Universidade Federal de São Carlos
Ciência da Computação
Banco de Dados
Profa. Sahudy Montenegro

Projeto Integrado - Entrega Final

Tema 5 - Cadastro de outros funcionários do aeroporto

Grupo 11
Isabela Salmeron - 552593
Luciane Lopes - 552348
Renan Rossignatti de França - 489697
Ricardo Mendes - 562262

Data de entrega: 02 de dezembro de 2016

SOROCABA
2016

ÍNDICE

1. Especificação do problema

1.1 Objetivo do sistema

O sistema deve gerenciar um cadastro dos funcionários do aeroporto com exceção de técnicos especialistas e controladores de voo. O sistema deve possuir uma interface onde possa ser realizado o cadastro e alterações destes funcionários.

O sistema deverá armazenar informações dos funcionários do aeroporto, funcionários terceirizados, e das empresas que fornecem os serviços. Também será possível alterar os dados dos funcionários e colocá-los como Inativo no sistema, caso eles deixem de trabalhar no aeroporto.

O sistema também deverá gerar um relatório contendo a quantidade de funcionários que estão ativos em cada função, dado um determinado período de trabalho, e um outro relatório contendo a quantidade de funcionários que trabalham em cada terminal do aeroporto, dada uma determinada função.

Também será possível realizar buscas de funcionários dado seu nome, CPF, cargo, terminal, cidade ou uma faixa salarial.

1.2 Especificação

Para o cadastro de funcionários do aeroporto, é necessário armazenar um código único de identificação, nome, CPF, telefone, data de nascimento, endereço (rua, número, bairro, cidade, estado e CEP). Informações sobre o contrato, como a data de entrada, horas trabalhadas semanalmente, salário, setor (Financeiro, Limpeza, Segurança, Informações) e terminal em que trabalha (A, B, C, D, E), período (Manhã, Tarde, Noite) e o seu cargo (Comum/Gerente). O campo estado do funcionário é inserido no banco como ATIVO.

Para o cadastro de funcionários terceirizados, temos os mesmos campos do funcionário do aeroporto, com exceção da data de entrada, horas trabalhadas semanalmente, salário e o cargo. Também será necessário armazenar o CNPJ da empresa que está fornecendo o serviço.

Para cadastrar empresas que fornecem serviços terceirizados para o aeroporto. Para isso será necessário armazenar dados como o CNPJ, o seu nome e razão social.

Para a visualização de funcionários, o usuário poderá realizar diferentes consultas utilizando como chaves o nome do funcionário, CPF, cargo, terminal ou cidade. Informações como código, nome, CPF, terminal, período e estado, serão exibidas. Ao realizar a busca será possível alterar qualquer dado do funcionário selecionado, com exceção do seu código, e também alterar seu estado para INATIVO/ATIVO caso ele pare ou volte a trabalhar no aeroporto, respectivamente.

Será possível solicitar a emissão de três tipos de relatórios, sendo eles:

- Relatório por Função: onde o usuário informa um período (Manhã, Tarde ou Noite) e o sistema retorna a quantidade de funcionários ativos em cada

função no determinado período, e também dados de cada funcionário como o código, nome, CPF, terminal e período.

- Relatório por Terminal: onde o usuário informa um cargo (Comum ou Gerente) e o sistema retorna a quantidade de funcionários ativos em cada terminal do aeroporto no determinado cargo, e também dados de cada funcionário como o código, nome, CPF e período.
- Relatório por Faixa Salarial: onde o usuário informa uma faixa salarial e o sistema retorna a quantidade de funcionários ativos que possuem o salário dentro dessa faixa, e também dados de cada funcionário como o código, nome, CPF e período.

2. Elicitação de Requisitos

Para o desenvolvimento do projeto foram levantados os requisitos do sistema, apresentados abaixo.

2.1 Requisitos Funcionais

Requisitos funcionais são os requisitos que estão diretamente ligados a funcionalidade do sistema, dizendo como ele deve reagir a entradas específicas e como se comportar em determinadas situações.

Nesta seção, são apresentados os requisitos funcionais do sistema.

