UNIVERSIDADE CIDADE DE SÃO PAULO

BRUNO VIDEIRA DOS SANTOS

SISTEMA DE HELPDESK

São Paulo

2018

BRUNO VIDEIRA DOS SANTOS

SISTEMA DE HELPDESK

Trabalho de Conclusão de Curso de Engenharia da Computação da Universidade da cidade de São Paulo, como requisito para obtenção do diploma de graduação.

PROFESSOR ORIENTADOR:

MAURICIO WIELER ORELLANA

São Paulo

2018

BRUNO VIDEIRA DOS SANTOS

SISTEMA DE HELPDESK

Trabalho de Conclusão de Curso de Engenharia da Computação da Universidade da cidade de São Paulo, como requisito para obtenção do diploma de graduação.

São Paulo, 12 de novembro de 2018.

**BANCA EXAMINADORA**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Prof. Mauricio Wieler Orellana)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Dedico este trabalho a minha família e amigos que me apoiaram até o momento.

**AGRADECIMENTOS**

Agradeço a minha família por ter me apoiado durante o curso.

Ao Prof. Mauricio Wieler Orellana, por ter orientado este trabalho.

Aos amigos de trabalho, pelos conselhos, força e incentivo.

A todos qυе direta оυ indiretamente fizeram parte de minha formação, о meu muito obrigado.

**RESUMO**

Este projeto tem como objetivo o desenvolvimento de um “sistema de helpdesk mobile multiplataforma”. Hoje quando existe a necessidade de se desenvolver um aplicativo mobile ou desktop, existem diversas plataformas para as quais o aplicativo deve funcionar. No caso de aplicativo mobile as principais plataformas são Android e IOS, e cada uma dessas plataformas envolvem uma linguagem e tecnologia de desenvolvimento próprio, sendo assim o desenvolvedor deveria estudar cada uma das tecnologias para todas plataformas nas quais o aplicativo deve rodar e desenvolver uma versão do aplicativo para cada plataforma, o que acarreta numa curva de aprendizado maior (mais linguagens para estudar) e maior tempo de desenvolvimento (várias versões do mesmo produto). Uma alternativa para otimizar este processo é a utilização de uma tecnologia de desenvolvimento cross platform (multiplataforma), onde a partir de um mesmo código pode-se gerar um aplicativo para todas as plataformas necessárias com pouca ou nenhuma modificação. Uma das opções para o desenvolvimento multiplataforma mobile é o Apache Cordova, que permite através do uso de tecnologias web gerar aplicativos para diferentes plataformas como Android e IOS. Para exemplificar o uso dessa tecnologia será desenvolvido um sistema de helpdesk, com as funcionalidades: Gerenciamento e atendimento de chamados, gerenciamento de catálogo de serviços, gerenciamento de usuários, base de conhecimento, além de outras.

Palavras-chaves: HelpDesk, Mobile, Multiplataforma, Web.

**ABSTRACT**

This project has object develop a helpdesk mobile cross platform system. Today when there is a need to develop a mobile or desktop application, there are several platforms for which the application should work. In the case of a mobile application, the main ones are Android and IOS, and each of these platforms involves a language and development technology of its own, so the developer should study each of the technologies for all platforms in which the application must run and develop a version of application for each platform, which leads to a larger learning curve (more languages ​​to study) and longer development time (several versions of the same product). An alternative to optimize this process is the use of a cross platform development technology, where from the same code you can generate an application for all the necessary platforms with little or no modification. One of the options for the cross platform mobile development is Apache Cordova, which allows through the use of web technologies to generate applications for different platforms like Android and IOS. To exemplify the use of this technology we have developed a helpdesk system, with the following functions: Call management, service catalog management, user management, knowledge base, among others.

Keywords: HelpDesk, Mobile, Cross Platform, Web.

SUMÁRIO

[1.INTRODUÇÃO 9](#_Toc465800076)

[1.2**.** OBJETIVOS 10](#_Toc465800077)

1.3 Justificativa...........................................................................................................11

2. Revisão de literatura...............................................................................................12

3. Materiais e métodos...............................................................................................14

[4. Resultados 25](#_Toc465800089)

[**5.** CONCLUSÃO 36](#_Toc465800097)

[**6.** REFERÊNCIAS 37](#_Toc465800097)

7. Apêndice ...............................................................................................................38

# INTRODUÇÃO

. Atualmente, quando é necessário desenvolver um aplicativo mobile ou desktop, existem diversas plataformas para as quais o aplicativo deve funcionar. No caso de aplicativo mobile as principais são Android e IOS, e cada uma dessas plataformas envolve uma linguagem e tecnologia de desenvolvimento próprio, sendo assim o desenvolvedor deveria estudar cada uma das tecnologias para todas plataformas nas quais o aplicativo deve rodar e desenvolver uma versão do aplicativo para cada plataforma, o que acarreta numa curva de aprendizado maior (mais linguagens para estudar) e maior tempo de desenvolvimento (várias versões do mesmo produto).

Para otimizar este processo uma das alternativas é a utilização de uma tecnologia de desenvolvimento cross platform, onde a partir de um mesmo código pode-se gerar um aplicativo para todas as plataformas necessárias com pouca ou nenhuma modificação. Uma das opções para o desenvolvimento cross platform mobile é o Apache Cordova/Phonegap, que permite através do uso de tecnologias web gerar aplicativos para diferentes plataformas como Android, IOS, Windows Phone e etc.

Para exemplificar o uso dessa tecnologia será desenvolvido um sistema de helpdesk, com as funcionalidades: Gerenciamento e atendimento de chamados, gerenciamento de catálogo de serviços, gerenciamento de usuários, base de conhecimento, além de outras. A interface do usuário foi escrita com tecnologias web (html, css e javascript) e exportada para aplicativo com o uso do PhoneGap/Cordova.

A interface do sistema será em um aplicativo mobile, porém a inteligência do sistema está em um API Web, ou seja, o backend, desenvolvido com NODE.JS EXPRESS e MongoDB.

No decorrer deste será abordado sistemas helpdesk, multiplataforma mobile, phonegap/cordova e as tecnologias usadas no desenvolvimento.

**O que é HelpDesk?**

O termo Help Desk refere-se ao serviço de suporte para problemas técnicos encontrados por usuários na utilização de sistemas ou equipamentos (COHEN, 2008). Entre os principais serviços atendidos pelo Help Desk estão os problemas técnicos de hardware, inconsistências em softwares, não funcionamento de redes e serviços, instalação de equipamentos, implantação de softwares, entre outros (COHEN, 2008).

**O que é multiplataforma/crossplatform?**

Um software que pode ser executado em mais do que uma plataforma (sistema operacional) é chamado de software multiplataforma. Para um programa ser desenvolvido tendo em vista o suporte multiplataforma, um conceito essencial é a portabilidade do código-fonte, para que a partir de um mesmo código-fonte possa ser executado em diferentes plataformas. Uma das tecnologias mais utilizadas para desenvolvimento mobile multiplataforma é o Phonegap/Cordova, que terá sua utilização demonstrada neste trabalho. O termo em inglês para multiplataforma é CrossPlatform.

# 1.2 OBJETIVOS

# OBJETIVO GERAL

Desenvolver um sistema de helpdesk mobile multiplataforma, sendo assim a partir de um único código podemos exportar o sistema para diferentes plataformas móveis como Android e IOS.

# OBJETIVO ESPECÍFICO

Os objetivos específicos deste projeto são:

* Conhecer mais sobre desenvolvimento mobile multiplataforma e suas vantagens
* Conhecer mais sobre sistemas helpdesk

**1.3 Justificativa**

Os sistemas helpdesk são muito utilizados no dia-a-dia, quando um cliente vai abrir um chamado para solucionar um problema em uma empresa. Porém, há poucos trabalhos acadêmicos sobre o assunto, e também poucos livros em português. Dos trabalhos sobre o assunto alguns falam sobre aspectos como implantação, análise, entre outros, mas neste trabalho será tratado sobre a parte do desenvolvimento. E não será feito modificando um sistema já existente mas sim uma codificação do zero. E nos trabalhos sobre desenvolvimento de sistema help desk disponíveis, geralmente é tratado sobre web ou desktop, mas este tratará sobre mobile e com multiplataforma. Então o trabalho acrescenta a contribuição de o sistema ter a diferença de ser mobile e multiplataforma (que é uma estratégia que traz a vantagem de a partir de um único código desenvolver para diferentes plataformas). E o fato de ser mobile traz a vantagem de que pode ser vendido nas lojas de aplicativo e usar os recursos do dispositivo mobile.

**2. Revisão de literatura**

No livro “Construindo Aplicativos Android com HTML, CSS e JavaScript”, de Jonathan Stark com Brian Jepson, é apresentado as técnicas de desenvolvimento mobile híbrido, porém com foco em Android. Apesar de que o conteúdo desse livro pode ser aproveitado para multiplataforma, como o próprio autor diz, pois o conteúdo fala sobre PhoneGap e Cordova. O livro inicia fazendo definições, depois trata da criação de interfaces utilizando HTML, CSS e Javascript, depos ele apresenta como tornar o aplicativo web em um nativo, a usar recursos do nativo e a enviar o aplicativo para a loja Android.

No livro “Implantação de Help Desk e Service Desk” do autor “Roberto Cohen” é abordado conceitos sobre Helpdesk, porém o autor não fala sobre a parte desenvolvimento de sistemas (como desenvolver um software de helpdesk), mas sim sobre conceitos e a parte de negócio.

No trabalho acadêmico “Sistema para Gerenciamento de Chamados Técnicos” do autor “Leandro Teixeira Perondi” da “UNIVERSIDADE DE CAXIAS DO SUL” ele trata sobre sistemas de helpdesk, porém focando mais na parte de implantação e configuração, e não desenvolvimento. Pois ele utiliza um software pronto, o GLPI, que é um software Open Source que pode ser utilizado para help desk. Porém, os conceitos e definições apresentados no trabalho do Leandro foram úteis para adquirir conhecimento sobre a área para que este trabalho pudesse ser feito.

No trabalho acadêmico “SSP: UMA SOLUÇÃO INTELIGENTE PARA ATENDIMENTO HELP DESK E SERVICE DESK” do autor “FELIPE FERNANDO DE SOUZA CARVALHO” da instituição “UNIVERSIDADE FEDERAL DO PARANÁ” é abordado o desenvolvimento de sistemas helpdesk. Ele se propõe a desenvolver o sistema para a plataforma JAVA EE (que é utilizado para web e desktop, diferente deste trabalho que é focado em mobile). As funcionalidades que o autor apresentou para o sistema foram: “catálogo de serviços, gerenciamento de equipe de atendimento, roteamento parametrizável de chamados ou incidentes e SLA configurável com base na prioridade”. As funcionalidades apontadas por este autor no trabalho dele serviram de inspiração para as funcionalidades aqui deste trabalho. O trabalho deste autor não aborda o assunto mobile.

