CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA
FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA"
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

ÉRIC CORDEIRO DE SOUZA

SISTEMA MÓVEL DE ORDEM DE SERVIÇO

MARÍLIA
2012

CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA
FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA"
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

SISTEMA MÓVEL DE ORDEM DE SERVIÇO

Monografia apresentada ao Curso de Graduação em Sistemas de Informação, do Centro Universitário Eurípides de Marília, mantido pela Fundação de Ensino Eurípides Soares da Rocha, como requisito para obtenção do grau de Bacharel em Sistemas de Informação.

Orientador:
Profª. Ms. Ricardo José Sabatine.

Marília, 2012

CORDEIRO DE SOUZA, Eric
Sistema Móvel de Ordem de Serviço / Eric; orientador: Profª. MS. Ricardo Jose Sabatine. Marília, SP: [s.n.], 2012.
85 f.

Trabalho de Conclusão de Curso (TCC) - Centro Universitário Eurípides de Marília, Fundação de Ensino Eurípides Soares da Rocha.

1. Aplicação Móvel 2. iOS 3. GIS

CDD: 005.2

CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA
BACHARELADO EM SISTEMAS DE INFORMAÇÂO

TRABALHO DE CONCLUSÃO DE CURSO – AVALIAÇÃO FINAL

Éric Cordeiro de Souza

DESENVOLVIMENTO DE SISTEMA MÓVEL DE ORDEM DE SERVIÇO

Banca examinadora da monografia apresentada ao Curso de Bacharelado em Sistemas de Informação do UNIVEM/F.E.E.S.R., para obtenção do Título de Bacharel em Sistemas de Informação.

Nota: 10.0 (Dez)

Orientador: Ricardo José Sabatine — Ricardo Sabatine

1°. Examinador: Mauricio Duarte

2°. Examinador: Fábio Dacêncio Pereira

Marília, 05 de dezembro de 2012.

AGRADECIMENTOS

Agradeço a Deus,
Por ter me dado a sabedoria, a coragem e a força necessária para realizar esse sonho.

Aos meus Pais, Cléia e Fausto,
Por sempre apoiarem minhas decisões, por sempre estarem presentesacreditando na minha capacidade e torcendo por mim.

Às minhas irmãs, Micheli e Andréia,
Pela confiança depositada, por estarem presentese pelo apoio durante esses anos.

À minha Tia Marta e a minha amiga Cândida,
Pela motivação e apoio necessário para iniciar a realização desse sonho.

Ao meu Professor/Orientador Ricardo Sabatine,
Pela orientação deste trabalho, por seus ensinamentos, pela motivação e pelo conhecimento adquirido em sua disciplina do curso de sistema de informação

Aos professores do Curso de Sistema de Informação,
Pela importante participação no conhecimento adquirido ao longo da graduação.

Cordeiro de Souza, Eric.Sistema Móvel de Ordem de Serviço. Monografia (Bacharelado em Sistemas de Informação) – Centro Universitário Eurípides de Marília, Fundação de Ensino Eurípides Soares da Rocha, Marília, 2012.

RESUMO

Com a grande evolução da computação móvel e da Internet, surgiu a nova geração de celulares denominados *smarthpones*. A maioria desses dispositivos são produzidos com um GPS integrado que possibilita identificar a localização do dispositivo na superfície terreste. Embora esses dispositivos tenha fonte de energia e processamento limitado, existe uma grande variedade de aplicativos disponíveis tornando o *smartphone* indispensável no dia-a-dia de muitas pessoas. Com a união da tecnologia móvel, internet e GPS, é possivel desenvolver aplicações móveis que podem ser a solução para muitas empresas que precisam de mobilidade em seus sistemas. Considerando que essas tecnologias estão consolidadas, este trabalho tem como objetivo demonstrar a viabilidade da melhoria no tempo de execução de ordem de serviço através de um sistema móvel para o *smartphone* iPhone, explorando a plataforma iOS e o GPS. Esta aplicação será integrada com um sistema web que fará o gerenciamento das ordens de serviço e gestão de todos os processos envolvidos.

