

CURSO DE PÓS-GRADUAÇÃO LATO SENSU

EM DESENVOLVIMENTO WEB

**Equipe:**

Diogo Meira Trindade  
Matheus Coqueiro Andrade   
Yuri Rodrigues Santos Silva

**PROJETO DE BANCO DE DADOS:**

Sistema de Chamados

Vitória da Conquista-BA

Outubro/2019

**Equipe:**

Diogo Meira Trindade  
Matheus Coqueiro Andrade   
Yuri Rodrigues Santos Silva

**PROJETO DE BANCO DE DADOS:**

Sistema de Chamados

Trabalho apresentado ao Curso de Pós-Graduação Lato Sensu em Desenvolvimento Web do Instituto Federal de Educação, Ciência e Tecnologia da Bahia *campus* Vitória da Conquista como requisito parcial para a aprovação na disciplina Banco de Dados para a Web.

Professor Orientador: Me. Pablo Freire Matos

Vitória da Conquista-BA

Outubro/2019

**HISTÓRICO DE PARTICIPAÇÃO**

|  |  |  |
| --- | --- | --- |
| **Período** | **Participante** | **Função** |
| 28/09/2019 - 19/10/2019 | Matheus | Desenvolvimento da API |
| Diogo | Análise do Banco e Conversão para não relacional |
| Yuri | Desenvolvimento da Aplicação |

**RESUMO**

Pretende‐se neste projeto abordar conceitos que norteiam a construção de um banco de dados para a unidade hospitalar, Andro, situada na cidade de Vitória da Conquista - BA. Devido à grande demanda de serviços prestados, é muito comum que os aparelhos tecnológicos: Tanto os de informática, quanto os hospitalares, precisam de manutenção. Partindo dessa necessidade, a empresa pretende implantar um sistema no setor de TI para controlar os chamados originados por defeitos ou mal funcionamento dos equipamentos, bem como dar o suporte ao usuário. Toda vez que um equipamento precisar de reparo, ou o funcionário dê um auxílio nas execuções de suas tarefas, este, irá abrir um chamado para que o problema seja solucionado por um técnico responsável. No banco de dados serão registrados o tipo do problema, a data e o horário de ocorrência, os dados dos funcionários que necessitaram do serviço e também de quem executou a tarefa, e os registros caso o serviço precisou de algum produto que fosse substituído (Memória, Fonte). A princípio foi realizado a análise de requisitos, elaboração do minimundo e com base nele foi desenvolvido o diagrama de entidade-relacionamento do banco de dados do sistema.

**Palavras-chave:** Banco de Dados. Necessidade. Controlar. Chamado. Esquema Conceitual.

**SUMÁRIO**

[**1**](#_1fob9te) **INTRODUÇÃO 6**

[**2**](#_2et92p0) **PROJETO DE BANCO DE DADOS RELACIONAL 9**

[2.1](#_tyjcwt) Descrição do Minimundo 9

[2.1.1](#_3dy6vkm) Abertura e Finalização de Chamados 9

[2.1.2](#_1t3h5sf) Agregação na Utilização de Produtos [10](#_4d34og8)

[2.1.3](#_4d34og8) Definição de Prioridades 10

[2.2](#_2s8eyo1) Esquema Conceitual 10

[2.3](#_17dp8vu) Esquema Lógico 13

[2.4](#_3rdcrjn) Esquema Físico 14

[2.4.1](#_26in1rg) Povoamento das Tabelas 14

[2.4.2](#_lnxbz9) Consultas em MySQL 15

[**3**](#_1ksv4uv) **PROJETO DE BANCO DE DADOS NÃO RELACIONAL 18**

[3.1](#_44sinio) Modelo de Dados 18

[3.2](#_2jxsxqh) Povoamento dos Dados 19

[3.3](#_z337ya) Consultas 19

[**4**](#_3j2qqm3) **INTEGRAÇÃO DO BD NÃO RELACIONAL COM APLICAÇÃO WEB 23**

[4.1](#_1y810tw) Mockup 23

[4.2](#_4i7ojhp) Sistema 23

[**5**](#_2xcytpi) **CONCLUSÃO 26**

[**REFERÊNCIAS 27**](#_1ci93xb)

[**APÊNDICE A – ARQUIVOS BANCO DE DADOS RELACIONAL 28**](#_3whwml4)

[**APÊNDICE B – ARQUIVOS BANCO DE DADOS NÃO RELACIONAL 29**](#_2bn6wsx)

# INTRODUÇÃO

Com o passar dos anos, o uso de tecnologias cresceu de forma exponencial, qualquer organização independente do seu porte ou área possui máquinas que são fundamentais para o bom funcionamento de todo o processo. Logo, quando ocorre um defeito ou mal funcionamento, todo o processo fica parado, e para que esses problemas possam ser resolvidos há necessidade de um setor de TI (Tecnologia da Informação) que é um dos mais requisitados dentro de uma empresa.

O setor de TI identifica três momentos para a resolução de problemas. O primeiro é o momento em que o setor é notificado do problema que pode ser de diversos tipos, desde um suporte ao usuário que não sabe utilizar alguma ferramenta até a troca de peças. É importante que no ato da notificação sejam passadas informações detalhadas do que está acontecendo, qual o setor e o usuário para que possa ser solucionado de forma eficiente.

O segundo momento é, após análise, identificar o que está causando erro na máquina e iniciar a manutenção, e a terceira e última é aplicar a correção que vai sempre depender do tipo do problema. Visto que, é de grande importância a resolução de forma rápida e eficaz para não atrasar o rendimento da empresa, surge a necessidade de um sistema que controle essas adversidades. Este sistema facilitará a comunicação entre o setor de TI e os demais setores e também fará um registro do que pode-se denominar de chamados, que são as notificações dos problemas.

**Caracterização da Empresa**

O projeto do banco de dados que a ser desenvolvido será utilizado pelo hospital urológico Andro, localizado na Avenida Otávio Santos, nº 444, Bairro Recreio na cidade de Vitória da Conquista – BA. Inicialmente eram prestados apenas atendimentos urológicos, mas com o crescimento da demanda e a solicitação dos pacientes pela ampliação dos serviços e especialidades fez com que surgissem mais.

Então, atualmente oferecem 30 especialidades em uma unidade física de aproximadamente 2.130 m2 distribuídos em subsolo, térreo, primeiro e segundo andares com 4 consultórios, 16 leitos de internamento mais 6 leitos ambulatoriais e pouco mais de 100 colaboradores diretos e atende a uma média de 4.300 pacientes por mês.