No trabalho “DESENVOLVIMENTO DE UM PROTÓTIPO DE APLICATIVO MÓVEL UTILIZANDO FERRAMENTA MULTIPLATAFORMA” de “HELENA BASSOTTO” a autora fala sobre desenvolvimento de aplicativos multiplataforma e cita outras opções de framework além do Phonegap abordado neste trabalho.

# 3. Materiais e métodos

Nesta parte do trabalho será apresentado a metodologia, tecnologias utilizadas no desenvolvimento, funcionalidades do sistema e documentação.

**METODOLOGIA**

A interface do usuário foi escrita com tecnologias web (html, css e javascript) e exportada para aplicativo com o uso do PhoneGap/Cordova.

A interface do sistema será em um aplicativo mobile, porém a inteligência do sistema está em um API Web, ou seja, o backend, desenvolvido com NODE.JS EXPRESS e MongoDB.

Foi utilizado programação em camadas e orientação a objetos.

**Html, Css e Javascript**

HTML (Hypertext Markup Language) é uma das linguagens utilizadas para criação de websites. Ela é baseada em marcação, com ela definimos o conteúdo e elementos de uma página. Foi criada por Tim Berners-Lee.

CSS (Cascading Style Sheets) é uma linguagem de estilização. É usada para dar o estilo a uma página web, como cor de fundo, cor de texto, efeitos e etc.

Javascript é uma linguagem de programação interpretada que foi originalmente feita como parte dos navegadores da web para scripts pudessem ser executados do lado cliente. Ela foi criada por Brendan Eich.

# NodeJs e MongoDB

Node.js É uma tecnologia que permite utilizar o Javascript, que originalmente foi feito para rodar no browser, para rodar no servidor, como PHP, Java e etc. Tem como característica programação assíncrona. Foi criada por Ryan Dahl.

MongoDB É um banco de dados orientada a documentos, é NoSQL (não utiliza sql) usa documentos semelhantes a JSON no armazenamento.

# Phonegap/Cordova

Apache Cordova é um framework para desenvolvimento de aplicativos móveis código aberto.

A empresa Nitobi criou o Phonegap, que foi adquirido pela Adobe. A adobe doou o código para a Apache, que criou uma versão open source do PhoneGap chamada ApacheCordova. O PhoneGap é semelhante ao Cordova, porém com alguns recursos a mais, e é mantido pela Adobe.

O Cordova tem acesso a funções dos aparelhos móveis como acelerômetro, câmera e geolocalização, e permite que os desenvedores criem aplicaçoes híbridas

utilizando HTML5, CSS3 e JavaScript. Uma aplicação híbrida está entre um aplicativo mobile web e um aplicativo mobile nativo.

O aplicativo mobile web é um website otimizado para o mobile que é acessado a partir de uma url, já um aplicativo nativo é instalado a partir da loja de aplicativos do sistema seja ele android/ios/etc, possui acesso aos recursos de hardware e geralmente é mais rápido que um website por estar instalado no aplicativo.

Já uma aplicação hibrida combina os dois, pegando um web site mobile e colocando dentro de webview, combinando com um api para acessar os recursos nativos e de hardware, e depois tudo isso é empacota e exportado gerando um aplicativo que pode ser instalado a partir da loja, com a diferença de a interface ter sido feita com tecnologias web.

Os aplicativos criados são compatíveis com iOS, Windows Phone e Android.

Através do uso de plugins é extendida as funcionalidades do phonegap para o uso dos recursos do sistema do dispositivo. Por exemplo, temos plugin para usar câmera, notificações e etc. Então basta o desenvolver adicionar o plugin no projeto e chamar as funções utilizando javascript para usar os recursos do dispositivo. E sendo assim a mesma programação Javascript para controlar os recursos como câmera, microfone, localização, notificações e etc vai funcionar em todas plataformas (android, ios, etc). Se fosse uma programação nativa, cada plataforma teria uma sintaxe diferente para controlar os recursos de sistema (câmera, microfone, notificações e etc).

# Funcionalidades do sistema

O sistema de helpdesk implementado tem 3 tipos de usuário: Administrador, Técnico e Cliente.

As funcionalidades que foram implementadas no sistema são: Gerenciamento de chamados e atendimento, Gerenciamento de catalógo de serviços, Roteamento de chamado, Base de conhecimento e Gerenciamento de Usuários

# Gerenciamento de chamados e atendimento

O cliente pode abrir chamados, acompanhar o andamento, responder eventuais solicitaçoes de informaçao por parte do técnico, e ver quais são os serviços ofertados.

O técnico deve acompanhar o chamados que são abertos no seu departamento e atende-los, dando uma resposta e solução para o cliente e depois encerrar o chamado.

# Gerenciamento de catálogo de serviços e departamentos

O cliente pode ver quais são os serviços ofertados antes de abrir um chamado.

Para mostrar a importância dessa funcionalidade no sistema, pode-se citar o que disse o autor Robert Cohens em seu livro “Implantação de Help Desk e Service Desk”: “Tal como um cardápio de restaurante no qual são apresentados os serviços que o mesmo oferece, você precisa listar o que disponibiliza ao seu usuário. Esse procedimento facilita que ambos os lados compreendam que coisas podem ser pedidas e quais precisam de uma negociação especial.”

No sistema Thunder Help Desk todo serviço está atrelado a um departamento.

O responsável por cadastrar departamentos e serviços é o administrador.

# Roteamento de chamados

O roteamento é feito com base no departamento, quando um usuário abre um chamado para um serviço do departamento de TI, por exemplo, na tela dos técnicos

do departamento de TI aparecerá este chamado como não alocado. Então o chamado aberto é encaminhado para os técnicos do departamento do chamado e um deles deve assumir.

# Base de conhecimento

O sistema permite aos ténicos pesquisar por chamados antigos, pois eles podem ter a resposta para algum novo chamado sobre o mesmo assunto

# Gerenciamento de usuários

O administrador é responsável por cadastrar os usuários do sistema. Os usuários do sistema são cliente, ténico e administrador. O diagrama de caso de uso apresentado na seção de documentação mostra uma relação entre os usuários do sistema e as funcionalidades.

# Notificações

Uma característica das aplicações mobile são as notificações. Por exemplo, quando através do aplicativo de mensagens WhatsApp você envia uma mensagem para uma pessoa, instantaneamente ela receberá uma notificação, mesmo se o aplicativo não estiver aberto, ou o celular estiver bloqueado, essa pessoa receberá uma notificação e então desbloqueará o celular ou apenas abrirá o aplicativo para ver a nova mensagem.

Este recurso também seria muito útil num sistema de helpdesk, para que quando um cliente abrir um chamado os técnicos sejam rapidamente notificados desse novo chamado, e quando os técnicos responderem o chamado o cliente seja rapidamente notificado da resposta. Isso agiliza a comunicação e também faz com que não seja necessário ficar acessando o sistema para ver se há alguma resposta ou novidade.

Para poder implementar esse recurso no sistema de helpdesk mobile desenvolvido neste trabalho, será utilizado dois plugins do Phonegap/Cordova. Um plugin dá o recurso de notificações, e o outro plugin permite executar uma função/serviço em segundo plano (ou background). Esta função fica trabalhando mesmo quando o aplicativo não está aberto ou quando o dispositivo está bloqueado. Esta função fica “escutando” ou “verificando” por novidades, e quando encontra, chama o plugin de notificações para notificar o usuário sobre alguma novidade relacionada aos chamados do sistema.

Para poder utilizar estes plugins devemos adicionar uma referência a eles no arquivo config.xml do phonegap.

“*<plugin name="cordova-plugin-local-notification" spec="0.9.0-beta.3" source="npm" />*

*<plugin name="cordova-plugin-background-mode" source="npm" />*”

Deposi utilizamos os plugins com Javascript. Se fossemos implementar notificação em um aplicativo nativo, cada plataforma (android, ios) teria uma sintaxe diferente para implementar (Android seria usado por exemplo a linguagem Java, já o IOS seria usado a linguagem Switf, Objective-C ou outra)

**Utilização do plugin de notificação**

cordova.plugins.notification.local.schedule({

title: 'My first notification',

text: 'Thats pretty easy...',

foreground: true

});

**Utilização do plugin de serviço em segundo plano**

cordova.plugins.backgroundMode.on('activate', function() {

cordova.plugins.backgroundMode.disableWebViewOptimizations();

});

Implementação combinando os dois plugins para se chegar ao objetivo das notificações dinâmicas

document.addEventListener('deviceready', function () {

cordova.plugins.backgroundMode.enable();

cordova.plugins.backgroundMode.on('activate', function() {

setInterval(function(){

$.ajax(

{

"method":"get",

"url": self.service\_path+"checkNotification",

"headers": self.getHeaders(),

"dataType": "json",

"success":function(data) {

var quant = 0

var msguser = ''

if(data.length) {

quant = data.length

if(quant>1) {

msguser = quant+' chamados novos'

} else {

msguser = quant+' chamado novo'

}

cordova.plugins.notification.local.schedule({

title: 'HelpDesk',

text: msguser

});

}

}

}

);

},30000)

});

}, false);

**Testando essa implementação**

Uma imagem mostrando o resultado do teste dessa implementação está disponívei na seção de resultados deste trabalho, onde são aprestados as telas do sistema.

**Documentação**

Nesta seção será apresentado a documentação do sistema e conceitos relacionados. É mostrado dois diagramas da UML, o diagrama de caso de uso e diagrama de classes

**UML**

UML (Unified Modeling Language, traduzido para Linguagem de Modelagem Unificada) é uma linguagem para elaboração da estrutura de projetos de software. Ela é empregada para visualização, especificação, construção e documentação de artefatos de software.

**Orientação a objetos**

É um paradigma de programação, usado para análise, projeto e programação de software.