Palavras-Chave: Aplicação móvel, iOS, GPS, Sistema Web.

Cordeiro de Souza, Eric. Sistema Móvel de Ordem de Serviço. Monografia (Bacharelado em Sistemas de Informação) – Centro Universitário Eurípides de Marília, Fundação de Ensino Eurípides Soares da Rocha, Marília, 2012.

ABSTRACT

With the great evolution of mobile computing and the Internet, emerged a new generation of phones called smarthpones. Most of these devices are manufactured with an integrated GPS that allow identify the device's location of the terrestrial surface. Although these devices have an energy source and limited processing, there is a wide variety of *smartphone* apps available making essential on day-to-day lives of many people. With the merger of mobile technology, internet and GPS, it is possible to develop mobile applications that can be the solution for many businesses that need mobility in their systems. Whereas these technologies are consolidated, this project aims to demonstrate the feasibility of improving the time to complete service order through a mobile system for the iPhone *smartphone*, exploring the iOS platform and GPS. This application will be integrated with a web system that will manage service orders and management of all the processes involved.

Keywords: Mobile App, iOS, GPS.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE GRÁFICOS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

3G	Terceira Geração de Tecnologia de Internet Móvel
ACL	Access Control List
AGPS	Assisted Gloabl Positioning System
Ambiente Desktop	Ambiente de computador pessoal
API	Application Programing Interface
Buletooth	Tecnologia para transmissão de dados utilizando rede sem fio
CSS	Cascading Style Sheets
Chat	Sistema de bate-papo
CheckIn	Envio de coordenadas de localicação para uma aplicação
DER	Diagrama Entidade Relacionamento
Desktop	Computador Pessoal
Feed-back	Retorno de contato para analisar a qualidade do serviço prestado
GNU	Sistema Operacional Unix
GSM	Global System for Mobile Communications
GPS	Global Positioning System
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IDE	Integrated Development Environment
iOS	IPhone Operation System
JSON	Javascript Object Notation
Login	Acesso a um sistema informando dados de usuário
MVC	Model View Controller
Netbook	Computador móvel
Notebook	Computador móvel
PDA	Personal Digital Assistants
PHP	Hypertext Preprocessor
Rest	Representational State Transfer
RIA	Rich Internet Application
SDK	Software Development Kit
SOAP	Simple Object Access Protocol
Software	Programa de computador
Trigger	Procedimento executado após atualização de uma instância de Banco de Dados
Wi-fi	Tecnologia para transmissão de dados por rede sem fio
WPS	Wireless Provisioning Services

INTRODUÇÃO

A computação móvel surgiu como um paradigma que permite usuários carregarem seus dispositivos computacionais e os serviços associados sem perder conectividade com outras máquinas (COULOURIS*et al.*, 2007).

Segundo Coulouris (2007) quando se refere a dispositivos móveis, deve-se ter em mente os mais variados tipos, entre eles *notebooks*, *netbooks*, celulares, PDAs, *smartphones* e *tablets*, esses dispositivos aliados as combinações de conectividade sem fio, incluindo as tecnologias de telecomunicações como *WiFi*, GSM, 3G, entre outras, permite ao mais leigo, sem perceber, a utilização a qualquer momento e em qualquer lugar de um sistema de computação, através de um software.

Na tabela 1, é possível analisar o crescimento no número de habilitações de aparalhos celulares.

Tabela 1: Número de habilitações de aparelhos celulares (ANATEL, 2012)

