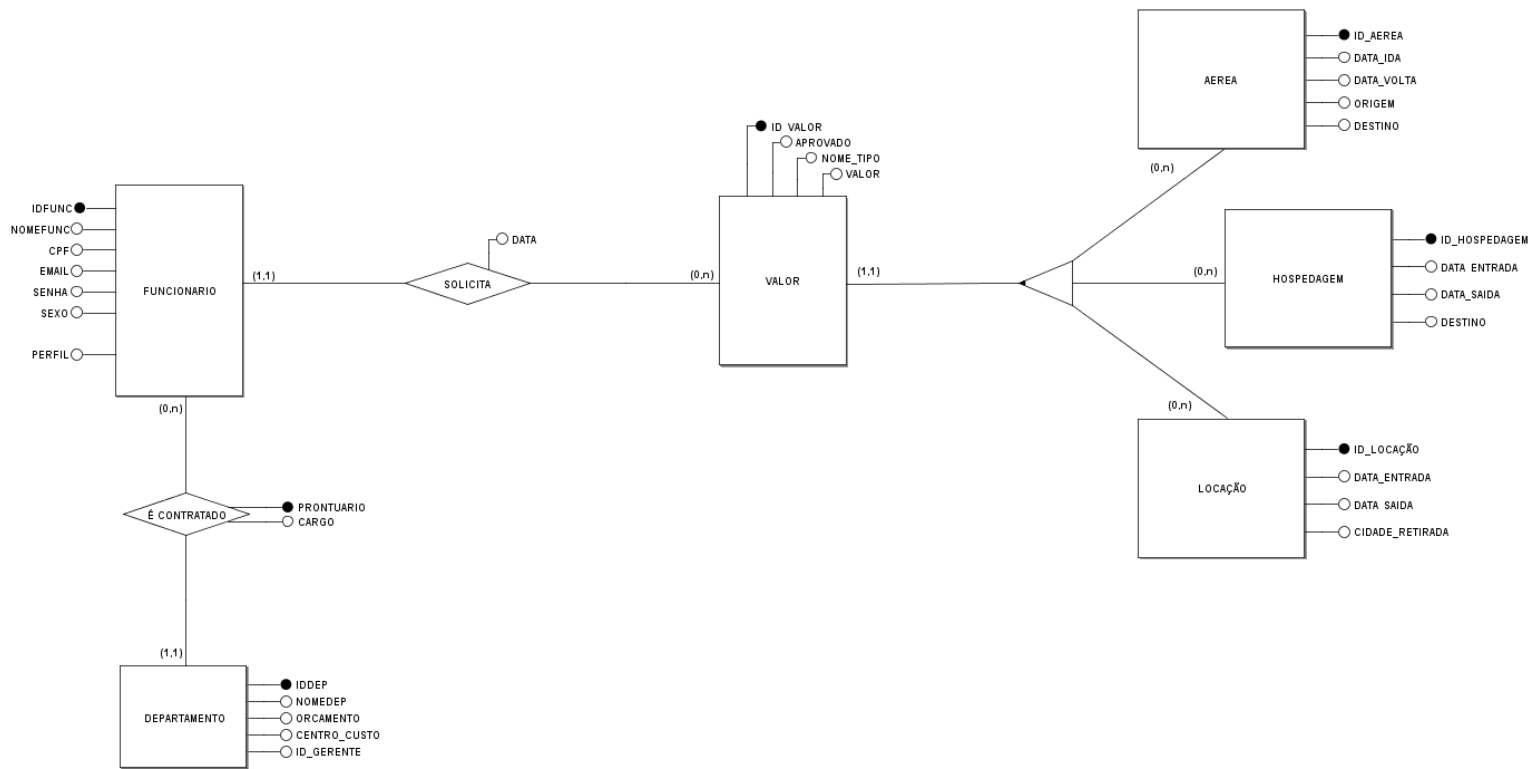
Todas as solicitações devem ser passadas por aprovação para serem emitidas. Após o funcionário solicitar a mesma fica em estado “pendente”, juntamente com valor da passagem, data da solicitação, nome da solicitação e “aprovado” que guardara o status da aprovação, caso seja aprovada a mesma ficara em estado “aprovado” ou se for rejeitada fica com estado “rejeitada”.

Se for solicitar aéreo deve ser informado data da ida, data da volta, origem e destino. Se for hospedagem, data da entrada, data da saída, destino do hotel. Se for locação, data da entrada, data da saída, cidade de retirada.