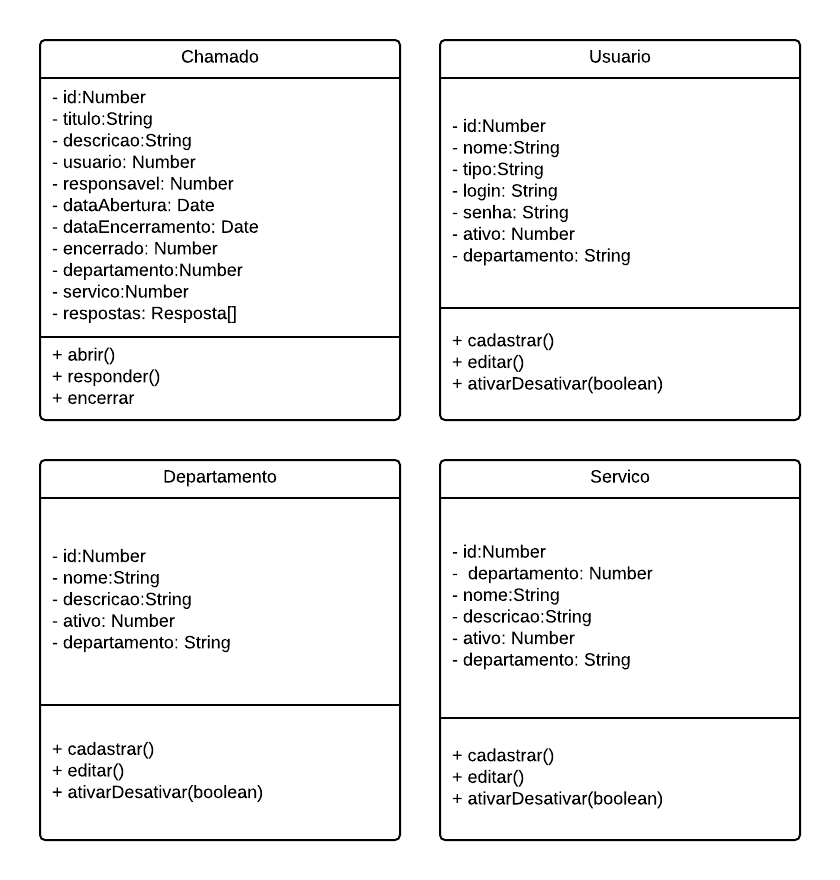
Traz o conceito de classes e objetos. Sendo a classe um modelo de uma entidade do mundo real que possui propriedades (características) e métodos

(funcionalidades, ações). Enquanto a classe é apenas

um modelo ou projeto, o objeto é uma instância dessa classe, uma implementação desse modelo. A partir de uma única classe pode-se instanciar vários objetos. Exemplo: a partir da classe usuário pode-se instanciar os vários usuários que há no sistema, cada um desses usuários é um objeto e o modelo que os representa é a classe.

**Diagrama de classes**

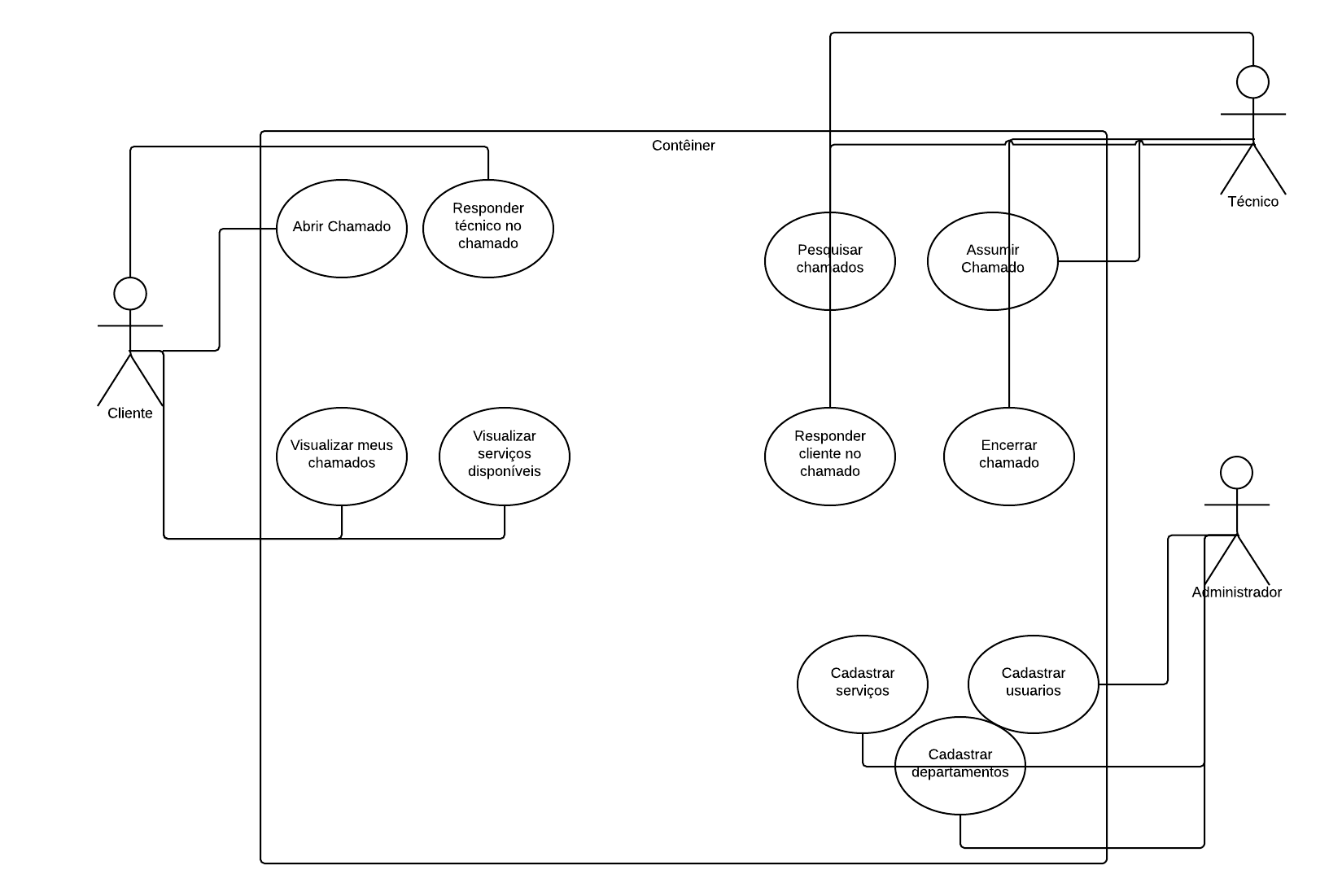
Representa as classes do sistema de helpdesk desenvolvido, com suas propriedades e métodos



*Diagrama de classes do sistema de helpdesk*

**Diagrama de caso de uso**

Demonstra os atores (usuários) do sistema e quais funcionalidades do sistema eles utilizam. Representando assim a relação entre os usuários e as principais funções do sistema.



*Diagrama de caso de uso do sistema de helpdesk*

# 4. Resultados

Nesta parte será apresentado como o sistema ficou, as telas do sistema com fotos tiradas do teste em dispositivo mobile

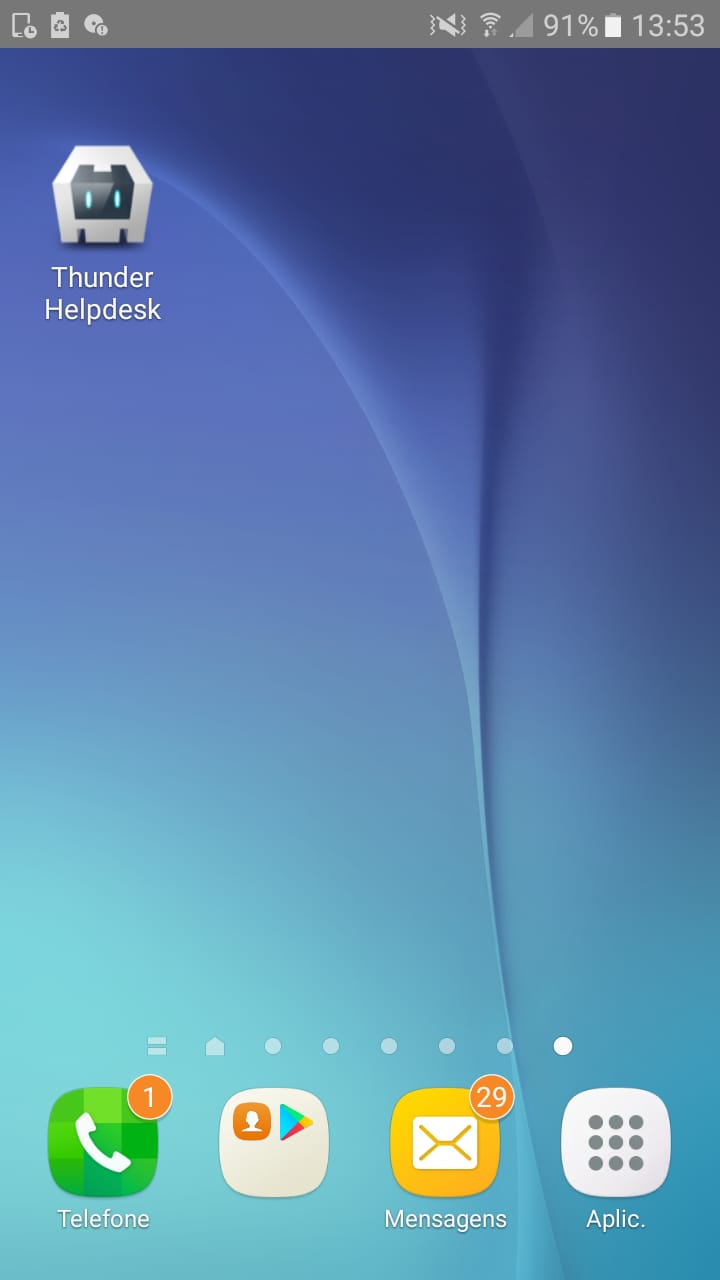


Figura 1 – Ícone do aplicativo de help desk em um celular android. Print tirado do celular.