O banco de dados será implementado no setor de TI. Conforme citado previamente, há uma grande necessidade de implantar um sistema que controle os chamados, de modo a tornar o processo mais rápido e eficiente, registrando e fornecendo informações sobre cada chamado. O setor de TI do hospital atende a vários tipos de serviços, suporte ao usuário, manutenção de computadores, impressoras, câmeras de segurança e equipamentos hospitalares tanto o hardware quanto o software. Logo Na Tabela 1 o total de computadores do hospital, na qual é possível notar que possui uma grande necessidade de um sistema para controlar os chamados, visto que o hospital possui muitas máquinas.

**Tabela 1 -** Total de computadores do hospital Andro.

|  |  |
| --- | --- |
| **TOTAL DE COMPUTADORES POR SETOR** | |
| **TOTAL DE**  **COMPUTADOR** | **SETOR** |
| 2 | ADMINISTRAÇÃO |
|
| 1 | ARQUIVO |
| 2 | ESTAGIÁRIO |
| 1 | AUDITORIA PRONTUÁRIO |
| 2 | AUTORIZAÇÃO |
|
| 1 | AUDITÓRIO |
| 2 | BIOIMAGEM |
| 5 | CENTRO CIRÚRGICO |
| 1 | COLONOSCOPIA |
| 1 | COMERCIAL |
| 3 | CONTABILIDADE \ RH |
|
| 1 | ELETROCARDIOGRAMA |
| 1 | COORDENAÇÃO ENFERMAGEM |
| 2 | FARMÁCIA |
|
| 6 | FATURAMENTO |
|
| 3 | FINANCEIRO |
|
| 3 | INTERNAMENTO |
| 1 | LABORATÓRIO |
| 1 | LECO |
| 1 | MANUTENÇÃO |
| 1 | HOTELARIA |
| 1 | MARKETING QUALIDADE |
| 4 | MÉDICOS |
|
| 2 | PABX |
|
| 1 | PORTARIA |
| 3 | POSTO DE ENFERMAGEM |
| 4 | RECEPÇÃO |
|
| 1 | MARCAÇÃO |
| 1 | SND |
| 1 | ULTRASSOM |
| 1 | URODINÂMICA |
| 1 | UROFLUXO |
| 2 | TI |
|
| 6 | SERVIDORES |
|
| **69** | |

# PROJETO DE BANCO DE DADOS RELACIONAL

## Descrição do Minimundo

O hospital Andro presta diversos serviços aos seus pacientes, e para atender sua demanda possui muitas máquinas hospitalares e de informática, que precisam estar sempre em bom funcionamento para que os procedimentos sejam executados corretamente. Visto que, há uma grande demanda para o setor de TI que é responsável por gerir as ocorrências de erros com os equipamentos tecnológicos, surgiu a necessidade de um sistema que controle os chamados.

Este sistema deverá conter matrícula e nome dos funcionários que identificarão suas ações dentro do sistema, e cada funcionário deve pertencer a um setor com código do setor e seu respectivo nome. Terá uma divisão de grupo dos funcionários, onde os que pertencem ao setor da TI serão responsáveis por resolver os chamados, e os demais poderão abrir um chamado. Do chamado é importante saber sua identificação, status (Aberto, Cancelado, Resolvido) e descrição do problema e registrar data e hora da abertura e da resolução do chamado.

Caso para resolver o chamado precise trocar alguma peça/acessório (Memória, Fonte), deverá constar no banco de dados a utilização desse produto possuindo identificador, nome, descrição e quantidade, e cada produto pertence a um tipo que deverá conter um identificador, quantidade e o nome do tipo.

Para cada chamado será estipulado um tipo de serviço (Manutenção do ECG/MAPA, Manutenção de Computador), os tipos já são determinados na Tabela 2 e cada tipo de serviço terá um nível de prioridade que é definida conforme explicação na seção 2.3. Sobre o tipo de serviço é importante saber o código do serviço, e seu nome e a prioridade.

### Abertura e Finalização de Chamados

Quando o funcionário logar no sistema com seu usuário e senha será verificado suas permissões. Caso seja do setor de TI terá permissão total, e se for de outros setores terá permissão apenas para abrir o chamado e nesta abertura terá um campo para descrever detalhadamente o problema e o status ficará automaticamente em aberto. E para os funcionários da TI também terá um campo para descrição da solução e terão acesso para alterar o status do chamado para resolvido ou cancelado quando for o caso. O sistema será programado para no ato da abertura/resolução capturar data e hora atual para manter registros dos chamados.

### Agregação na Utilização de Produtos

Os chamados podem ser resolvidos com ou sem a necessidade de trocar peças ou acessórios. Logo, para resolver essa questão utilizamos uma agregação na relação cuja cardinalidade é (NxN) no ato da solução do problema. Caso seja necessário fazer algum reparo físico onde terá uma relação de utilização de produtos, e esses produtos terão um tipo específico determinado na própria base de dados.

### Definição de Prioridades

As prioridades são estipuladas pelo tipo de serviço de acordo com o critério de que quanto mais próximo do cliente maior será sua prioridade, que pode ser classificada como Alta, Média ou Baixa. Dessa forma é possível determinar as reais necessidades visto que, o hospital presa pela qualidade e agilidade dos seus serviços. Segue na Tabela 2 a lista de prioridade do hospital Andro, conforme os tipos de serviços. Se houver necessidades de um novo tipo de serviço, o mesmo deve ser informado ao setor de TI para que seja feita a inclusão.

**Tabela 2 - Prioridades dos Serviços**

|  |  |
| --- | --- |
| **TIPO DE SERVIÇO** | **NÍVEL DE PRIORIDADE DO SERVIÇO** |
|
| ALTERAR CPS | Baixa |
| AVISO DE LENTIDÃO\TRAVAMENTO | Alta |
| DESTRAVAR SAÍDA | Baixa |
| GERENCIAR USUÁRIO | Média |
| MANUTENÇÃO DE CÂMERAS | Média |
| MANUTENÇÃO DE COMPUTADOR | Alta |
| MANUTENÇÃO DE EMAIL | Média |
| MANUTENÇÃO DE IMPRESSORA/SCANNER | Alta |
| MANUTENÇÃO DE INTERNET | Alta |
| MANUTENÇÃO DE OUTROS EQUIPAMENTOS | Baixa |
| MANUTENÇÃO DE OUTROS PROGRAMAS | Baixa |
| MANUTENÇÃO DE TELEFONE/RAMAL | Baixa |
| MANUTENÇÃO DO CALL CENTER | Alta |
| MANUTENÇÃO DO ECG/MAPA | Alta |
| MANUTENÇÃO DO SISTEMA CPC | Alta |
| MANUTENÇÃO DO SISTEMA DE PONTO | Alta |
| MANUTENÇÃO DO SISTEMA DO SUS | Baixa |
| MANUTENÇÃO DO SISTEMA DOMÍNIO | Alta |
| MANUTENÇÃO UROFLUXO/URODINÂMICA | Alta |
| RETIRAR ALTA | Baixa |
| SOLICITAÇÃO DE DESBLOQUEIOS | Baixa |
| SOLICITAÇÃO DE EQUIPAMENTOS | Baixa |
| SUPORTE AO USUÁRIO | Baixa |
| TROCA DE TONER | Alta |
| UNIFICAR CADASTRO | Baixa |