Cada tipo de passagem podem gerar diversos valores, entretanto, para cada valor pode haver somente um tipo de passagem.

Todos os usuários criados no sistema, entram com perfil “usuário” por default. O perfil pode ser alterado somente pelo administrador do banco de dados, podendo se tornar “Administrador” (Aprova as viagens de todos os departamentos) ou “Aprovador” (Em casos de chefes de departamento).

Cada passagem deve pertencer apenas a um funcionário, e o mesmo pode adquirir diversas passagens.



## MODELAGEM CONCEITUAL:

### Atributos:

**Funcionário** (IDFUNC, NOMEFUNC, CPF, EMAIL, EMAIL, SENHA, SEXO, PERFIL)

**Relação contratado** (PRONTUARIO, CARGO)

**Departamento** (IDDEP, NOMEDEP, ORCAMENTO, CENTRO\_CUSTO, ID\_GERENTE)

**Valor** (ID\_VALOR, APROVADO, NOME\_TIPO, VALOR)

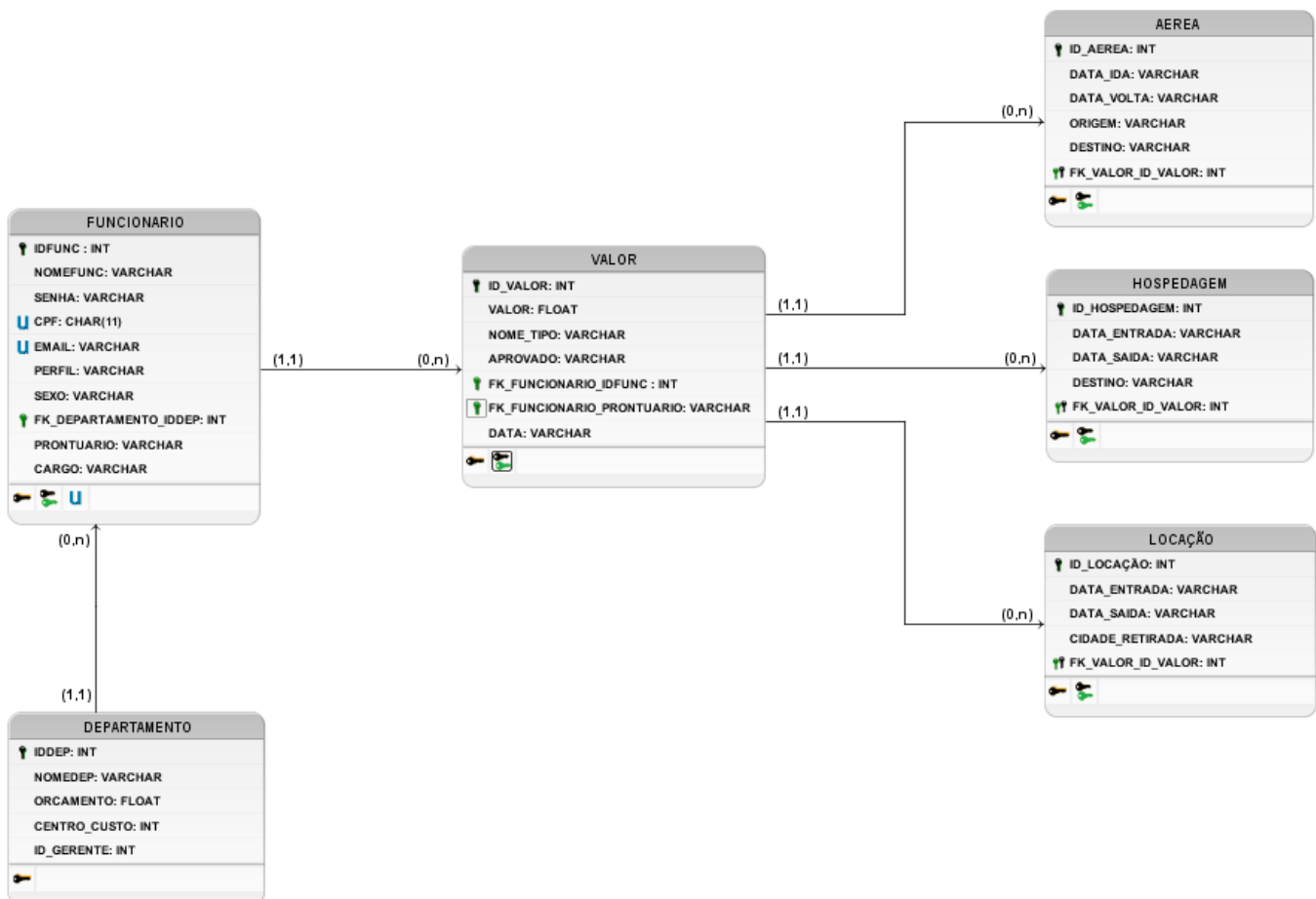
**Aérea** (ID\_AEREA, DATA\_IDA, DATA\_VOLTA, ORIGEM, DESTINO)