RF01: O sistema deverá manter o cadastro de funcionários.
RF02: O sistema deverá emitir relatório contendo a quantidade de funcionários ativos para cada função em um determinado período de tempo, e também nome, código, CPF, terminal e período de cada funcionário.
RF03: O sistema deverá emitir relatório contendo a quantidade de funcionários ativos em cada terminal do aeroporto em uma determinada função, e também nome, código, CPF, terminal e período de cada funcionário.
RF04: Em caso de fim de vínculo com um dado funcionário, o sistema deve manter registro da causa de desvinculação e alterar seu estado para inativo.
RF05: O sistema deve verificar se funcionário já está cadastrado antes de efetivar o cadastro, caso o funcionário já esteja cadastrado, o sistema deve informar o usuário.
RF06: O sistema deverá permitir a consulta de funcionários através de seu nome, CPF, cargo, terminal e cidade, mostrar dados como nome, código, CPF, terminal, período e estado.
RF07: O sistema deverá retornar todos os funcionários ativos dado uma faixa salarial, mostrar dados como nome, código, CPF, terminal, período e estado.

2.2 Requisitos Não Funcionais

Requisitos não-funcionais são os requisitos relacionados ao uso da aplicação em termos de desempenho, portabilidade, disponibilidade, usabilidade, segurança, etc. Eles expressam as qualidades e restrições do sistema.

Abaixo são apresentados os requisitos não-funcionais do sistema.

RNF01: O sistema deve encontrar um dado funcionário em 0,5 segundo.
RNF02: O sistema deve ser implementado em PHP.
RNF03: O sistema deve ser compatível com os navegadores de internet “Google Chrome” e “Mozilla Firefox”
RNF04: O sistema deve utilizar o sistema de gerenciamento de banco de dados MySQL.

3. Requisitos de Dados

Nesta seção, são apresentados os requisitos de dados, que especificam quais consultas o sistema deverá ser capaz de fazer.

RD01: O sistema terá cadastro de funcionários contendo os seguintes campos: código, nome, CPF, código do terminal, cargo, período da jornada de trabalho, quantidade de horas trabalhadas por dia, data de entrada no aeroporto, estado (vinculado/desvinculado), motivo pela desvinculação (no caso de demissão), data de fim do vínculo com o aeroporto, telefone, data de nascimento, endereço.
RD02: O sistema permitirá consulta por funcionário (ativo/inativo) pelos seguintes campos: CPF, nome, terminal, cargo, cidade e faixa salarial.
RD03: O sistema permitirá a modificação no cadastro de funcionários em todos os campos, com exceção do código do funcionário.
RD04: O sistema permitirá a visualização da quantidade e dos funcionários de cada terminal.
RD05: O sistema permitirá a visualização da quantidade e dos funcionários que trabalham em cada período (manhã/tarde/noite).
RD06: O sistema utilizará informações sobre os terminais do aeroporto: nome, código e localização.
RD07: O sistema permitirá a consulta de funcionários ativos pelo campo salário.

4. Matriz de Associação dos Requisitos de Dados (RF x RD)

Nesta seção é apresentada a matriz (*Tabela 1*) que representa a associação dos Requisitos Funcionais com os Requisitos de Dados do sistema.

	RD01	RD02	RD03	RD04	RD05	RD06	RD07
RF01	X						
RF02		X		X	X	X	
RF03		X		X		X	
RF04			X				
RF05	X						
RF06		X					
RF07							X

Tabela 1 - Matriz de Associação dos Requisitos de Dados

5. Projeto Conceitual

5.1 Modelo Entidade Relacionamento

O Modelo Entidade Relacionamento (MER) é um modelo de dados usado para descrever, de maneira conceitual, os dados que serão utilizados no sistema. Seus principais componentes são as entidades e seus relacionamentos.