## Esquema Conceitual

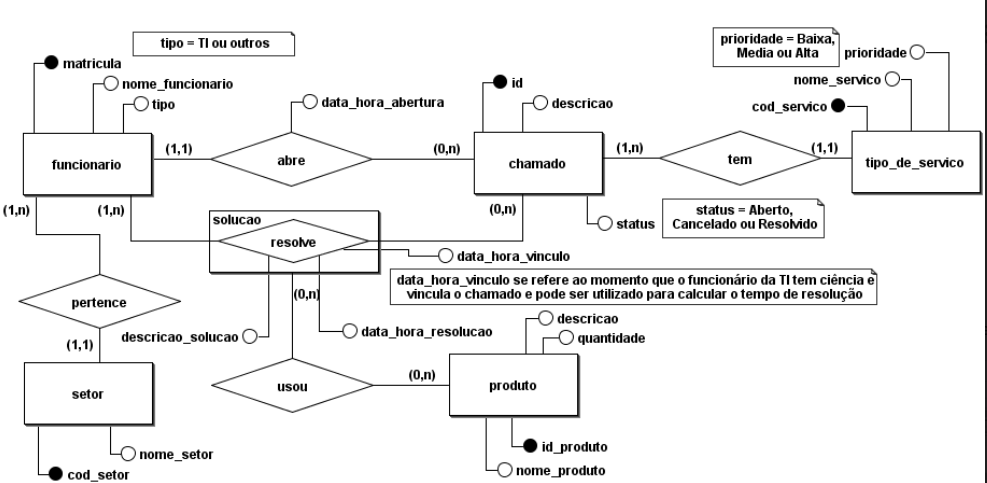
Na Figura 1 é mostrado o Diagrama Entidade-Relacionamento do Banco de Dados do Sistema de Chamados do Hospital Andro antigo, conforme minimundo descrito no Capítulo 2. Foram elaborados todos os relacionamentos entre as entidades e inseridos os atributos, conforme a necessidade e o problema da empresa, de modo a facilitar a compreensão já que essa é a característica principal deste modelo. Na Figura 2 é mostrado as alterações realizadas no esquema conceitual visto que um chamado poderia ser resolvido mais de uma vez por mais de um funcionário de TI a relação foi modificada para 1:N, além de adicionar a quantidade de produtos utilizados para resolver o chamado.

**Consultas**

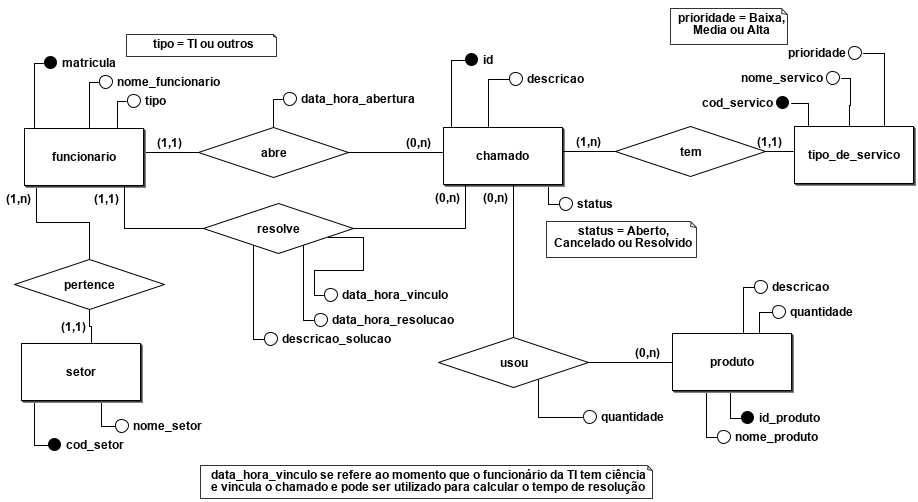
Estando já com o diagrama de entidade-relacionamento pronto, pode definir algumas possíveis consultas a serem realizada no banco de dados:

1. Quantos chamados um determinado funcionário da TI resolveu em um período.
2. Listar todos os chamados de um determinado tipo de serviço.
3. Quais chamados foram abertos por um determinado setor no período “x”.
4. Qual o funcionário da TI resolveu o chamado aberto pelo funcionário “x”.
5. Quais produtos foram utilizados para resolver um chamado.
6. Qual a quantidade de um determinado tipo de produto no estoque.
7. Quais chamados foram atendidos, mas ainda não foram resolvidos.
8. Listar os chamados por data e hora de abertura/resolução.
9. Qual o funcionário abriu o chamado “y”.
10. Listar os chamados resolvidos que não utilizou produtos em um determinado tempo
11. Qual o tempo médio de resolução dos chamados com o tipo de serviço “x”