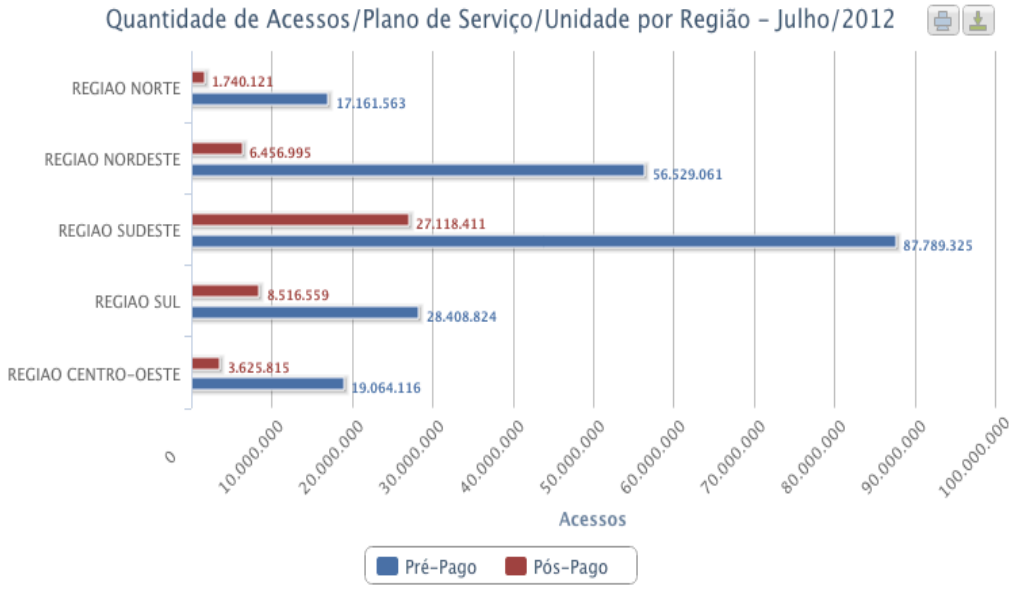
Ano	Habilitações	Crescimento (%)
2000	525.017	3,38
2001	326.258	1,38
2002	163.764	0,56
2003	411.175	1,17
2004	910.349	1,94
2005	811.101	1,22
2006	594.407	0,68
2007	469.568	0,47
2008	1.264.902	1,03
2009	415.909	0,27
2010	1.171.778	0,67
2011	2.415.235	1,18
2012	2.439.008	0,99

Além disso, alguns dispositivos móveis apresentam funcionalidades como GPS (*Global Positioning System*), câmeras fotográficas digitais, jogos eletrônicos, entre outros. Porém, esses dispositivos são limitados em sua fonte de energia (as baterias têm duração limitada) e em seus recursos computacionais (processamento, memória, interface, entre outros).

Com o avanço da Internet e a grande expansão da área de cobertura, surgiram novas tecnologias de conexão de internet como a 3G (terceira geração de tecnologia móvel) que possibilita navegação na internet em alta velocidade. Com essa velocidade de conexão é possível integrar sistemas móveis com diversos tipos de sistemas proporcionando ao usuário a liberdade de acessar seus *softwares* a qualquer hora em qualquer lugar.

A cada dia cresce o número de usuários que acessam internet a partir de *smartphones*. De acordo com a Anatel, houve um maior crescimento no acesso a partir de celulares pré-pagos, como é possível observar no Gráfico 1.

Gráfico 1: Número de acessos a internet móvel (ANATEL, Julho 2012)



BRASIL	Pré-Pago	Percentual (%)	Pós-Pago	Percentual (%)	Total
REGIAO CENTRO-OESTE	22.658.630	84,07	4.292.029	15,93	26.950.659
REGIAO NORDESTE	72.411.004	89,44	8.551.559	10,56	80.962.563
REGIAO NORTE	21.822.227	91,09	2.134.067	8,91	23.956.294
REGIAO SUDESTE	112.045.415	77,78	31.999.732	22,22	144.045.147
REGIAO SUL	38.520.120	78,33	10.658.347	21,67	49.178.467
TOTAL	267.457.396	82,27	57.635.734	17,73	325.093.130

Com o crescimento na quantidade de dispositivos conectados na Internet, surge a necessidade de adaptação das empresas para este novo tipo de tecnologia.

Este crescimento desperta o interesse de muitas empresas em atualizar suas plataformas e utilizar esse novo cenário de tecnologia que muitas vezes é a solução para alguns processos internos.

Ter um aplicativo móvel significa ter a informação sempre na mão para consultar a hora que precisar.