Figura 2 – Tela de login



Figura 3 – Logado como Administrador, cadastro de usuário

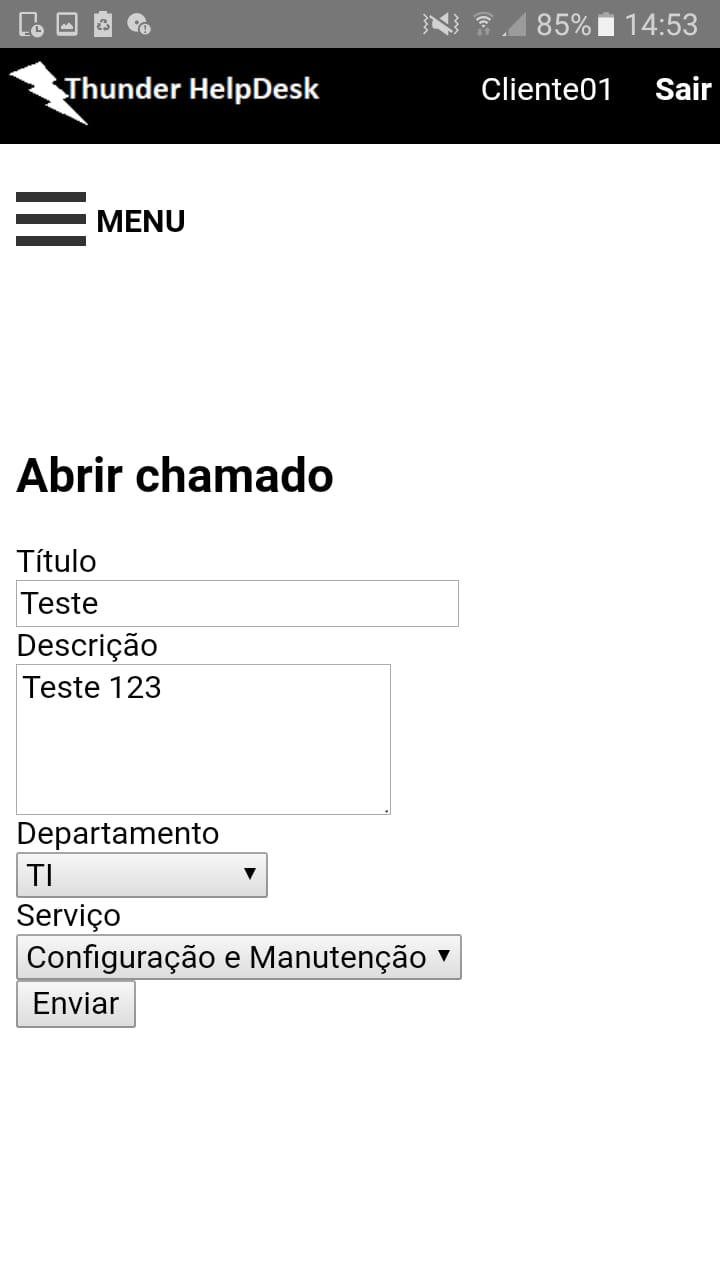


Figura 4 – Logado como Cliente, tela de abrir chamado



Figura 5 - Área do cliente



Figura 6 – Tela de lista de serviços logado como adm

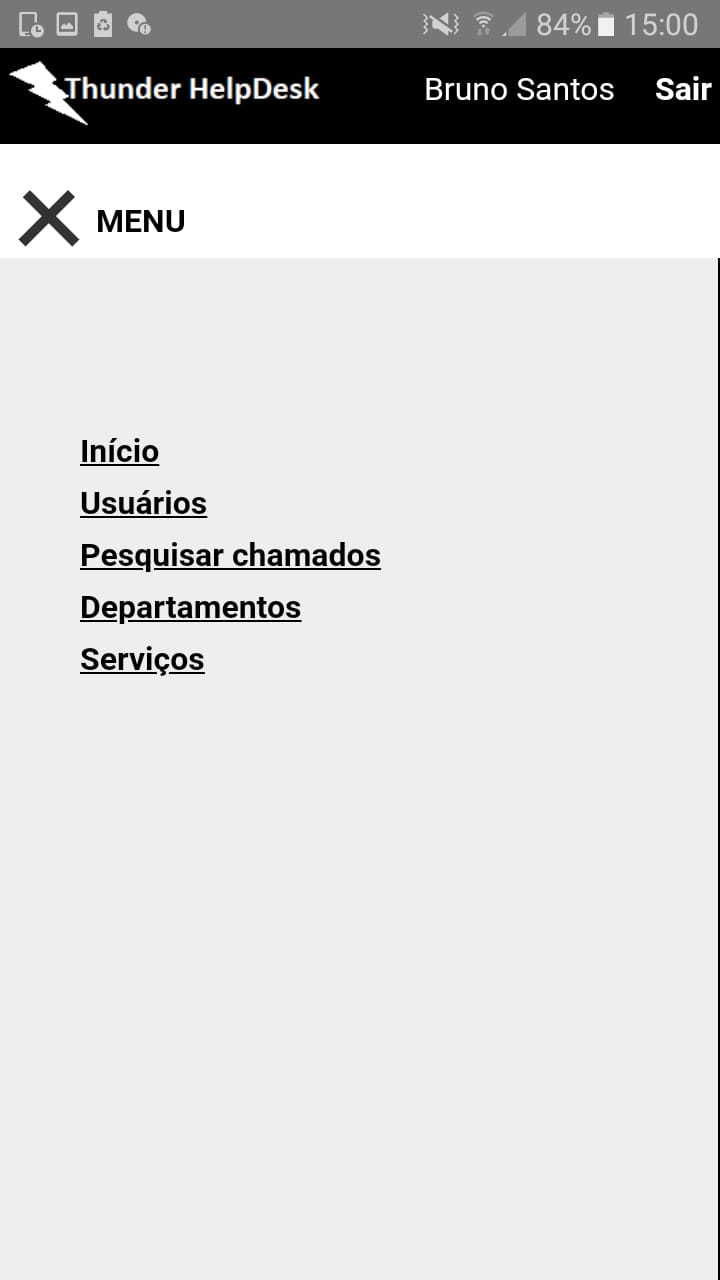


Figura 7 – Menu do administrador quando clicado no botão “Menu”