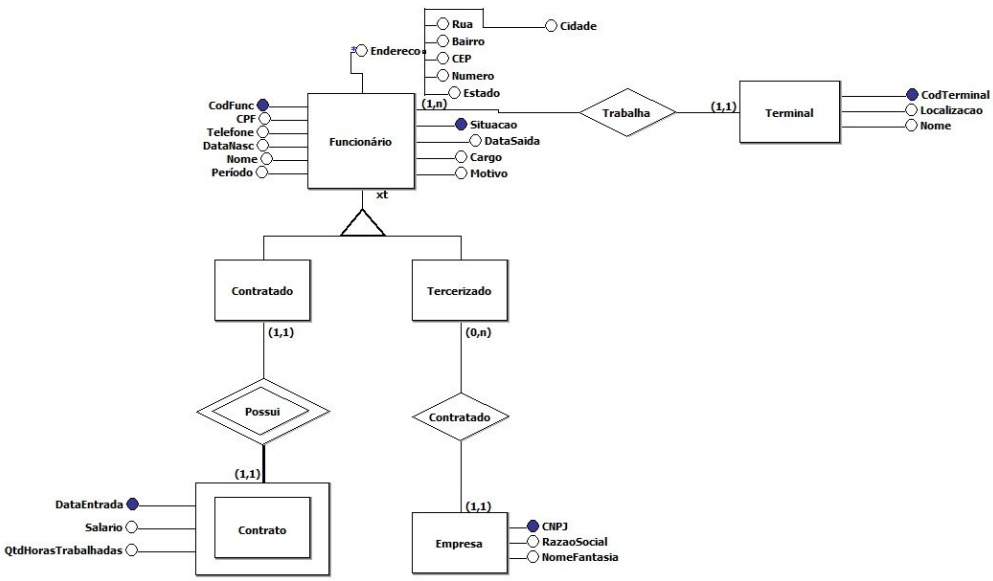

Figura 1 - Modelo Entidade-Relacionamento

5.2 Descrição do tipo de atributos por entidade

A seguir é mostrado quais atributos, e suas descrições de tipo, que cada entidade do projeto apresenta.

Tipo entidade	Atributo	Tipo
Funcionário	`CodFunc`	identificador
	`Nome`	monovalorado, armazenado, simples, obrigatório
	`CPF`	monovalorado, armazenado, simples, obrigatório e 123.456.789-10
	`End`	multivalorado
	`DataNasc`	monovalorado, armazenado, simples, obrigatório e dd/mm/aaaa
	`Tel`	monovalorado, armazenado, simples, obrigatório e (xx) nnnn-nnnn
	`Período`	monovalorado, armazenado, simples, obrigatório
	`Cargo`	monovalorado, armazenado, simples, obrigatório e comum ou gerente
	`Motivo`	monovalorado, armazenado, simples
	`Estado`	monovalorado, armazenado, simples, obrigatório e ativo/inativo
	`DataSaida`	monovalorado, armazenado, simples e > DataEntr

Tabela 2 - Descrição dos tipos de atributos do tipo entidade Funcionário

Tipo entidade	Atributo	Tipo
Terminal	`CodTerminal`	identificador
	`Nome`	monovalorado, armazenado, simples, obrigatório
	`Localizacao`	monovalorado, armazenado, simples, obrigatório

Tabela 3 - Descrição dos tipos de atributos do tipo entidade Terminal

Tipo entidade	Atributo	Tipo
Contrato	`DataEntr`	identificador
	`Salario`	monovalorado, armazenado, simples, obrigatório
	`QtddHorasTrabalhadas`	monovalorado, armazenado, simples, obrigatório e > 0

Tabela 4 - Descrição dos tipos de atributos do tipo entidade Contrato

Tipo atributo	Valores	Tipo
`Endereco`	`Rua`	monovalorado, armazenado, simples, obrigatório
	`CEP`	monovalorado, armazenado, simples, obrigatório e nnnnn-nnn
	`Numero`	monovalorado, armazenado, simples, obrigatório

	Bairro	monovalorado, armazenado, simples, obrigatório
	Cidade	monovalorado, armazenado, simples, obrigatório

Tabela 5 - Descrição dos tipos de atributos do campo Endereco

Tipo entidade	Atributo	Tipo
Empresa	CNPJ	identificador
	RazaoSocial	monovalorado, armazenado, simples, obrigatório
	NomeFantasia	monovalorado, armazenado, simples, obrigatório

Tabela 6 - Descrição dos tipos de atributos da entidade Empresa

6. Projeto Lógico

A seguir é apresentada a identificação do conjunto de relações que especificam o banco de dados relacional a ser implementado. Para a transição entre o Modelo Entidade-Relacionamento e o Modelo Relacional foi utilizada a ferramenta BrModelo.