Empresas que possuem funcionários que trabalham externamente não têm controle sobre as tarefas que estes executam e nem sobre a sua carga horária. Este projeto propõe o desenvolvimento de um sistema móvel de ordem de serviço para a plataforma iOS do *smartphone* iPhone para que o funcionário execute suas tarefas fora do âmbito da empresa e um sistema web para que o administrador da empresa possa ter controle sobre as atividades que serão executadas.

Objetivos

O trabalho proposto tem como principal objetivo fornecer melhor controle para as empresas que possuem funcionários que executam tarefas fora da empresa, através da tecnologia móvel e da internet.

Com a conclusão desse trabalho, a empresa poderá localizar seus funcionários em tempo real, delegar e acompanhar as tarefas que estão sendo executadas além de facilitar a rotina diária dos funcionários que deixarão de utilizar papéis para controlar suas atividades e passarão a utilizar um *smartphone*.

Organização do Documento

Estruturalmente esta monografia é composta por cinco capítulos.

No capítulo 1 é feita uma revisão bibliográfica sobre aplicações móveis.

No capítulo 2 é feita uma análise dos principais ambientes de móveis.

No capítulo 3 são apresentados detalhes do desenvolvimento de aplicativos para iOS.

No capítulo 4 são apresentados os resultados obtidos.

No capítulo 5 é apresentada a conclusão do trabalho.

1 APLICAÇÃO MÓVEL

Fling (2009) define que aplicações móveis são aplicativos desenvolvidos para funcionar em dispositivos móveis e que apresentam várias características e limitações (por exemplo: processamento, memória, entre outros) que agregam funcionalidade aos usuários, como mobilidade, imediatismo e localização.

Os *smartphones* ganharam novos recursos através de milhares de aplicativos disponíveis para *download*, sendo que, a cada momento cresce o número de desenvolvedores de aplicativos para *smartphone* e a variedade de aplicações encontradas.

Entretanto, é importante realçar que aplicativos móveis não fornecem exatamente as mesmas capacidades das versões dos computadores de mesa, sendo importante conhecer as restrições, limitações e melhorias oferecidas para dispositivos móveis, sendo até necessário desaprender de algumas práticas habituais utilizadas no ambiente *desktop*.

Desenvolver aplicações para dispositivos móveis requer maior atenção e planejamento, pois estes dispositivos possuem memória, processamento e fonte de energia limitada além de possíveis problemas de conectividade.

Quando é necessário desenvolver uma aplicação móvel, existem diversos tipos de aplicações que são diferenciadas entre recursos, custo, tempo e forma de desenvolvimento.

1.1 Tipos de aplicações móveis

A proliferação de dispositivos móveis e plataformas de tecnologia representa uma mudança no ambiente de desenvolvimento em vários níveis. As empresas devem decidir não apenas o melhor uso estratégico de plataformas móveis, mas também a forma mais eficiente de implementação para os mesmos.

Essencialmente, os desenvolvedores podem escolher entre quatro abordagens de aplicação móveis diferentes (nativa, *web*, *middleware*, híbridas), cada um com seu

próprio conjunto de prós e contras (IBM WORKLIGHT, 2012; CHARLAND, LEROUX, 2011).

1.1.1 Aplicações Nativas

Aplicações Nativas (*Native App*) são aplicações desenvolvidas para uma plataforma específica (sistema operacional), ou seja, as aplicações só funcionarão no sistema para o qual foi desenvolvido (IBM WORKLIGHT, 2012; CHARLAND, LEROUX, 2011).

Normalmente são oferecidos pelo fabricante da plataforma ferramentas como SDK e linguagens de programação (por exemplo, Xcode/Objective-C para iOS, Eclipse/Java para Android, o Visual Studio/C# para Windows Phone).