**Hospedagem** (ID\_HOSPEDAGEM, DATA\_ENTRADA, DATA\_SAIDA, DESTINO)

**Locação** (ID\_LOCACAO, DATA\_ENTRADA, DATA\_SAIDA, CIDADE\_RETIRADA)

**OBS:** chaves primárias grifadas.

## MODELAGEM LÓGICA:



Todas as entidades foram convertidas em tabelas. E nos relacionamentos  $(*,n) \rightarrow (*,1)$  foi colocada uma chave estrangeira na cardinalidade máxima  $(*,1)$ . Não houve outro tipo de conversão para o problema proposto.

### Chaves estrangeiras:

FK\_DEPARTAMENTO\_ID, FK\_FUNCIONÁRIO\_IDFUNC e  
FK\_VALOR\_ID\_VALOR

### Restrições de integridades:

- **Restrição de Chave**

Todas chaves primárias foram definidas como “primary key” o que garante que elas sejam únicas e não se repitam.

- **Restrição de Domínio**

Todos os campos foram definidos de acordo com seus respectivos conjuntos. Números foram usados inteiros e texto varchar. Ressaltamos que para datas também foi utilizado varchar e não date para melhor uso na linguagem Java.

- **Integridade de vazios**

Todos os campos foram definidos como “NOT NULL”, ou seja, não poderão receber nulo, exceto, o campo perfil na tabela funcionário, pois será definido posteriormente pelo ADM.

- **Integridade Referencial**

Todas as chaves estrangeiras foram tratadas na aplicação para que não viole a regra da integridade referencial, ou seja, foi tomado cuidado para que uma chave estrangeira de uma relação tenha que coincidir com uma chave primária da sua tabela “pai” a que a chave estrangeira se refere.

### RESTRIÇÕES SEMÂNTICAS:

- 1 - Todas as senhas devem ser maior que 8.
- 2 – Prontuário deve ter 6 dígitos.
- 3 - CPF deve ter 11 dígitos.
- 4 - Todas as datas de idas e voltas devem ser posterior à data atual.

### CONSULTAS EM ALGEBRA RELACIONAL

- 1 - Consulta de todas aprovações pendentes do perfil ADM

$$R1 \leftarrow (FUNCIONARIO \bowtie VALOR)$$
$$R2 \leftarrow \sigma (\text{aprovado} = \text{"pendente"})(R1)$$