A Figura 2 mostra o Modelo Relacional do projeto:

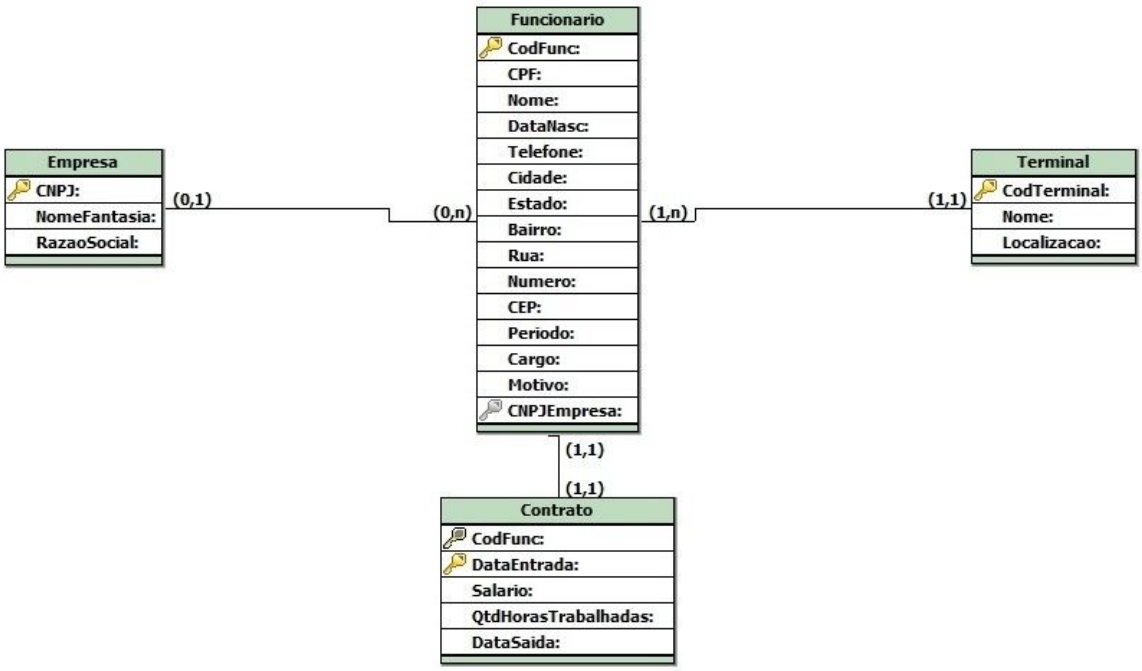

Figura 2 - Modelo Lógico

7. Especificação de consultas em Álgebra Relacional e SQL

As consultas descritas abaixo foram utilizadas no projeto e foram pensadas de forma a facilitar e especificar seus usos na aplicação.

7.1 Consulta 1

Esta consulta recupera o código do funcionário, nome, CPF, terminal, período e estado, dado um certo nome de funcionário (`<nomeFunc>`).

Álgebra Relacional:

$$\pi_{codigo\_funcionario,\ nome,\ cpf,\ cidade,\ terminal,\ periodo,\ estado}(\sigma_{nome\ =\ <nomeFunc>}(funcionario))$$

SQL:

```
SELECT codigo_funcionario, nome, cpf, cidade, terminal,
periodo, estado
FROM funcionario
WHERE nome LIKE "%<nomeFunc>%" ORDER BY nome
```

7.2 Consulta 2

Esta consulta recupera o código do funcionário, nome, CPF, terminal, período e estado dado, um certo CPF do funcionário (<cpfFunc>).

Álgebra Relacional:

$$\pi_{codigo\_funcionario,\ nome,\ cpf,\ cidade,\ terminal,\ periodo,\ estado}(\sigma_{cpf = <cpfFunc>}(funcionario))$$

SQL:

```
SELECT codigo_funcionario, nome, cpf, cidade, terminal,
periodo, estado
FROM funcionario
WHERE cpf LIKE "<cpfFunc>"
```

7.3 Consulta 3

Esta consulta recupera o código do funcionário, nome, CPF, terminal, período e estado, dado um certo cargo do funcionário (<cargoFunc>).