**Figura 1 - Diagrama Conceitual Antigo**



**Figura 2 - Diagrama Conceitual novo**

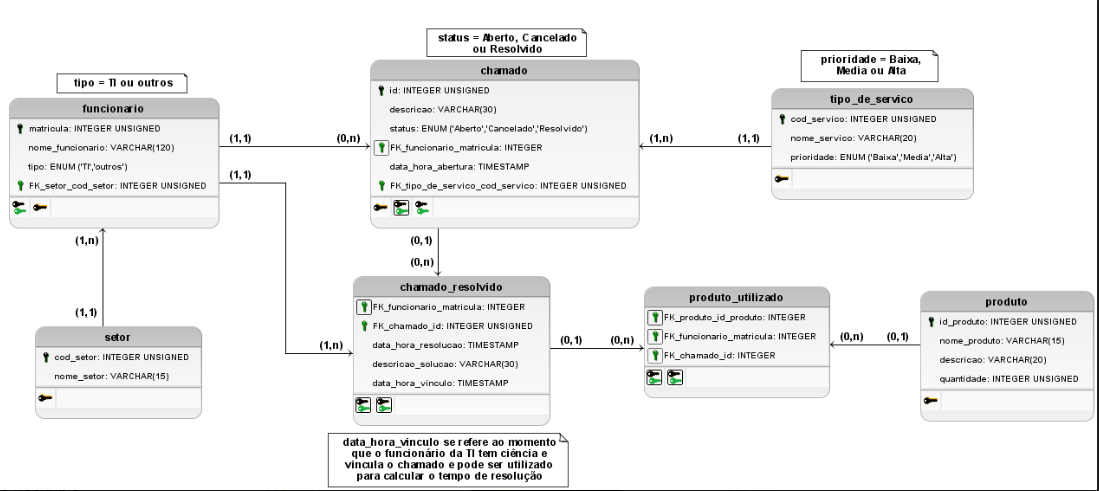


## Esquema Lógico

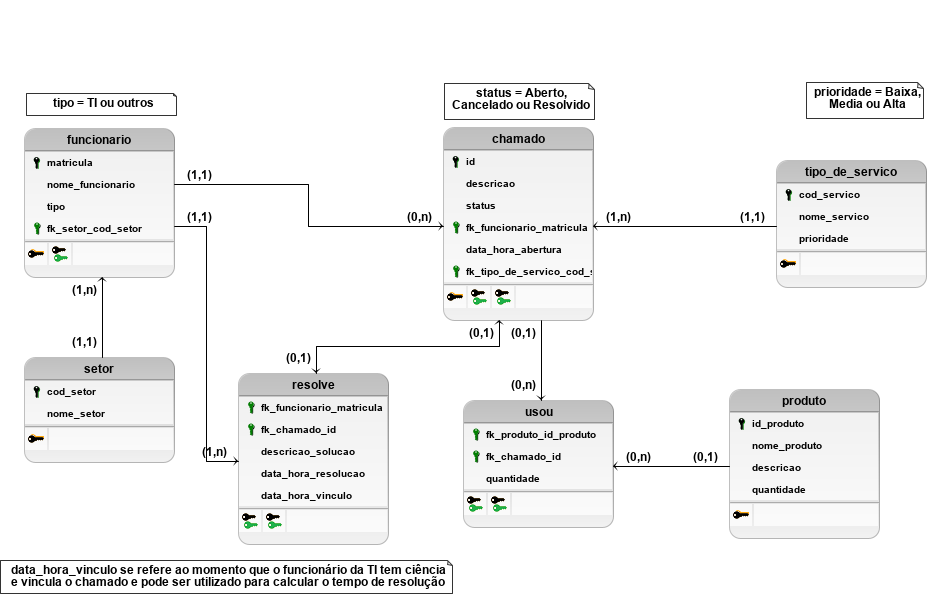
É a representação do esquema conceitual ER (Entidade-Relacionamento) do banco de dados em um modelo lógico de dados, e neste projeto será utilizado o modelo relacional que segundo (SILBERSCHATZ, 2002) “Usa um conjunto de tabelas para representar tanto os dados como a relação entre eles. Cada tabela possui múltiplas colunas e cada uma possui um nome único”.

Para isso, é realizado o mapeamento das relações existentes no esquema conceitual para o modelo lógico escolhido. Tornando-se um nível intermediário, uma ponte entre a camada de maior abstração (conceitual) e a de menor abstração (física). Nas figuras 3 e 4 temos o modelo físico antigo e o novo modelo físico após as alterações do projeto.

**Figura 3 - Esquema Lógico antigo**



**Figura 4 - Esquema físico novo**

****

## Esquema Físico

Um esquema de banco de dados físico define como os dados são armazenados fisicamente em um sistema de armazenamento. Esse modelo depende do SGBD que está sendo usado. A partir daqui, são detalhados os componentes da estrutura física do banco, como tabelas, campos, tipos de valores, índices, dentre outros. Para o desenvolvimento deste projeto, utilizou-se o MySQL SGBD, que utiliza a linguagem SQL (*Structured Query Language*). A partir do esquema lógico apresentado na Figura 2 foi gerado o esquema físico, desenvolvendo assim o banco de dados do mesmo e das tabelas que compõe. Para o desenvolvimento do esquema físico foi utilizado o programa brModelo (CÂNDIDO, 2005).

### Povoamento das Tabelas

A partir da criação das tabelas do banco é possível a inserção dos dados que serão guardados para possíveis consultas posteriores. Os dados inseridos são apenas exemplos de como é possível povoar as tabelas.

### Consultas em MySQL

Após a criação do banco de dados e povoamento das respectivas tabelas, é possível a visualização das consultas que serão utilizadas no banco, a saber:

1. Quantos chamados um determinado funcionário da TI resolveu em um período

SELECT COUNT(\*) quantidade

FROM chamado\_resolvido

WHERE chamado\_resolvido.data\_hora\_resolucao BETWEEN '2018-07-11 08:50:17' AND '2018-08-04 12:45:47'

AND chamado\_resolvido.funcionario\_matricula = '20180010';

2. Listar todos os chamados de um determinado tipo de serviço.

SELECT chamado.id,

chamado.descricao,

chamado.status,

chamado.funcionario\_matricula,

chamado.data\_hora\_abertura,

tipo\_de\_servico.nome,

tipo\_de\_servico.prioridade

FROM chamado

JOIN tipo\_de\_servico ON chamado.servico\_codigo = tipo\_de\_servico.codigo

WHERE tipo\_de\_servico.nome LIKE 'TROCA DE TONER';

3. Quais chamados foram abertos por um determinado setor no período “x”.

SELECT chamado.id,

chamado.descricao,

chamado.status,

chamado.funcionario\_matricula,

chamado.data\_hora\_abertura,

chamado.servico\_codigo

FROM chamado

JOIN funcionario ON funcionario.matricula = chamado.funcionario\_matricula

JOIN setor ON setor.codigo = funcionario.setor\_codigo

WHERE setor.nome LIKE 'CONTABIL. RH'

AND chamado.data\_hora\_abertura BETWEEN '2018-08-04 12:30:45' AND ' 2018-08-10 12:30:45';

4. Qual o funcionário da TI resolveu o chamado aberto pelo funcionário “x”.

SELECT chamado\_resolvido.funcionario\_matricula,

chamado.id

FROM chamado

JOIN chamado\_resolvido ON chamado.id = chamado\_resolvido.chamado\_id

WHERE chamado.funcionario\_matricula = '20180001';

5. Quais produtos foram utilizados para resolver um chamado.

SELECT chamado\_resolvido.chamado\_id,

produto.nome,

produto\_utilizado.quantidade

FROM chamado\_resolvido

JOIN produto\_utilizado ON produto\_utilizado.chamado\_resolvido\_id = chamado\_resolvido.id

JOIN produto ON produto.id = produto\_utilizado.produto\_id

WHERE chamado\_resolvido.chamado\_id = 6;

6. Qual a quantidade de um determinado tipo de produto no estoque.

SELECT produto.nome,

SUM(produto.quantidade) quantidade

FROM produto

WHERE produto.nome LIKE 'HD'

GROUP BY produto.nome;

7. Quais chamados foram atendidos, mas ainda não foram resolvidos.

SELECT \*

FROM chamado

JOIN chamado\_resolvido ON chamado\_resolvido.chamado\_id = chamado.id

WHERE chamado.status = 'ABERTO';