Dentre as principais vantagens, destacam-se pode-se citar:

- Melhor acesso aos recursos do aparelho: Por se tratar de uma aplicação que utiliza a mesma linguagem do Sistema Operacional, a aplicação nativa consegue acessar recursos do dispositivo mais facilmente através das APIs do Sistema operacional como GPS, câmera, acelerômetro e bússola.
- Integração com aplicativos: Torna-se mais fácil fazer a integração dos aplicativos pois todos estes utilizam a mesma linguagem de programação e tem acesso aos mesmos métodos da API do Sistema Operacional.
- Aplicação com melhor desempenho: A aplicação tem melhor performance pois não é necessário um interpretador de outra linguagem para executar a aplicação.
- Armazenamento de dados: Com aplicações nativas é possível armazenar dados nos dispositivos móveis.

Dentre as principais desvantagens pode-se citar:

- Alto custo: Uma aplicação nativa tem alto custo pois exige conhecimentos específicos do sistema operacional no qual será desenvolvido.

- Processo de aprovação: As aplicações passam por um processo de aprovação. Somente se a aplicação passar no teste poderá ser disponibilizada para instalação.
- Escassez de profissionais qualificados: Ainda há uma dificuldade em encontrar profissionais qualificados para desenvolver aplicações nativas.

1.1.2 Aplicações Web

Aplicações *Web* (*WebApp*) são aplicações desenvolvidas utilizando a linguagem HTML, CSS e JavaScript, estando disponíveis por um servidor web e sendo executadas, por exemplo, pelo navegador do celular (IBM WORKLIGHT, 2012; CHARLAND, LEROUX, 2011).

Dentre as principais vantagens pode-se citar:

- Menor custo: Devido ao grande número de desenvolvedores web, torna-se fácil encontrar mão de obra qualificada e de baixo custo.
- Menor prazo de desenvolvimento: Aplicações Web são desenvolvidas mais rapidamente pois não é necessário conhecimentos específicos do dispositivo para qual será desenvolvido.
- Compatibilidade com diversas plataformas: Aplicações Web são compatíveis com diversos dispositivos móveis, basta possuir um navegador de internet que a aplicação poderá ser executada. Não é necessário retrabalho em código ou adaptação para o novo dispositivo.
- Não há processo de aprovação: Diferente das aplicações nativas, não é necessário que a aplicação passe por um processo de aprovação.

Dentre as principais desvantagens pode-se citar:

- Conectividade: A maioria das aplicações *web* necessitam de conectividade *web* para funcionar. O Brasil está em constante

crescimento no número de dispositivos móveis conectados na internet, porém, ainda há instabilidade no acesso.

- Acesso limitado a recursos do dispositivo: Este tipo de aplicação possui limitação ao acesso de recursos do Sistema Operacional e do dispositivo móvel como câmeras, GPS, acelerômetro e outros.

1.1.3 Híbrida

São aplicações que combinam código nativo do sistema operacional com outros códigos como, por exemplo, o HTML (IBM WORKLIGHT, 2012).

Para facilitar o desenvolvimento de aplicações híbridas existem diversos frameworks que facilitam a implementação como PhoneGap (PHONEGAP, 2012) e Titanium (TITANIUM, 2012).

Dentre as principais vantagens pode-se citar:

- Menor Custo: Possui um custo relativamente baixo ao custo de desenvolvimento de uma aplicação nativa.
- Compatibilidade com diversas plataformas: Não é necessário reescrever o código, pois os frameworks dão suporte a diversos tipos de sistemas operacionais móveis.
- Frameworks de Desenvolvimento: Existem diversos frameworks disponíveis na internet com documentação bem rica que facilitará o aprendizado.

Dentre as principais desvantagens pode-se citar:

- Acesso limitado a recursos: Embora aplicações híbridas mesclem código nativo com código HTML, as aplicações ainda têm limitações de acesso físico do aparelho ou métodos do sistema operacional.

1.1.4 *Middlewares*

Segundo Coulouris (2007), o *middleware* é uma camada de software que visa esconder a heterogeneidade entre os componentes de software e hardware do ambiente.

Portanto, os *middlewares* são utilizados para transferir dados de diferentes protocolos de comunicação e sistemas operacionais e têm como objetivo diminuir a complexidade do desenvolvimento, ou seja, ocultando do programador diferenças de protocolos de comunicação, plataformas e dependências do sistema operacional.