$R43 \leftarrow \pi \text{id\_valor, nome\_func, nome\_tipo, valor}(R2)$

**2** – Consulta nome do funcionário e valor de todos os tipos de solicitações exceto hospedagem.

$R1 \leftarrow (\text{FUNCIONARIO} \bowtie \text{VALOR})$

$R2 \leftarrow \sigma(\text{nome\_tipo} = \text{"hospedagem"})(R1)$

$R3 \leftarrow (R1 - R2)$

$R4 \leftarrow \pi \text{nome, valor}(R3)$

**3** – Apresente o nome, valor, origem e destino das solicitações áreas que ficaram com valor acima de R\$ 1000.

$R1 \leftarrow (\text{FUNCIONARIO} \bowtie \text{VALOR})$

$R2 \leftarrow (R1 \bowtie \text{AEREA})$

$R3 \leftarrow \sigma(\text{VALOR.valor} > 1000)$

$R4 \leftarrow \pi \text{nome, valor, origem, destino}(R3)$

**4** – Liste todos os nomes dos funcionários valores e nome do tipo de solicitação para hospedagem e locação.

$R1 \leftarrow (\text{FUNCIONARIO} \bowtie \text{VALOR})$

$R2 \leftarrow (\sigma(\text{idfunc} = \text{id\_valor\_func})(R1))$

$R3 \leftarrow (\sigma(\text{nome\_tipo} = \text{'locação'})(R2))$

$R4 \leftarrow (\sigma(\text{nome\_tipo} = \text{'hospedagem'})(R2))$

$R5 \leftarrow (R3 \cup R4)$

$R6 \leftarrow \pi \text{nomefunc, valor, nome\_tipo}(R5)$

**5** – Apresente a média do valor gasto por todos os funcionários do departamento TI.

$R1 \leftarrow \sigma(\text{nomedep} = \text{'TI'})(\text{FUNCIONARIO})$

$R2 \leftarrow (R1 \times VALOR)$

$R3 \leftarrow \sigma(idfunc=id\_valor\_func)(R2)$

$R4 \leftarrow \gamma avg(valor)(R3)$

$R5 \leftarrow \pi valor(R4)$

## CONSULTAS EM SQL:

1 – Consultas número da solicitação, nome do funcionário, tipo de solicitação e valor de todas as solicitações pendentes para o gestor do departamento TI.

```
SELECT ID_VALOR, F.NOMEFUNC, NOME_TIPO, VALOR FROM VALOR  
JOIN FUNCIONARIO AS F  
ON IDFUNC = ID_VALOR_FUNC  
WHERE ID_DEP = 1 AND APROVADO = 'PENDENTE';
```

2 – Selecionar as mesmas especificações da consulta “1” para o administrador do sistema (observação aprova todas as pendências).

```
SELECT ID_VALOR, F.NOMEFUNC, NOME_TIPO, VALOR FROM VALOR  
JOIN FUNCIONARIO AS F  
ON IDFUNC = ID_VALOR_FUNC  
WHERE APROVADO = 'PENDENTE';
```

3 – Apresente o nome do departamento e o seu respectivo valor total gasto.

```
SELECT NOMEDEP, SUM(VALOR) FROM valor v  
JOIN funcionario  
ON ID_VALOR_FUNC = IDFUNC JOIN departamento d ON d.IDDEP = ID_DEP  
WHERE APROVADO = "APROVADO" GROUP BY ID_DEP ;
```

4 – Apresente o valor gasto por cada funcionário por ordem do maior para o menor.

```
SELECT DISTINCT(IDFUNC), SUM(VALOR) FROM valor  
JOIN funcionario  
ON ID_VALOR_FUNC = IDFUNC  
GROUP BY IDFUNC  
ORDER BY VALOR;
```