8. Listar os chamados por data e hora de abertura/resolução.

SELECT \*

FROM chamado

JOIN chamado\_resolvido ON chamado\_resolvido.chamado\_id = chamado.id

ORDER BY chamado.data\_hora\_abertura, chamado\_resolvido.data\_hora\_resolucao;

9. Qual o funcionário abriu o chamado “y”.

SELECT funcionario.matricula,

funcionario.nome

FROM chamado

JOIN funcionario ON funcionario.matricula = chamado.funcionario\_matricula

WHERE chamado.id = 1;

10. Listar os chamados resolvidos que não utilizou produtos em um determinado tempo.

SELECT \*

FROM chamado\_resolvido

WHERE chamado\_resolvido.id NOT IN (SELECT produto\_utilizado.chamado\_resolvido\_id FROM produto\_utilizado);

11. Qual o tempo médio de resolução dos chamados com o tipo de serviço “x”.

SELECT SEC\_TO\_TIME(

AVG(

TIMESTAMPDIFF(

SECOND,

chamado.data\_hora\_abertura,

chamado\_resolvido.data\_hora\_resolucao

)

)

) MEDIA

FROM chamado

JOIN chamado\_resolvido ON chamado\_resolvido.chamado\_id = chamado.id

JOIN tipo\_de\_servico ON tipo\_de\_servico.codigo = chamado.servico\_codigo

WHERE chamado.status = 'RESOLVIDO'

AND chamado.servico\_codigo = 6;

# PROJETO DE BANCO DE DADOS NÃO RELACIONAL

## Modelo de Dados

* + 1. Funcionários

**"funcionarios"** : {

**"\_id"** : "ObjectId",

**"matricula"** : "Number",

**"nome"** : "String",

**"tipo"** : "String",

**"senha"** : "String",

**"setor"** : "String"

}

* + 1. Produtos

**"produtos"** : {

**"\_id"** : "ObjectId",

**"nome"** : "String",

**"descricao"** : "String",

**"quantidade"** : "Number"

}

* + 1. Chamados

**"chamados"** : {

**"\_id"** : "ObjectId",

**"descricao"** : "String",

**"status"** : "String",

**"funcionario\_id"** : "ObjectId",

**"data\_hora\_abertura"** : "Date",

**"tipo\_servico"** : {

**"\_id"** : "ObjectId",

**"nome"** : "String",

**"prioridade"** : "String"

},

**"resolvido"** : {

**"\_id"** : "ObjectId",

**"funcionario\_id"** : "ObjectId",

**"data\_hora\_vinculo"** : "Date",

**"data\_hora\_finalizado"** : "Date",

**"descricao"** : "String",

**"produtos\_utilizados"** : [

{

**"\_id"** : "ObjectId",

**"produto\_id"** : "ObjectId",

**"quantidade"** : "Number"

}

]

}

}

## Povoamento dos Dados

Como os arquivos para povoamento do banco são grandes, foi disponibilizado no Apêndice B os links para os mesmos.

## Consultas

* + 1. Quantos chamados um determinado funcionário da TI resolveu em um período.

db.chamados.find({

$and:[

{"resolvido.data\_hora\_finalizado":{

$gte: ISODate("2019-10-01T00:00:00.000Z")

}},

{"resolvido.data\_hora\_finalizado":{

$lte: ISODate("2019-10-24T00:00:00.000Z")

}},

{"resolvido.funcionario\_id": ObjectId("5dab43503db2734f40519377")}

]

}).count();

* + 1. Listar todos os chamados de um determinado tipo de serviço.

db.chamados.aggregate([

{$match: {"tipo\_servico.nome": "MANUT. DE EMAIL"}},

{$project :{ resolvido: 0 }}

]);

* + 1. Quais chamados foram abertos por um determinado setor no período “x”.

db.chamados.aggregate([

{$lookup: {

from: "funcionarios",

localField: "funcionario\_id",

foreignField: "\_id",

as: "funcionario"

}},

{$match:

{$and:[

{"funcionario.setor":

{$eq: "CONTABIL. RH"}

},

{"data\_hora\_abertura":

{$gte: ISODate("2019-10-19T17:09:12.000Z")}

},

{"data\_hora\_abertura":

{$lte: ISODate("2019-10-19T17:10:43.000Z")}

},

]}

}

]);

* + 1. Qual o funcionário da TI resolveu o chamado aberto pelo funcionário “x”.

db.chamados.aggregate([

{$match: {"funcionario\_id": ObjectId("5dab43603db2734f40519378")}},

{$project: {"resolvido.funcionario\_id": 1, "chamado.\_id": 1}}

]);

* + 1. Quais produtos foram utilizados para resolver um chamado.

db.chamados.aggregate([

{$match: {"\_id": ObjectId("5dab436f3db2734f40519379")}},

{$project: {"produtos": "$resolvido.produtosUtilizados"}},

{$lookup: {

from: "produtos",

localField: "produtos.produto\_id",

foreignField: "\_id",

as: "produto"

}},

{$project: {

"produtos": "$produto.nome",

"quantidade":"$produtos.quantidade"

}},

]);

* + 1. Qual a quantidade de um determinado tipo de produto no estoque.

db.produtos.aggregate([

{$match: {"nome":"PC"}},

{$project: {nome: 1, quantidade: 1}}

]);

* + 1. Quais chamados foram atendidos, mas ainda não foram resolvidos.

db.chamados.aggregate([

{$match: {$and:[

{"resolvido.funcionario\_id":{$exists: true}},

{"status":'Aberto'}

]}}

]);

* + 1. Listar os chamados por data e hora de abertura/resolução.

db.chamados.aggregate([{

$lookup: {

from: "chamado\_resolvido",

localField: "\_id",

foreignField: "chamado\_id",

as: "chamados\_resolvidos"

}}, {

$group: {

\_id:{

abertura: "$data\_hora\_abertura",

resolucao: "$resolvido.data\_hora\_finalizado"

},

count: { $sum: 1 },

chamados: {$push:{

\_id: "$\_id",

descricao: "$descricao",

funcionario\_id: "$funcionario\_id",

status: "$status",

tipo\_servico: "$tipo\_servico",

}}

}

}]);

* + 1. Qual o funcionário abriu o chamado “y”.

db.chamados.aggregate([{

$lookup: {

from: "funcionarios",

localField: "funcionario\_id",

foreignField: "\_id",

as: "funcionario"

}},

{$match:{\_id: ObjectId("5dab436f3db2734f40519379")}},

{$project:{\_id: 0, id:"$funcionario.\_id", nome:"$funcionario.nome"}}

]);

* + 1. Listar os chamados resolvidos que não utilizou produtos em um determinado tempo.