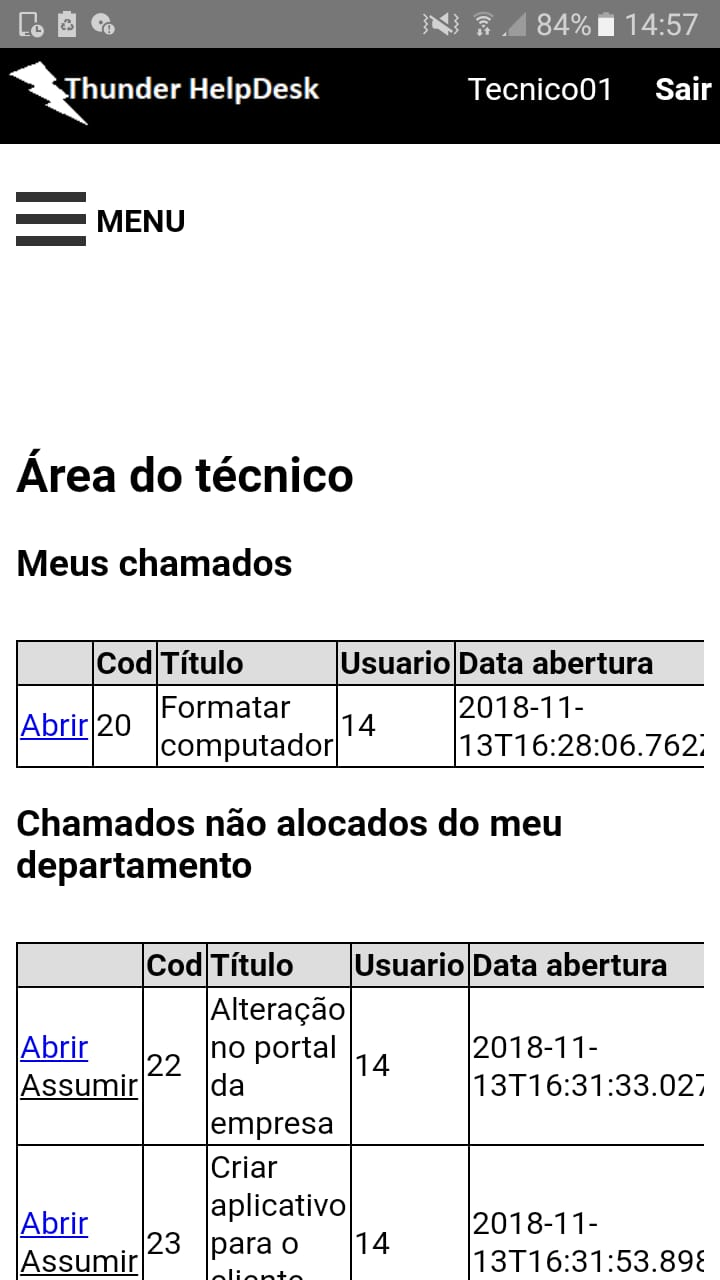


Figura 8 – Área do técnico

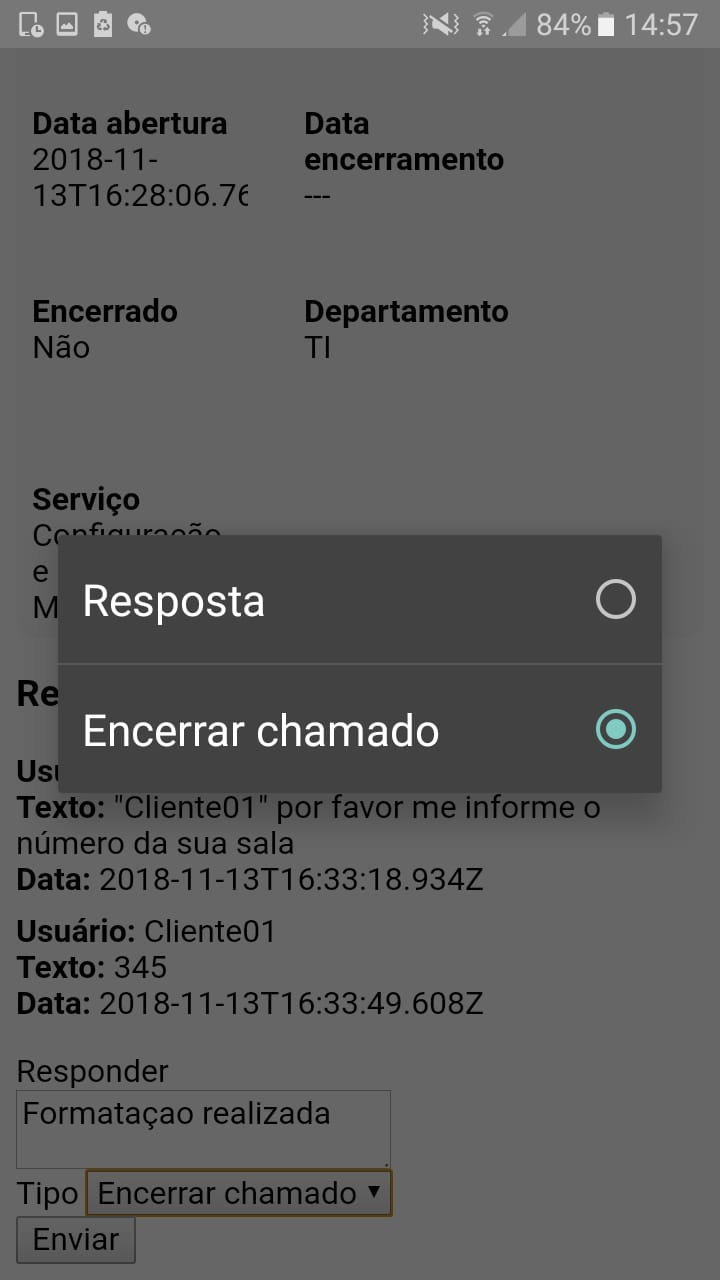


Figura 9 – Técnico encerrando chamado

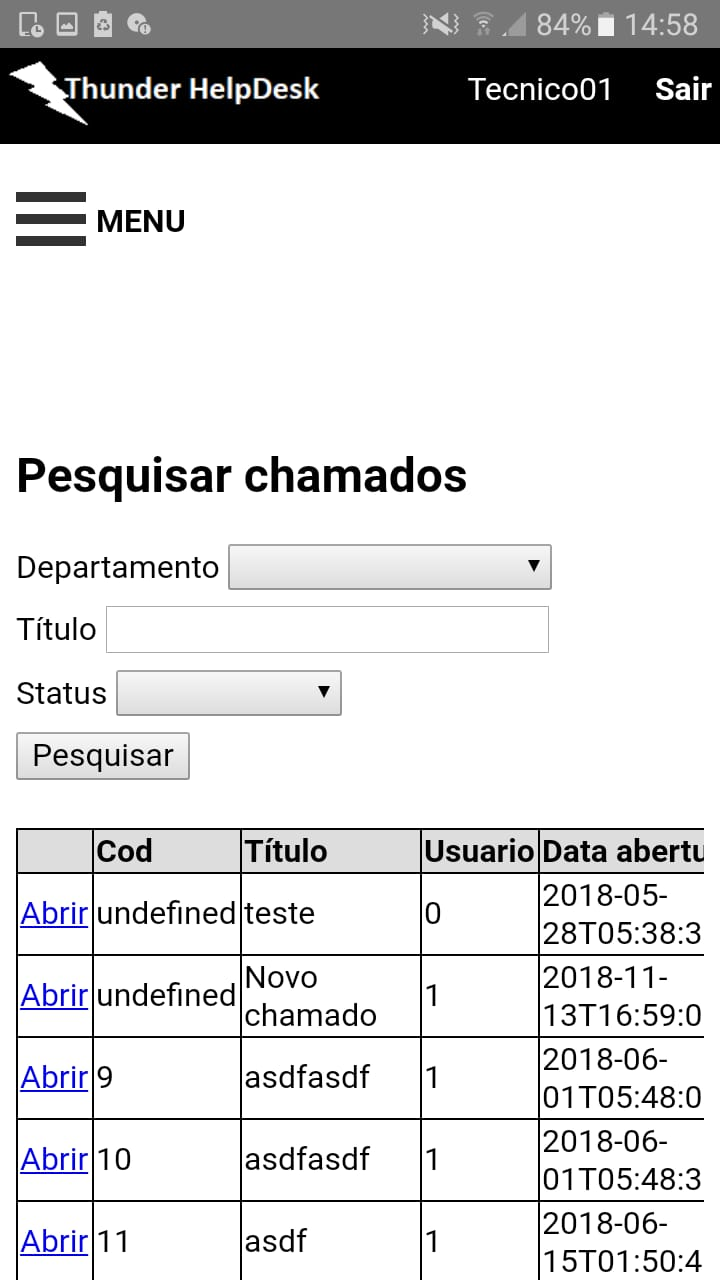


Figura 10 – Pesquisa de chamado

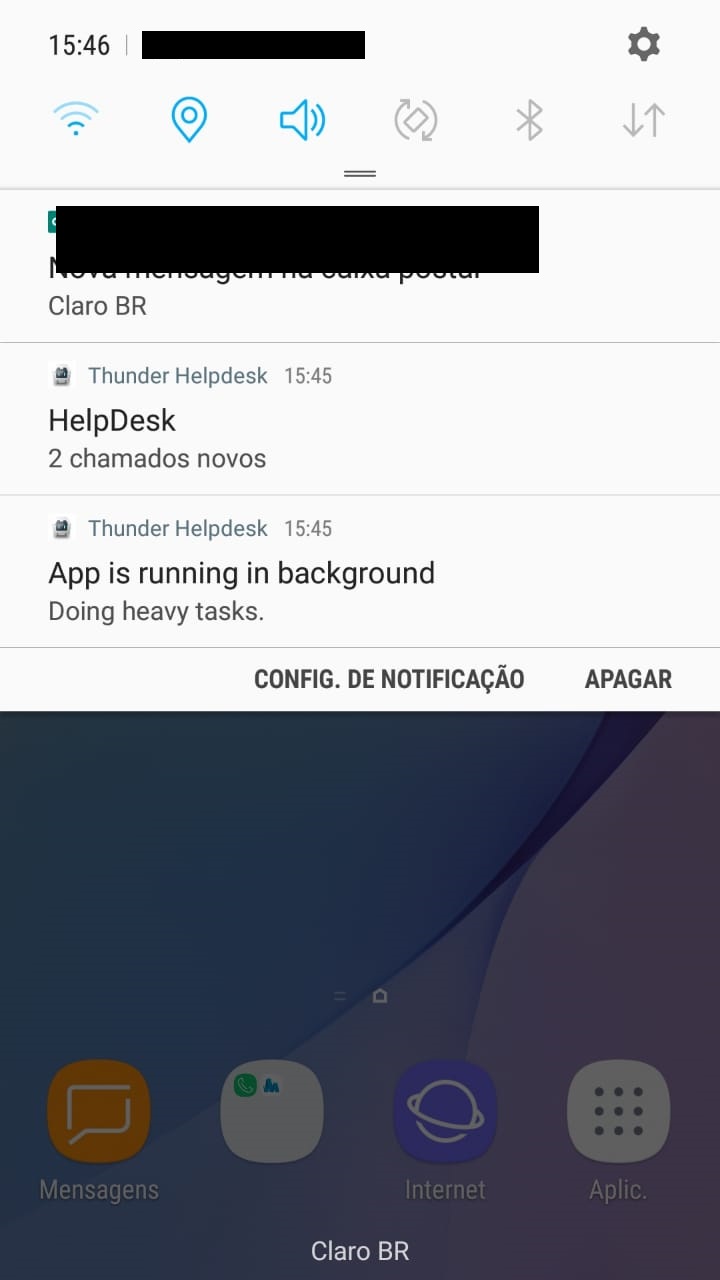


Figura 11 – Notificação exibida quando chega um chamado novo

# 5. CONCLUSÃO

Este trabalho trouxe a contribuição de desenvolver uma versão de sistema de helpdesk em formato de aplicativo mobile, que pode ser distribuído nas lojas de aplicativo e utilizar os recursos nativos do dispositivo. Além disso, com a utilização de tecnologia multiplataforma ele pode ser reaproveitado nas principais plataformas do mercado a partir do mesmo código.

Ao término deste trabalho, constata-se que o desenvolvimento mobile cross platform se mostra um grande aliado do desenvolvedor, pois a partir de único projeto podemos exportar o aplicativo para diferentes plataformas, com o uso do Cordova. Com a utilização do Node no backend, acrescentou-se um reaproveitamento ainda maior de conhecimento, pois a mesma linguagem (JavaScript) foi usada tanto para programar a API no servidor (Node.js) quanto para a interação da interface do aplicativo (Javascript).

Como sugestão de trabalhos futuros, pode-se citar o uso de mais recursos nativos do dispositivo como uso de câmera e microfone, localização entre outros. Pode também se acrescentar novas funcionalidades de helpdesk ao sistema ou melhorar as já existentes.

# . 6. REFERÊNCIAS

BASSOTTO, Helena. **DESENVOLVIMENTO DE UM PROTÓTIPO DE APLICATIVO MÓVEL UTILIZANDO FERRAMENTA MULTIPLATAFORMA.**2014. 82 f. TCC (Graduação) - Curso de CiÊncia da ComputaÇÃo, Universidade de Caxias do Sul, Caxias do Sul, 2014. Disponível em: <https://repositorio.ucs.br/xmlui/bitstream/handle/11338/1512/TCC%20Helena%20Bassotto.pdf?sequence=1&isAllowed=y>. Acesso em: 01 ago. 2018.

CARVALHO, Felipe Fernando de Souza. **SSP: UMA SOLUÇÃO INTELIGENTE PARA ATENDIMENTO HELP DESK E SERVICE DESK.**2015. 279 f. Monografia (Especialização) - Curso de Engenharia de Software, Universidade Federal do Paraná, Curitiba, 2015. Disponível em: <https://acervodigital.ufpr.br/bitstream/handle/1884/39560/R%20-%20E%20-%20FELIPE%20FERNANDO%20DE%20SOUZA%20CARVALHO.pdf?sequence=2&isAllowed=y>. Acesso em: 01 ago. 2018.

PERONDI, Leandro Teixeira. **Sistema para Gerenciamento de Chamados Técnicos.**2013. 91 f. TCC (Graduação) - Curso de Sistemas de Informação, Universidade de Caxias do Sul, Caxiasdo Sul, 2013. Disponível em: <https://repositorio.ucs.br/xmlui/bitstream/handle/11338/1246/TCC%20Leandro%20Teixeira%20Perondi.pdf?sequence=1&isAllowed=y>. Acesso em: 01 ago. 2018.

Cohen, Roberto.; **IMPLANTAÇÃO DE HELPDESK E SERVICEDESK**  São Paulo: novatec, 2008.

Stark, Jonathan.; **Construindo Aplicativos Android com html,css e javascript**  São Paulo: novatec, 2012.

Lopes, Sérgio.; **Aplicações mobile Híbridas com Cordova e Phonegap.** São Paulo: Casa do código, 2016.

**7. Apêndices**

**Código-fonte da aplicação**

Nesta seção será apresentado parte do código-fonte da aplicação. Tanto na parte api quando a parte de interface do aplicativo mobile.