## SCRIPT DLL :

```
CREATE TABLE departamento(  
    IDDEP INT(11) PRIMARY KEY AUTO_INCREMENT,  
    NOMEDEP VARCHAR(50) NOT NULL,  
    ORCAMENTO INT(11) NOT NULL,  
    CENTRO_CUSTO INT(11) NOT NULL,  
    ID_GERENTE INT(11)  
);
```

```
CREATE TABLE funcionario(  
    IDFUNC INT(11) PRIMARY KEY AUTO_INCREMENT,  
    PRONTUARIO INT(11) UNIQUE NOT NULL,  
    NOMEFUNC VARCHAR(50) NOT NULL,  
    EMAIL VARCHAR(50) UNIQUE NOT NULL,  
    SENHA VARCHAR(100) NOT NULL,  
    CPF VARCHAR(11) UNIQUE NOT NULL,  
    ID_DEP INT(11) DEFAULT NULL,  
    CARGO VARCHAR(30) NOT NULL,  
    PERFIL VARCHAR(30) DEFAULT 'USUARIO',  
    SEXO ENUM('Masculino','Feminino') NOT NULL
```



```
CHECK(CHAR_LENGTH(SENHA) > 8),  
CHECK(CHAR_LENGTH(PRONTUARIO) = 6),  
CHECK(CHAR_LENGTH(CPF) = 11),  
CHECK(DEPARTAMENTO IN(SELECT NOMEDEP FROM DEPARTAMENTO))  
  
);
```

```
CREATE TABLE solicita_aerea(  
    ID_SOLICITA_AEREA INT(11) PRIMARY KEY AUTO_INCREMENT,  
    DATA_IDA VARCHAR(50) NOT NULL,  
    DATA_VOLTA VARCHAR(50) NOT NULL,  
    ORIGEM VARCHAR(50) NOT NULL,  
    DESTINO VARCHAR(50) NOT NULL,  
    ID_SOLICITAA_FUNC INT(11) NOT NULL,  
    ID_SOLICITA_VALOR INT(11) NOT NULL  
);
```

```
CREATE TABLE solicita_hospedagem(  
    ID_SOLICITA_HOSPEDAGEM INT(11) PRIMARY KEY AUTO_INCREMENT,  
    DATA_ENTRADA VARCHAR(50) NOT NULL,  
    DATA_SAIDA VARCHAR(50) NOT NULL,  
    DESTINO_HOTEL VARCHAR(50) NOT NULL,  
    ID_SOLICITAH_FUNC INT(11) NOT NULL,  
    ID_SOLICITA_VALOR INT(11) NOT NULL  
);
```

```
CREATE TABLE solicita_locacao(  
    ID_SOLICITA_LOCACAO INT(11) PRIMARY KEY AUTO_INCREMENT,  
    DATA_ENTRADA VARCHAR(50) NOT NULL,
```

```
DATA_SAIDA VARCHAR(50) NOT NULL,  
CIDADE_RETIRADA VARCHAR(50) NOT NULL,  
ID_SOLICITAL_FUNC INT(11) NOT NULL,  
ID_SOLICITA_VALOR INT(11) NOT NULL  
);
```

```
CREATE TABLE valor(  
ID_VALOR INT(11) PRIMARY KEY AUTO_INCREMENT,  
VALOR INT(11) NOT NULL,  
DATA_SOLICITACAO VARCHAR(50) NOT NULL,  
NOME_TIPO VARCHAR(50) NOT NULL,  
ID_VALOR_FUNC INT(11) NOT NULL,  
APROVADO VARCHAR(10) NOT NULL  
);
```

```
ALTER TABLE funcionario  
ADD CONSTRAINT FK_FUNC_DEP  
FOREIGN KEY (ID_DEP)  
REFERENCES departamento(IDDEP);
```

```
ALTER TABLE solicita_aerea  
ADD CONSTRAINT FK_AEREA_FUNC  
FOREIGN KEY (ID_SOLICITAA_FUNC)  
REFERENCES funcionario(IDFUNC);
```

```
ALTER TABLE solicita_hospedagem  
ADD CONSTRAINT FK_HOSPE_FUNC  
FOREIGN KEY (ID_SOLICITAH_FUNC)  
REFERENCES funcionario(IDFUNC);
```

```
ALTER TABLE solicita_locacao  
ADD CONSTRAINT FK_LOCAC_FUNC  
FOREIGN KEY (ID_SOLICITA_FUNC)  
REFERENCES funcionario(IDFUNC);
```

```
ALTER TABLE solicita_aerea  
ADD CONSTRAINT FK_AEREA_VAL  
FOREIGN KEY (ID_SOLICITA_VALOR)  
REFERENCES valor(IDVALOR);
```

```
ALTER TABLE solicita_hospedagem  
ADD CONSTRAINT FK_HOSPE_VAL  
FOREIGN KEY (ID_SOLICITA_VALOR)  
REFERENCES valor(IDVALOR);
```

```
ALTER TABLE solicita_locacao  
ADD CONSTRAINT FK_LOCAC_VAL  
FOREIGN KEY (ID_SOLICITA_VALOR)  
REFERENCES valor(IDVALOR);
```

## Forma Normal e Dependências funcionais

### Tabela departamento:

Dependência funcional: todos os atributos depende unicamente da chave primária.

1ª Não possuí atributos com valores multivalorados.

2ª Todos os atributos depende apenas da chave primária.

3ª A tabela não possui atributo não chave que dependa de outro atributo não chave

Portando a tabela está na 3ª forma normal.

## Tabela Funcionário:

Dependência funcional: todos os atributos depende unicamente da chave primária.

1ª Não possui atributos com valores multivalorados.

2ª Todos os atributos depende apenas da chave primária.

3ª A tabela não possui atributo não chave que dependa de outro atributo não chave

Portando a tabela está na 3ª forma normal.

## Tabela Valor:

Dependência funcional: Todos os atributos depende da chave primária, ressaltando que aprovado não depende do atributo valor, e sim, de uma escolha externa do seu gerente da área, portanto, depende também unicamente de sua chave primária.

1ª Não possui atributos com valores multivalorados.

2ª Todos os atributos depende apenas da chave primária.

3ª A tabela não possui atributo não chave que dependa de outro atributo não chave

Portando a tabela está na 3ª forma normal.