db.chamados.aggregate([

{$match: {$and:[

{"resolvido.produtosUtilizados.0":{$exists: false}},

{"status":'Resolvido'},

{"resolvido.data\_hora\_finalizado": {

$gte: ISODate("2019-10-19T17:26:29.000Z")

}},

{"resolvido.data\_hora\_finalizado": {

$lte: ISODate("2019-10-19T18:21:53.000Z")

}}

]}}

]);

* + 1. Qual o tempo médio de resolução dos chamados com o tipo de serviço “x”.

db.chamados.aggregate([

{$match: {$and:[

{"status":"Resolvido"},

{"tipo\_servico.nome": "MANUT. DE EMAIL"}

]}}, {

$group: {

\_id: "\_id",

media:{ $avg: { $multiply: [

{$sum:{$subtract:[

"$resolvido.data\_hora\_finalizado",

"$data\_hora\_abertura"

]}},

{$sum: 1}

]}}}

}]);

# INTEGRAÇÃO DO BD NÃO RELACIONAL COM APLICAÇÃO WEB

## Mockup

Devido o curto espaço de tempo para desenvolvimento da aplicação, optou-se por não desenvolver alguns artefatos necessários no processo de desenvolvimento de aplicações, por isso na documentação não foram incluídos alguns destes artefatos.

## Sistema

* + 1. Descrição

O sistema de chamados desenvolvido pela equipe é composto de duas partes, a primeira uma aplicação híbrida responsável pela apresentação e interface com o cliente, e a segunda uma API responsável por receber, tratar e responder as requisições realizadas pela aplicação.

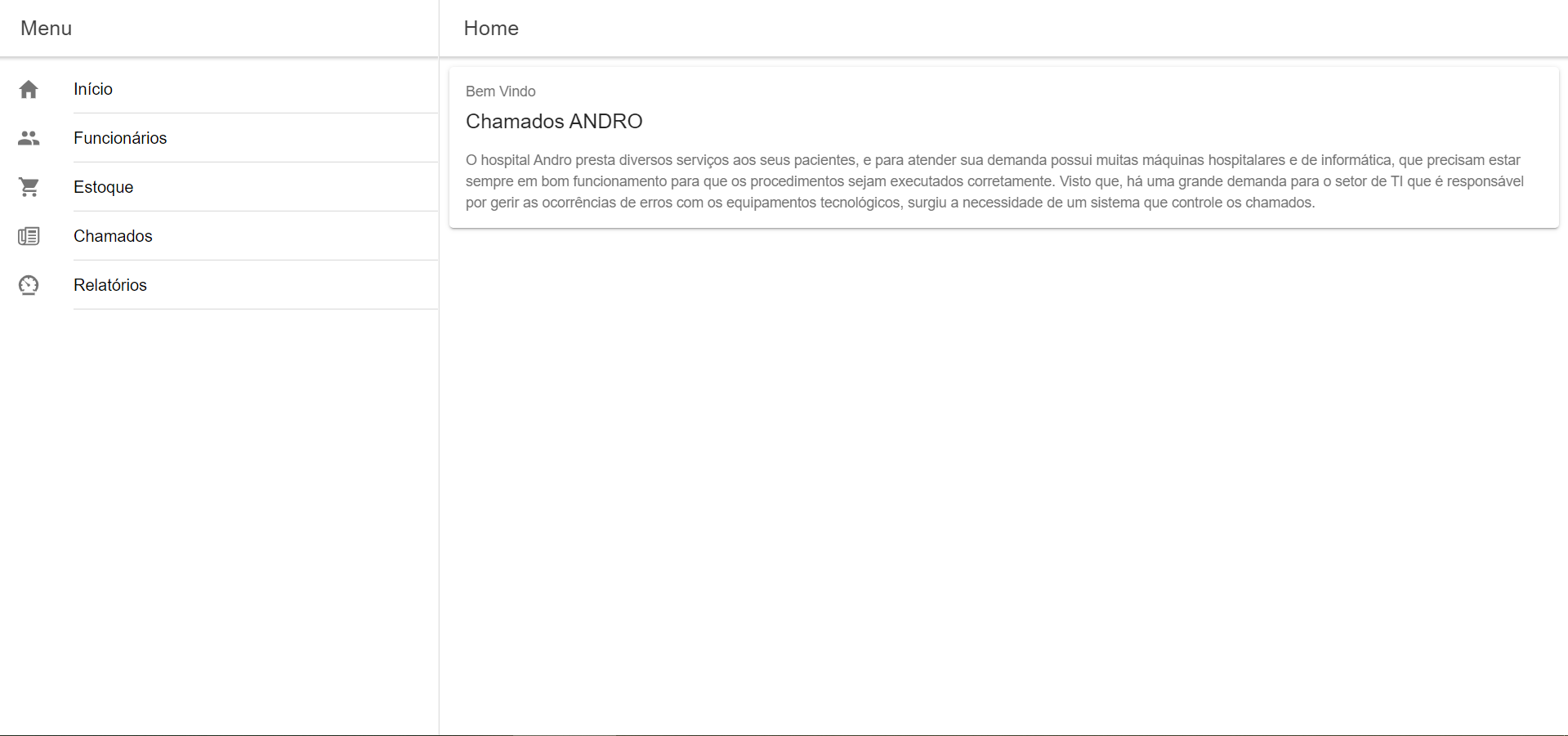
O aplicativo foi desenvolvido utilizando o framework Ionic, que foi escolhido pela facilidade no desenvolvimento de aplicações multi-plataformas, e da estabilidade do framework.

Já a API foi desenvolvida utilizando o Node.js, um interpretador de javascript no lado do servidor, foi escolhido o Node.js por conta da alta integração com banco não-relacionais, especialmente o MongoDB.