**API**

var express = require('express') ;

var cors = require('cors');

var app = express(),

load = require('express-load'),

mongoose = require('mongoose'),

bodyParser = require('body-parser');

app.use(cors());

app.use(bodyParser.urlencoded({ extended: false }));

app.use(bodyParser.json());

mongoose.connection.on("open", function(ref) {

console.log("Connected to mongo server.");

});

mongoose.connection.on("error", function(err) {

console.log("Could not connect to mongo server!");

return console.log(err);

});

mongoose.connect('mongodb://localhost:27017/helpdesk');

global.db = mongoose.connection;

load('models').then('controllers').then('routes').into(app);

app.set('views', \_\_dirname + '/views');

app.set('view engine', 'ejs');

app.use(express.static(\_\_dirname + '/public'));

global.userId = "";

global.token = "";

var autenticar = function(req,res,next) {

console.log("autenticar");

var Session = app.models.session;

var token = req.header("app-token");

console.log(token);

Session.findOne({"token":token}).exec(function(erro,session){

console.log("callback do autenticar");

if(erro) {

return res.json({"Error":"Não autenticado"});

} else {

if(session) {

global.userId = session.userId;

global.token = token;

console.log("user id: ");

console.log(global.userId);

console.log("...");

} else {

return res.json({"Error":"Não autenticado"});

}

}

return next();

});

}

app.get('/', function(req, res, next) {

res.json({appname:"thunderhelpdesk"});

});

app.get('/loginfake', function(req, res, next) {

res.json({userId:"123",userName:"bruno",userType:"tecnico",userLogin:"blogin"});

});

app.post('/login', function(req, res, next) {

var login = req.body.login;

var senha = req.body.senha;

var userId = ""; // variável que irá armazenar o id do usuário que será carregado caso o login seja feito com sucesso

var userName="";

var userType = "";

var token = ""; // Guardará o token que será gerado

var Usuario = app.models.usuario;

var Session = app.models.session;

console.log(req.body);

console.log({"login":login,"senha":senha,"ativo":1});

Usuario.findOne({"login":login,"senha":senha,"ativo":1}).exec(function(erro,usuario){

console.log(erro);

console.log(usuario);

if(usuario) {

userId = usuario["\_id"];

userName = usuario.nome;

token = Date.now().toString() + "," + userId;

userType = usuario.tipo;

console.log(usuario);

var jsonsession = {

"userId": userId,

"token": token,

"dataCriacao" : Date.now(),

"ultimoAcesso": Date.now()

}

// Guarda na sessão

//Session.insert(jsonsession);

Session.create(jsonsession, function(erro, session) {

console.log(erro);

if(erro){

//res.redirect('/');

res.json({Error:"Não foi possível logar"});

} else {

//req.session.usuario = usuario; res.redirect('/contatos');

res.json({"userId":userId,"token":token,"userName":userName,"userType":userType});

}

});

//res.send("logado com sucesso");

//console.log("logado com sucesso");

} else {

res.json({Error:"Nâo foi possível logar"});

console.log("usuário ou senha inválidos");

}

});

});

app.get('/logout', function(req, res, next) {

console.log('logout');

console.log(req.headers);

console.log(req.header("app-token"));

var token = req.header("app-token");

var Session = app.models.session;

var filter = {

"token": token

}

var response = {};

Session.remove(filter, function(err) {

if (!err) {

response = {"ok":"ok"};

console.log(response);

res.json(response);

}

else {

response = {"Error":err};

console.log(response);

res.json(response);

}

});

});

app.post('/chamados', autenticar, function(req, res, next) {

console.log("rota chamados");

var Chamado = app.models.chamado;

var query = {};

if(req.body.departamento){

query.departamento = req.body.departamento;

}

if(req.body.titulo) {

query.titulo = { "$regex": RegExp(req.body.titulo), "$options": 'i' };

}

if(req.body.status) {

if(req.body.status=='naoalocado') {

query.encerrado = 0;

query.responsavel = 0;

} else if(req.body.status=='emaberto') {

query.encerrado = 0;

query.responsavel = {'$gt': 0};

} else if(req.body.status=='encerrado'){

query.encerrado = 1;

}

}

console.log(query);

Chamado.find(query).exec(function(erro, chamados){

if(erro){

res.json(erro);

} else {

res.json(chamados);

}

})

});

app.get('/getchamado/:id', autenticar, function(req, res, next) {

console.log("chamados");

var result = [];

var Chamado = app.models.chamado;

var id = req.params.id;

Chamado.findById(id, function(err, doc) {

console.log("callback");

if (err) {

console.error('error, no entry found');

res.json({Error:err});

} else {

if(!doc) {

console.log('não encontrado');

res.json({Error:"não encontrado"});

} else {

console.log('encontrado');

result.push(doc)

var Usuario = app.models.usuario;

console.log(doc.usuario)

Usuario.findById(doc.usuario, function(err, usu) {

console.log('find usu')

console.log(err)

console.log(usu)

if(!err && usu) {

console.log('yes')

result.push(usu)

} else {

result.push({})

console.log('no')

}

var Responsavel = app.models.usuario;

Responsavel.findById(doc.responsavel, function(err, resp) {

if(!err && resp) {

result.push(resp)

} else {

result.push({})

console.log('no')

}

var Departamento = app.models.departamento;

Departamento.findById(doc.departamento, function(err, dep) {

if(!err && dep) {

result.push(dep)

} else {

result.push({})

console.log('no')

}

var Servico = app.models.servico;

Servico.findById(doc.servico, function(err, serv) {

if(!err && serv) {

console.log('serv........................')

console.log(serv)

result.push(serv)

} else {

result.push({})

console.log('no serv ...............')

console.log(doc.servico)

console.log(doc.\_id)

console.log(err)

console.log(serv)

}

console.log(doc)

console.log(result)

res.json(result);

})

})

})

})

}

}

});

});

function getNextSequenceValue(sequenceName,callback){

var Counter = app.models.counter;

var seqval = 0;

var sequenceDocument = Counter.findOneAndUpdate(

{ "\_id" : sequenceName },

{ $inc: { "sequence\_value" : 1 } },

{projection: { "\_id" : 1, "sequence\_value" : 1 },returnOriginal: false}

).exec(function(a,b){

console.log("exec");

console.log(a);

console.log(b);

if(b && b.sequence\_value) {

seqval = b.sequence\_value + 1;

} else {

seqval = 1

}

callback(seqval);

});

console.log(sequenceDocument);

console.log(sequenceName);

//return sequenceDocument.sequence\_value;

console.log(seqval);

return seqval;

}

app.post("/abrirchamado",autenticar, function(req,res,next){

var Chamado = app.models.chamado;

var obj = req.body;

getNextSequenceValue("productid",function(a,b){

obj["\_id"] = a;

Chamado.create(obj,function(erro,registro){

if(erro) {

res.json(erro);

} else {

res.json({"message":"Dados salvos com sucesso"});

}

});

});

if(obj["\_id"]) {

res.send("");

}

});

app.post("/atribuirchamado",autenticar, function(req,res,next){

console.log(req.body);

var Chamado = app.models.chamado;

var id = req.body.chamado;

var codusuario = req.body.usuario;

console.log(id);

Chamado.findById(id, function(err, doc) {

if (err) {

console.error('error, no entry found');

res.json({Error:err});

} else {

console.log(doc);

if(doc) {

doc.responsavel = codusuario;

doc.save(function(err){

if(err) {

res.json({Error:err});

} else {

res.json({"message":"Dados salvos com sucesso"});

}

});

} else {

console.log('não encontrado');

res.json({Error:"Nâo encontrado"});

}

}

});

});

app.post("/responderchamado/:id",autenticar, function(req,res,next){

console.log(req.body);

var Chamado = app.models.chamado;

var id = req.params.id;

var Usuario = app.models.usuario;

Usuario.findOne({"\_id":global.userId}).exec(function(erro,usu){

if(erro) {

res.json({Error:erro});

} else {

if(usu) {

//res.json(usu);

var descricao = req.body.descricao;

var usuario = global.userId;

var nomeUsuario = usu.nome;

var data = Date.now();

var solicitacaoAprovacao = false;

var aprovacaoCliente = false;

var encerramento;

var privado;

var tipo = req.body.tipo; // Esta variável não é armazenada (ou seja, não faz parte do model), ela serve para processar e definir comportamento e os valores de outras variáveis que fazem parte do model

if(tipo == 'comentario') {

privado = true;

encerramento = false;

} else if(tipo == 'encerramento') {

privado = false;

encerramento = true;

// alterar a propriedade encerrado do chamado e a propriedade data do encerramento

} else if(tipo=='resposta') {

privado = false;

encerramento = false;

}

Chamado.findById(id, function(err, doc) {

if (err) {

console.error('error, no entry found');

res.json({Error:err});

} else {

var newResponse = {

"descricao": descricao,

"usuario": usuario,

"nomeUsuario": nomeUsuario,

"data": data,

"solicitacaoAprovacao": solicitacaoAprovacao,

"aprovacaoCliente": aprovacaoCliente,

"encerramento": encerramento,

"privado": privado

}

doc.respostas.push(newResponse);

if(encerramento) {

doc.encerrado = 1;

doc.dataEncerramento = Date.now();

}

doc.save(function(err){

if(err) {

res.json({Error:err});

} else {

res.json({"message":"Dados salvos com sucesso"});

}

});

}

});

} else {

res.json({Error:"Usuário logado não encontrado"});

}

}

});

});

app.get('/getchamadosbyresponsavel', autenticar, function(req, res, next) {

var Chamado = app.models.chamado;

var query = {"responsavel":global.userId,"encerrado":0};

Chamado.find(query,function(err,doc){

if(err) {

res.json({"Error":err});

} else {

if(doc && doc.length) {

res.json(doc);

} else {

res.json({"Error":"not found"});

}

}

});

});

app.get('/getchamadosdepartamento', autenticar, function(req, res, next) {

var Chamado = app.models.chamado;

var Usuario = app.models.usuario;

Usuario.findOne({"\_id":global.userId}).exec(function(erro,usuario){

if(erro) {

res.json({Error:erro});

} else {

if(usuario) {

var idDepartamento = usuario.departamento

var query = {"departamento":idDepartamento,"encerrado":0,"responsavel":0};

Chamado.find(query,function(err,doc){

if(err) {

res.json({"Error":err});

} else {

if(doc && doc.length) {

res.json(doc);

} else {

res.json({"Error":"not found"});

}

}

});

} else {

res.json({Error:"not found"});

}

}

});

});

app.get('/servicos/:id', autenticar, function(req, res, next) {

var Servico = app.models.servico;

var query = {"departamento":req.params.id};

Servico.find(query,function(err,doc){

if(err) {

res.json({"Error":err});

} else {

if(doc && doc.length) {

res.json(doc);

} else {

res.json({"Error":"not found"});

}

}

});

});

app.get('/getchamadoscliente', autenticar, function(req, res, next) {

var Chamado = app.models.chamado;

var query = {"usuario":global.userId};

Chamado.find(query,null,{sort: {'\_id': -1}},function(err,doc){

if(err) {

res.json({"Error":err});

} else {

if(doc && doc.length) {

res.json(doc);