Álgebra Relacional:

$$\pi_{codigo\_funcionario,\ nome,\ cpf,\ cidade,\ terminal,\ periodo,\ estado}(\sigma_{cargo = <cargoFunc>}(funcionario))$$

SQL:

```
SELECT codigo_funcionario, nome, cpf, cidade, terminal,
periodo, estado
FROM funcionario
WHERE cargo LIKE "<cargoFunc>" ORDER BY nome
```

7.4 Consulta 4

Esta consulta recupera o código do funcionário, nome, CPF, terminal, período e estado, dado um certo terminal (<terminalFunc>).

Álgebra Relacional:

$$\pi_{codigo\_funcionario,\ nome,\ cpf,\ cidade,\ terminal,\ periodo,\ estado}(\sigma_{terminal = <terminalFunc>}(funcionario)$$

SQL:

```
SELECT codigo_funcionario, nome, cpf, cidade, terminal,
periodo, estado
FROM funcionario
```

```
WHERE terminal LIKE "<terminalFunc>" ORDER BY nome
```

7.5 Consulta 5

Esta consulta recupera o código do funcionário, nome, CPF, terminal, período e estado, dado uma certa cidade (<cidadeFunc>).

Álgebra Relacional:

$\pi_{codigo\_funcionario,\ nome,\ cpf,\ cidade,\ terminal,\ periodo,\ estado,\ cidade}(\sigma_{cidade\ =\ <cidadeFunc>}(funcionario))$

SQL:

```
SELECT codigo_funcionario, nome, cpf, cidade, terminal,
periodo, estado, cidade
FROM funcionario
WHERE cidade LIKE "%<cidadeFunc>%" ORDER BY cidade
```

7.6 Consulta 6

Esta consulta recupera o código do funcionário, nome, CPF, terminal e estado de funcionários ativos, dado um determinado período (<periodoFunc>).

Álgebra Relacional:

$\pi_{codigo\_funcionario,\ nome,\ cpf,\ terminal,\ estado}(\sigma_{estado\ =\ <ATIVO>\ \ AND\ \ setor\ =\ <setor>\ AND\ periodo\ =\ <periodo>}(funcionario))$

SQL:

```
SELECT cargo, periodo, COUNT(*) AS quantidade
FROM funcionario
WHERE funcionario.situacao LIKE "ATIVO" AND
funcionario.periodo = "MANHA" ORDER BY nome
```

7.7 Consulta 7

Esta consulta recupera o código do funcionário, nome, CPF, terminal e estado de funcionários ativos, dado um determinado setor (<setorFunc>).

Álgebra Relacional:

$\pi_{codigo\_funcionario,\ nome,\ cpf,\ terminal,\ periodo,\ estado}(\sigma_{estado\ =\ <ATIVO>\ AND\ setor\ =\ <setorFunc>}(funcionario))$

SQL:

```
SELECT f.terminal, f.cargo, COUNT(f.terminal) AS quantidade
FROM funcionario f WHERE f.situacao LIKE "ATIVO" AND f.cargo LIKE "COMUM" GROUP BY f.terminal
```

8. Considerações Finais

Tivemos uma grande dificuldade em elaborar o projeto como um todo, inicialmente o projeto estava bem simples e com poucas coisas no front end, porém com o desenvolvimento e as idéias para o front end o projeto foi se tornando mais complexo a cada etapa.

Uma grande dificuldade foi fazer o back end utilizando um modelo MVC mais modesto, fazer a manipulação dos dados em várias etapas do software foi a maior dificuldade, pois tínhamos idéias para mostrar os dados de uma forma mais clara para o usuário no front end e não conseguimos manipular bem os dados no modelo.

O projeto foi desafiador, mostrando que podemos manipular várias tecnologias ao mesmo tempo, como javascript, jquery, ajax, php, html e css, tudo em um único projeto.

Uma outra grande dificuldade foi utilizar a linguagem PHP, pois os membros do grupo não tinham um vasto conhecimento da linguagem, e foi necessário fazer várias pausas para pesquisar como eram feitas as operações e manipulações dos dados no software, tanto para se conectar com o banco até fazer leituras no front end.

Acreditamos que a experiência com o projeto foi produtiva embora tivemos uma grande dificuldade em vários pontos.