* + 1. A Aplicação

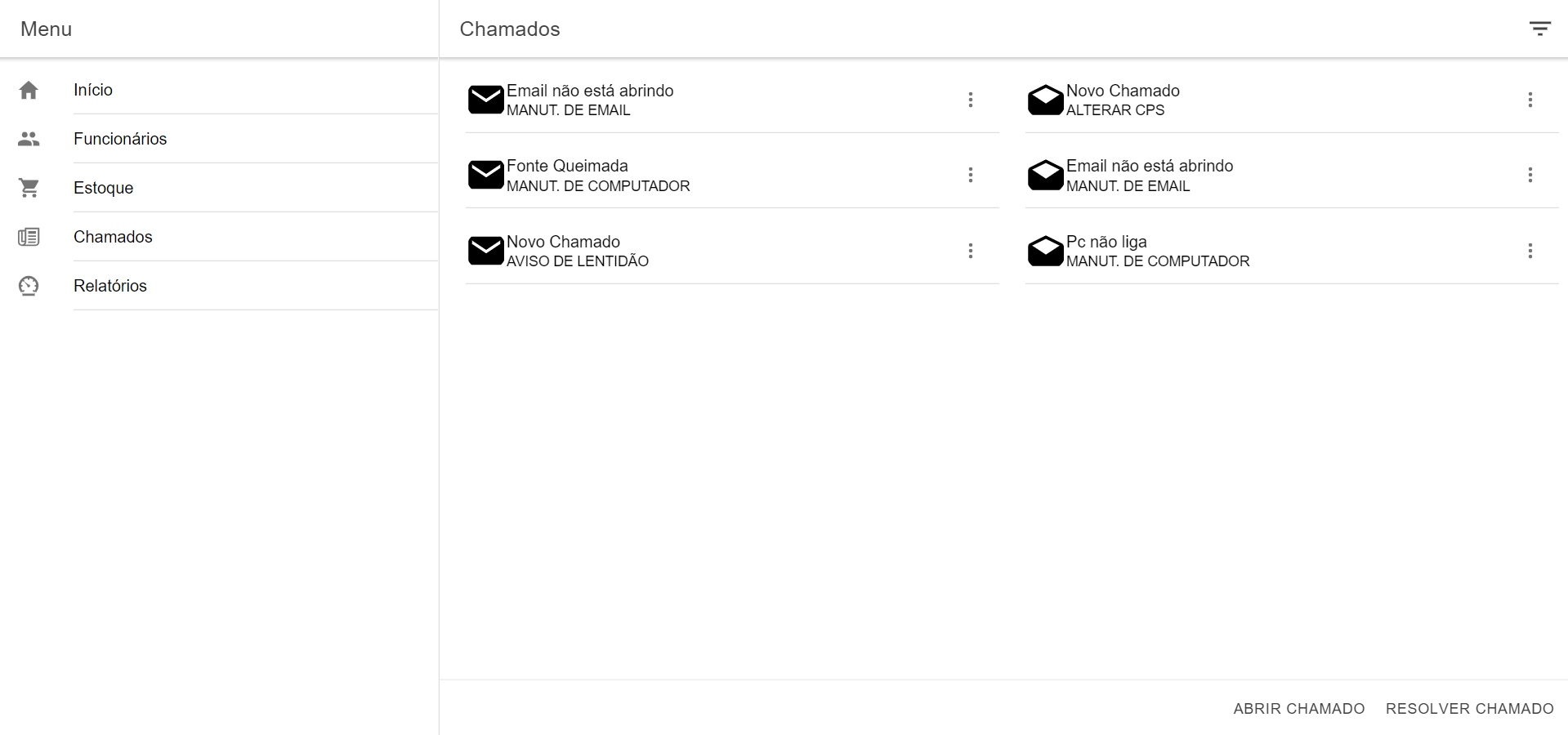
A figura 5 apresenta a tela inicial do sistema, nela consta um menu lateral com opções de navegação que será exibido em todas as telas, e apresenta também uma descrição do objetivo do sistema.

**Figura 5 - Tela Inicial da Aplicação**



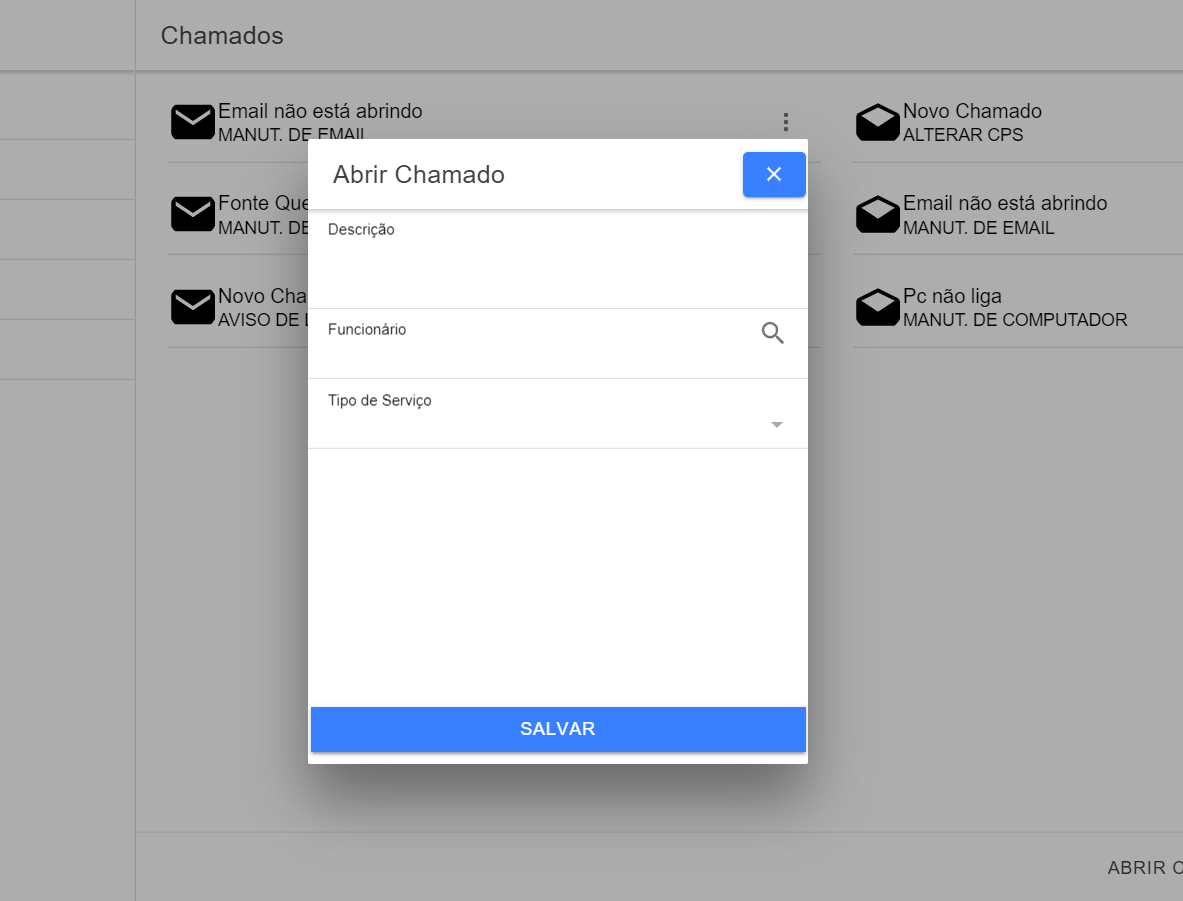
A figura 6 apresenta a tela de chamados, nessa tela é exibido a lista de chamados cadastrados, e no canto inferior direito da tela estão as opções para criar um novo chamado e resolver um chamado.