} else {

res.json({"Error":"not found"});

}

}

});

});

app.get('/getchamadosnaoalocado', autenticar, function(req, res, next) {

var Chamado = app.models.chamado;

var query = {"responsavel":0,"encerrado":0};

console.log(query);

Chamado.find(query,function(err,doc){

if(err) {

res.json({"Error":err});

} else {

if(doc && doc.length) {

console.log(doc);

res.json(doc);

} else {

res.json({"Error":"Não encontrado"});

}

}

});

});

app.get('/usuarios', autenticar, function(req, res, next) {

var Usuario = app.models.usuario;

Usuario.find({}).exec(function(err,data){

if(err) {

return res.json({"Error":err});

} else {

return res.json(data);

}

});

});

app.get('/departamentos', autenticar, function(req, res, next) {

var Departamento = app.models.departamento;

Departamento.find({}).exec(function(err,data){

if(err) {

return res.json({"Error":err});

} else {

return res.json(data);

}

});

});

app.get('/getuser/:id', autenticar, function(req, res, next) {

if(req && req.params && req.params.id) {

var id = req.params.id;

var Usuario = app.models.usuario;

Usuario.findOne({"\_id":id}).exec(function(erro,usuario){

if(erro) {

res.json({Error:erro});

} else {

if(usuario) {

res.json(usuario);

} else {

res.json({Error:"Usuário não encontrado"});

}

}

});

} else {

var result = {Error:"Não encontrado"};

}

});

app.get('/getservice/:id', autenticar, function(req, res, next) {

if(req && req.params && req.params.id) {

var id = req.params.id;

var Servico = app.models.servico;

Servico.findOne({"\_id":id}).exec(function(erro,servico){

if(erro) {

res.json({Error:erro});

} else {

if(servico) {

res.json(servico);

} else {

res.json({Error:"Serviço não encontrado"});

}

}

});

} else {

var result = {Error:"Não encontrado"};

}

});

app.get('/getdepartment/:id', autenticar, function(req, res, next) {

if(req && req.params && req.params.id) {

var id = req.params.id;

var Departamento = app.models.departamento;

Departamento.findOne({"\_id":id}).exec(function(erro,departamento){

if(erro) {

res.json({Error:erro});

} else {

if(departamento) {

res.json(departamento);

} else {

res.json({Error:"Departamento não encontrado"});

}

}

});

} else {

var result = {Error:"Não encontrado"};

}

});

//Cadastrar novo usuário

app.post('/saveuser',autenticar, function(req,res,next) {

console.log("novo usuario");

var Usuario = app.models.usuario;

getNextSequenceValue("userid",function(a,b){

console.log("getsequence");

var jsonbody = req.body;

jsonbody["\_id"] = a;

Usuario.create(jsonbody, function(erro, usuario) {

console.log("user create callback");

if(erro){

res.json({Error:erro});

} else {

res.json({msg:"Salvo com sucesso"});

}

});

});

});

//Cadastrar novo departamento

app.post('/savedepartment',autenticar, function(req,res,next) {

console.log("novo departamento");

var Departamento = app.models.departamento;

getNextSequenceValue("departmentid",function(a,b){

console.log("getsequence");

var jsonbody = req.body;

jsonbody["\_id"] = a;

Departamento.create(jsonbody, function(erro, departamento) {

console.log("department create callback");

if(erro){

res.json({Error:erro});

} else {

res.json({msg:"Salvo com sucesso"});

}

});

});

});

//Cadastrar novo serviço

app.post('/saveservice',autenticar, function(req,res,next) {

console.log("novo serviço");

var Servico = app.models.servico;

getNextSequenceValue("serviceid",function(a,b){

console.log("getsequence");

var jsonbody = req.body;

jsonbody["\_id"] = a;

Servico.create(jsonbody, function(erro, servico) {

console.log("service create callback");

if(erro){

res.json({Error:erro});

} else {

res.json({msg:"Salvo com sucesso"});

}

});

});

});

//Atualizar usuário

app.post('/saveuser/:id',autenticar, function(req,res,next) {

console.log("editar usuario");

var id = req.params.id;

var Usuario = app.models.usuario;

Usuario.findById(id, function(err, doc) {

if (err) {

console.error('error, no entry found');

res.json({Error:err});

} else {

doc.nome=req.body.nome;

doc.senha=req.body.senha;

doc.departamento=req.body.departamento;

doc.ativo=req.body.ativo;

doc.tipo=req.body.tipo;

if(req.body.gerenciarchamado!=undefined) {

doc.gerenciarchamado=req.body.gerenciarchamado;

}

doc.save(function(err){

if(err) {

res.json({Error:err});

} else {

res.json({"message":"Dados salvos com sucesso"});

}

});

}

});

});

//Atualizar departamento

app.post('/savedepartment/:id',autenticar, function(req,res,next) {

console.log("editar departamento");

var id = req.params.id;

var Departamento = app.models.departamento;

Departamento.findById(id, function(err, doc) {

if (err) {

console.error('error, no entry found');

res.json({Error:err});

} else {

doc.nome=req.body.nome;

doc.ativo=req.body.ativo;

doc.descricao=req.body.descricao;

doc.save(function(err){

if(err) {

res.json({Error:err});

} else {

res.json({"message":"Dados salvos com sucesso"});

}

});

}

});

});

//Atualizar serviço

app.post('/saveservice/:id',autenticar, function(req,res,next) {

console.log("editar serviço");

var id = req.params.id;

var Servico = app.models.servico;

Servico.findById(id, function(err, doc) {

if (err) {

console.error('error, no entry found');

res.json({Error:err});

} else {

doc.nome=req.body.nome;

doc.ativo=req.body.ativo;

doc.descricao=req.body.descricao;

//doc.departamento= não permitir alterar departamento

doc.save(function(err){

if(err) {

res.json({Error:err});

} else {

res.json({"message":"Dados salvos com sucesso"});

}

});

}

});

});

app.listen(3000, function(){ console.log("Helpdesk no ar."); });

**MOBILE/INTERFACE (javascript de algumas telas)**

/\*\*

\*@fileOverview O script principal da aplicação. Contém a classe App

\* @author Bruno Santos

\*

\*/

/\*\*

\* Classe principal

\* @class App

\*/

function App() {

var self=this;

this.rota="";

this.parametros=[];

this.pagina="";

this.scriptslist=[];

this.cachePage=false;

this.dataCache=[];

this.pageMessage="";

this.fromPage="";

this.toPage="";

this.actual\_callId="";

// Caminho de testes/desenvolvimento

//this.service\_path\_dev = "http://localhost:3000/";

// Caminho de produção

//this.service\_path\_production = "http://localhost:3000/";

// Caminho atual utilizado pela aplicação, pode ser o de produção ou desenvolvimento/testes

this.service\_path = Config.getServPath();

/\*\*

\* Fecha todos os modals

\* @method closeAllModals

\* @memberOf App#

\*/

this.closeAllModals=function() {

if(!$(".modal").hasClass("hide")) {

$(".modal").addClass("hide");

}

$(".modal .modal-box").each(function(index,value){

if(!$(this).hasClass("hide")) {

$(this).addClass("hide");

}

});

}

/\*\*

\* Exibe um modal de confirmação, que mostra uma mensagem com um botão de ok, e opcionalmente executa uma função quando o usuário clica no botão OK

\* @method showConfirmDialog

\* @memberOf App#

\*/

this.showConfirmDialog=function(message,fn) {

self.closeAllModals();

// Caso o usuário não informe a função define uma função vazia que não faz nada para evitar erro

if(!fn) {

fn=function(){}

}

//Esconde imagem de alerta

$("#alertimage").removeClass("hide");

$("#alertimage").addClass("hide");

//Remove binds antigos que foram vinculados aos botões

$("body").off("click","#btgeneric\_cancelar");

$("body").off("click","#btgeneric\_ok");

//Exibe o container de modal

$(".modal").removeClass("hide");

// Exibe o modal especifico para esta utlidade (modal generic)

$(".modal .generic").removeClass("hide");

// Esconde o botão CANCELAR

$("#btgeneric\_cancelar").removeClass("hide");

$("#btgeneric\_cancelar").addClass("hide");

// Mostra o botão ok

$("#btgeneric\_ok").removeClass("hide");

$("body").on("click","#btgeneric\_ok",self.hideConfirmDialog);

// Faz o bind da função callback

$("body").on("click","#btgeneric\_ok",fn);

$("#generic\_message").html("");

$("#generic\_message").html(message);

}

/\*\*

\* Esconde o modal de confirmação

\* @method hideConfirmDialog

\* @memberOf App#

\*/

this.hideConfirmDialog=function() {

$(".modal").addClass("hide");

$(".modal .generic").addClass("hide");

}

/\*\*

\* Exibe o modal de loading

\* @method showLoading

\* @memberOf App#

\*/

this.showLoading=function(){

self.closeAllModals();

$(".modal").removeClass("hide");

$(".loader").removeClass("hide");

}

/\*\*

\* Esconde o modal de loading

\* @method hideLoading

\* @memberOf App#

\*/

this.hideLoading=function(){

$(".modal").addClass("hide");

$(".loader").addClass("hide");

}

/\*\*

\* Inicializa a aplicação. Este método é executada quando começa a aplicação é chamada e também toda vez que ocorre mudança no hash da url (conteúdo após o sinal #)

\* @method inicializar

\* @memberOf App#

\*/

this.inicializar = function() {

document.addEventListener('deviceready', function () {

}, false);

Application.rotas();

// Checa login antes de desenhar a tela

if(localStorage.getItem("userId") && localStorage.getItem("token")) {

$("#name").html(localStorage.getItem("userName"));

$(".header-right").show();

} else {

window.location.href="login.html";

}

Application.desenharTela();

$("#lblnome").html(localStorage.getItem("userName"));

$(".sair").on("click",function(){

self.getOut();

});

}

/\*\*

\* Apaga o localstorage do usuário para deslogá-lo da aplicação. Depois é necessário redirecioná-lo

\* @method getOut

\* @memberOf App#

\*/

this.getOut=function(b) {

$.ajax(

{

"method":"get",

"url": self.service\_path+"logout",

"headers": self.getHeaders(),

"dataType": "json",

"success":function(data) {

if(!data.Error) {

// limpa e depois remove

window.localStorage.setItem("userId",null);

window.localStorage.setItem("userName",null);

window.localStorage.setItem("userType",null);

window.localStorage.setItem("servicepath",null);

window.localStorage.setItem("token",null);

window.localStorage.removeItem("userId");

window.localStorage.removeItem("userName");

window.localStorage.removeItem("userType");

window.localStorage.removeItem("servicepath");

window.localStorage.removeItem("token");

window.location.href="login.html";