**Figura 6 - Tela de Listagem de Chamados**



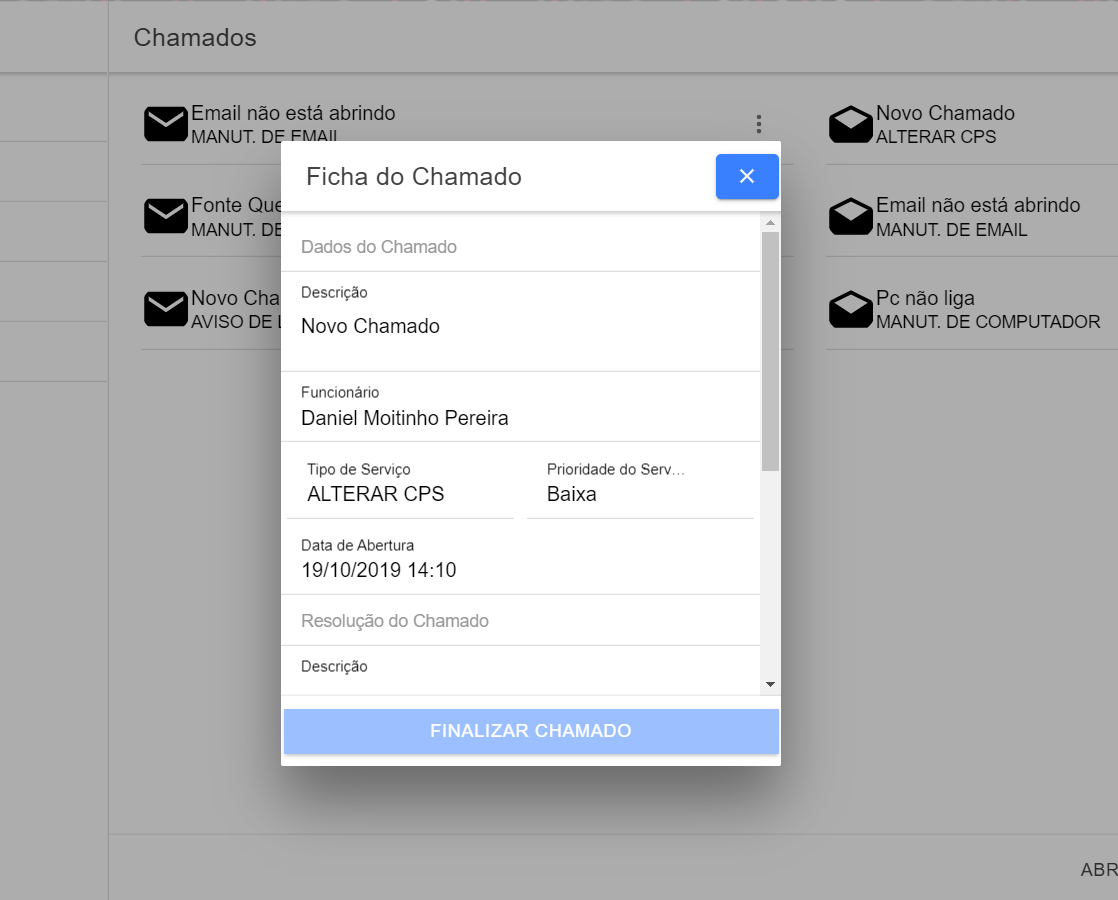
A figura 7 apresenta o modal de abertura de um chamado

**Figura 7 - Modal de abertura de um chamado**



A figura 8 apresenta um modal de exibição de um chamado aberto.

**Figura 8 - Modal de exibição de um chamado**



# CONCLUSÃO

O presente trabalho mostrou-se fundamental na solidificação dos conhecimentos adquiridos em sala de aula. O processo de conversão do banco relacional para não relacional, apresentou a possibilidade de modificação de sistemas já em produção. No que tange o desenvolvimento do sistema, o mesmo serviu para o aprendizado prático do funcionamento dos bancos de dados relacionais.

# REFERÊNCIAS

HEUSER, C. A. **Projeto de Banco de Dados**. 6a ed. Editora Bookman, 2009.

SILBERSCHATZ, A., KORTH, H. F.; SUDARSHAN, S. **Sistemas de bancos de dados**, 3ª ed., São Paulo: Makron Boks, 1999.

# APÊNDICE A – ARQUIVOS BANCO DE DADOS RELACIONAL

**Criação:** <https://raw.githubusercontent.com/mastercoks/chamados/master/docs/sql/create.sql>

**Inserção:**

<https://raw.githubusercontent.com/mastercoks/chamados/master/docs/sql/insert.sql>

**Consultas:**

<https://raw.githubusercontent.com/mastercoks/chamados/master/docs/sql/consultas.sql>

# APÊNDICE B – ARQUIVOS BANCO DE DADOS NÃO RELACIONAL

**Esquema:**

<https://raw.githubusercontent.com/mastercoks/chamados/master/docs/nosql/esquema.json>

**Inserção de Produtos:**

<https://raw.githubusercontent.com/mastercoks/chamados/master/docs/nosql/inserts/produtos.js>

**Inserção de Funcionários:**

<https://raw.githubusercontent.com/mastercoks/chamados/master/docs/nosql/inserts/funcionarios.js>

**Inserção de Chamados:**

<https://raw.githubusercontent.com/mastercoks/chamados/master/docs/nosql/inserts/chamados.js>

**Consulta 1:**

<https://raw.githubusercontent.com/mastercoks/chamados/master/docs/nosql/queries/query-01.js>

**Consulta 2:**

<https://raw.githubusercontent.com/mastercoks/chamados/master/docs/nosql/queries/query-02.js>

**Consulta 3:**

<https://raw.githubusercontent.com/mastercoks/chamados/master/docs/nosql/queries/query-03.js>

**Consulta 4:**

<https://raw.githubusercontent.com/mastercoks/chamados/master/docs/nosql/queries/query-04.js>

**Consulta 5:**

<https://raw.githubusercontent.com/mastercoks/chamados/master/docs/nosql/queries/query-05.js>

**Consulta 6:**

<https://raw.githubusercontent.com/mastercoks/chamados/master/docs/nosql/queries/query-06.js>

**Consulta 7:**

<https://raw.githubusercontent.com/mastercoks/chamados/master/docs/nosql/queries/query-07.js>

**Consulta 8:**

<https://raw.githubusercontent.com/mastercoks/chamados/master/docs/nosql/queries/query-08.js>

**Consulta 9:**

<https://raw.githubusercontent.com/mastercoks/chamados/master/docs/nosql/queries/query-09.js>

**Consulta 10:**

<https://raw.githubusercontent.com/mastercoks/chamados/master/docs/nosql/queries/query-10.js>

**Consulta 11:**

<https://raw.githubusercontent.com/mastercoks/chamados/master/docs/nosql/queries/query-11.js>