} else {

alert(data.Error);

// limpa e depois remove

window.localStorage.setItem("userId",null);

window.localStorage.setItem("userName",null);

window.localStorage.setItem("userType",null);

window.localStorage.setItem("servicepath",null);

window.localStorage.setItem("token",null);

window.localStorage.removeItem("userId");

window.localStorage.removeItem("userName");

window.localStorage.removeItem("userType");

window.localStorage.removeItem("servicepath");

window.localStorage.removeItem("token");

window.location.href="login.html";

}

},

error:function(){

alert("Erro na requisição");

// limpa e depois remove

window.localStorage.setItem("userId",null);

window.localStorage.setItem("userName",null);

window.localStorage.setItem("userType",null);

window.localStorage.setItem("servicepath",null);

window.localStorage.setItem("token",null);

window.localStorage.removeItem("userId");

window.localStorage.removeItem("userName");

window.localStorage.removeItem("userType");

window.localStorage.removeItem("servicepath");

window.localStorage.removeItem("token");

window.location.href="login.html";

}

}

)

}

this.getHeaders=function() {

var token;

if(localStorage.getItem("token")){

token = localStorage.getItem("token");

return {"app-token":token};

} else {

return {}

}

}

/\*\*

\* Configura as rotas

\* @method rotas

\* @memberOf App#

\*/

this.rotas=function() {

var partes = window.location.href.split("#");

if(partes.length>1) {

Application.rota = partes[1];

Application.parametros = Application.rota.split("/");

console.log(Application.parametros);

Application.pagina=Application.parametros.shift();

console.log(Application.parametros);

//alert(Application.parametros[0]);

} else {

window.location.href="#home";

}

}

/\*\*

\* Insere o html da página selecionada pelo usuário

\* @method desenharTela

\* @memberOf App#

\*/

this.desenharTela=function() {

var profile=localStorage.getItem("userType");

// Verifica quais menus o usuário pode ver

if(profile=="tecnico" ) {

$("#menuinicio").show();

$("#menuusuarios").hide();

$("#menupesquisar").show();

$("#menuservicos").show();

$("#menudepartamentos").hide();

$("#menurelatorios").hide();

$("#menuabrir").hide();

} else if(profile == "administrador"){

$("#menuinicio").show();

$("#menuusuarios").show();

$("#menupesquisar").show();

$("#menuservicos").show();

$("#menudepartamentos").show();

//$("#menurelatorios").show();

$("#menurelatorios").hide();

$("#menuabrir").hide();

//$("#limportar").hide();

} else if(profile == "cliente") {

$("#menuinicio").show();

$("#menuusuarios").hide();

$("#menupesquisar").hide();

$("#menuservicos").show();

$("#menudepartamentos").hide();

$("#menurelatorios").hide();

$("#menuabrir").show();

}

var loadJs = function(page,callback) {

if($.inArray( page, Application.scriptslist )>=0) {

callback();

} else {

var script = document.createElement('script');

script.src = "js/"+page+".js"; // URL do seu script aqui

document.body.appendChild(script);

script.onload = function() {

callback();

// Coloca o script na lista de carregados após ter executado o callback, para que não função callback seja possível detectar se é primeira execução

Application.scriptslist.push(page);

};

}

}

var carregarPagina = function(page,callback) {

$.ajax(

{

"url":"pages/"+page+".html",

"type":"get",

"dataType":"html",

"success":function(data){

$(".content").html(data);

loadJs(page,function(){callback();});

}

}

);}

if(Application.pagina=="area-tecnico" || Application.pagina=="area-administrador" || Application.pagina=="area-cliente"){

Application.pagina = "home";

}

switch(Application.pagina) {

case "home":

var userType;

if(localStorage.getItem("userType")) {

userType = localStorage.getItem("userType");

if(userType=="tecnico") {

Application.pagina="area-tecnico";

} else if(userType=="administrador") {

Application.pagina="area-administrador";

} else if(userType=="cliente") {

Application.pagina="area-cliente";

}

}

carregarPagina(Application.pagina,function(){

//var home = new Home();

//home.open();

//Application.showModalError();

if(userType=="tecnico") {

areaTecnico = new AreaTecnico();

areaTecnico.open();

} else if(userType=="administrador") {

areaAdministrador = new AreaAdministrador();

areaAdministrador.open();

} else if(userType=="cliente") {

areaCliente = new AreaCliente();

areaCliente.open();

}

});

break;

case "abrir-chamado":

carregarPagina(Application.pagina,function(){

var abrirchamado = new AbrirChamado();

abrirchamado.open();

});

break;

case "cadastro-usuario":

carregarPagina(Application.pagina,function(){

var cadastrousuario = new CadastroUsuario();

cadastrousuario.open();

});

break;

case "cadastro-servico":

carregarPagina(Application.pagina,function(){

var cadastroservico = new CadastroServico();

cadastroservico.open();

});

break;

case "cadastro-departamento":

carregarPagina(Application.pagina,function(){

var cadastrodepartamento = new CadastroDepartamento();

cadastrodepartamento.open();

});

break;

case "chamado":

carregarPagina(Application.pagina,function(){

var chamado = new Chamado();

chamado.open();

});

break;

case "pesquisar-chamados":

carregarPagina(Application.pagina,function(){

var pesquisarchamados = new PesquisarChamados();

pesquisarchamados.open();

});

break;

case "usuarios":

carregarPagina(Application.pagina,function(){

var usuarios = new Usuarios();

usuarios.open();

});

break;

case "servicos":

carregarPagina(Application.pagina,function(){

var servicos = new Servicos();

servicos.open();

});

break;

case "departamentos":

carregarPagina(Application.pagina,function(){

var departamentos = new Departamentos();

departamentos.open();

});

break;

case "atribuir-chamado":

carregarPagina(Application.pagina,function(){

var atribuir = new AtribuirChamado();

atribuir.open();

});

break;

}

}

/\*\*

\* Chamar esta função nas páginas em que o usuário cliente não tem permissão de acessar

\* @method blockClientAccess

\* @memberOf App#

\*/

this.blockClientAccess=function() {

var type="";

var login = "";

var cookie=Application.getCookie();

if(cookie && cookie.userType ) {

type=cookie.userType;

login=cookie.userLogin;

if(type=="cliente" ) {

// Usuário cliente não tem permissão de acessar esta página

window.location.href="#home";

window.location.reload();

}

} else {

self.getOut();

}

}

}

var Application = new App();

Application.inicializar();

$(window).off('hashchange');

$(window).on('hashchange', function() {

//alert(window.location.href);

Application.inicializar();

});

$(".modal-box .close-modal").on("click",function(){

$(this).closest(".modal-box").addClass("hide");

$(".modal").addClass("hide");

});

/\*\*

\*@fileOverview Page Chamado.

\* @author Bruno Santos

\*

\*/

/\*\*

\* Chamado

\* @class Chamado

\* @classDesc Class that control the page 'Chamado '

\*/

function Chamado() {

var self=this;

var idchamado;

/\*\*

\* Method that is invoked by the application. This method makes the bind between DOM buttons/elements and the methods of this class (Criar). This method also makes other initial preparations for this page.

\* @method open

\* @memberOf Criar#

\*/

this.open=function() {

var id = Application.parametros[0];

self.idchamado=id;

$.ajax({

method: "GET",

url: Application.service\_path+"getchamado/"+self.idchamado,

dataType: "json",

headers: Application.getHeaders(),

success: function(result) {

console.log(result);

if(result.Error) {

alert("Erro ao abrir o chamado");

} else {

var data = result[0]

$("#lblcod").html(data["\_id"]);

$("#lbltitulo").html(data["titulo"]);

$("#lbldescricao").html(data["descricao"]);

$("#lblusuario").html(result[1]['nome']);

if(result[2] && result[2].nome) {

$("#lblresponsavel").html(result[2].nome);

} else {

$("#lblresponsavel").html("");

}

$("#lbldataabertura").html(data["dataAbertura"]);

$("#lbldataencerramento").html(data["dataEncerramento"]||"---");

if(data['encerrado']){

$("#lblencerrado").html('Sim');

} else {

$("#lblencerrado").html('Não');

}

if(data['encerrado']){

$("#descricao").prop("disabled",true);

$("#enviarresposta").prop("disabled",true);

$("#tipo").prop("disabled",true);

} else {

$("#descricao").prop("disabled",false);

$("#enviarresposta").prop("disabled",false);

//tipo cliente só pode enviar resposta

if(localStorage.getItem("userType")=='cliente'){

$("#tipo").val('resposta');

$("#tipo").prop("disabled",true);

} else {

$("#tipo").prop("disabled",false);

}

}

if(result[3] && result[3].nome) {

$("#lbldepartamento").html(result[3].nome);

} else {

$("#lbldepartamento").html('');

}

if(result[4] && result[4].nome) {

$("#lblservico").html(result[4].nome);

} else {

$("#lblservico").html('');

}

self.listarRespostas(data['respostas']);

}

},

error:function(a,b,c){

alert("Erro ");

console.log(a);

console.log(b);

console.log(c);

}

});

$("#enviarresposta").on("click",function(){

var descricao = $("#descricao").val();

var tipo = $("#tipo").val();

if(tipo == 'encerramento') {

var r = confirm("Tem certeza que deseja encerrar o chamado?");

if (r == true) {

console.log("chamado vai ser encerrado");

} else {

console.log('encerramento cancelado');

return false;

}

}

console.log("pós confirm");

var post\_data = {

"descricao": descricao,

"tipo": tipo

}

$.ajax({

url: Application.service\_path+'responderchamado/'+self.idchamado,

method: 'POST', //POST

headers: Application.getHeaders(),

dataType: "json",

data: post\_data,

success: function(data) {

console.log(data);

if(data.Error) {

alert("Não foi possível salvar os dados");

} else {

alert("Dados salvos com sucesso");

window.location.reload()

}

},

error:function(){

alert("Erro na requisição");

}

});

});

}

this.listarRespostas = function(list) {

var html = "";

$(".answers\_session.sessioncall").html("");

for(var i=0; i<list.length;i++){

var item = list[i];

var usuval = "";

if(item.nomeUsuario) {

usuval = item.nomeUsuario

} else {

usuval = item.usuario

}

html="<div class='respostaitem'><div><span class='anslbl'>Usuário: </span><span>"+usuval+"</span></div> <div><span class='anslbl'>Texto: </span><span>"+item.descricao+"</span></div> <div><span class='anslbl'>Data: </span><span>"+item['data']+"</span></div></div>";

$(".answers\_session.sessioncall").append(html);

}

}

}