Uma aplicação móvel do tipo *middleware*, pode ser classificada como sendo uma aplicação móvel nativa integrada com um serviço *web* (SOAP ou RESTFul) para obter ou manipular os dados.

Dentre as principais vantagens pode-se citar:

- Pouco Processamento: Estas aplicações executam em segundo plano, isto é, não são visíveis para o usuário final e por esse motivo não são necessários componentes de interface visual economizando processamento e memória do dispositivo.

Dentre as principais desvantagens podemos citar:

- Limitação: Não é possível fazer grandes aplicações utilizando somente *middlewares*.

1.2 Desafios da computação móvel

O desenvolvimento móvel tem diversos obstáculos que devem ser analisados para garantir uma aplicação funcional e que tenha alta disponibilidade (COULOURIS *et al.*, 2007).

Dentre vários obstáculos, destacam-se os seguintes (COULOURIS *et al.*, 2007; LECHETA, 2009):

- Complexidade:Pelo fato de que os *smartphones* possuem hardware e fonte de energia limitada, o desenvolvedor, além de aprender linguagens de programação específicas do dispositivo, deve

desenvolver a aplicação da forma mais otimizada possível para garantir o maior tempo do uso de sua aplicação e disponibilidade dos serviços.

- Segurança: A segurança ainda é um dos assuntos mais questionados no desenvolvimento de aplicações móveis. Ao desenvolver uma aplicação, deve-se levar em consideração diversas características que quando abusadas, podem comprometer a segurança da informação (CERT BR - Segurança em dispositivos móveis).
- Perda ou furto: Por ser um dispositivo pequeno e de custo elevado, o *smartphone* os *smartphones* atraem a atenção de assaltantes. Tais informações podem ser utilizadas para novos furtos ou outras ações que envolvam o usuário.
- Aplicações com código malicioso que podem coletar informações pessoais sem autorização do usuário ou até mesmo tentar quebrar a segurança de outras aplicações.
- Diferentes Modelos de Hardware: As aplicações devem ser projetadas de forma que funcionem em diversos dispositivos sem existir uma grande diferença de performance.
- Resolução da Tela: A aplicação deve estar preparada para adaptar as diferentes resoluções de tela ou será necessário desenvolver versões diferentes com resoluções diferentes.
- Conexões de Rede: Ainda existe grande instabilidade nas conexões de rede (internet). A aplicação móvel não deve ser totalmente dependente da conexão de rede. É comum encontrar problemas de ausência de sinal e baixa velocidade de conexão
- Memória e Processamento limitados: A aplicação deve ter ciência dos recursos que estarão disponíveis no dispositivos de forma que não comprometa o funcionamento de outras aplicações no dispositivo.

1.3 APLICAÇÕES MÓVEIS DE GEOPOSICIONAMENTO

O GPS (*Global Position System*) revolucionou a forma de localizar pessoas através de todo o mundo. Sem dúvida, essa tecnologia combinada com dispositivos móveis pode ajudar na comunicação e também na resolução de diversos problemas de pessoas e empresas, como o problema abordado nesse trabalho.

As seções seguintes descrevem sistemas de localização global como GPS e algumas aplicações móveis que fazem uso de geoposicionamento.

1.3.1 GPS

O GPS é um sistema que informa a localização de um determinado dispositivo na superfície Terrestre utilizado diariamente por diversas pessoas através de dispositivos móveis (MONICO, 2000).O nome oficial é NAVSTAR e foi inicialmente desenvolvido para fins militares, tornando-se mais tarde aberto ao público.

Os aparelhos GPS são auxiliados por 24 satélites posicionados na órbita terrestre. Estes satélites fazem a triangulação do sinal do dispositivo, isto é, seu funcionamento consiste em medir a distância do dispositivo em relação a três satélites.A posição relativa é determinada obtendo distância que separa esses três pontos através da intersecção de três circunferências cujos raios são as distâncias medidas entre o receptor e os satélites (MONICO, 2000).

Com um receptor de GPS obtém-se todas as coordenadas terrestres, latitude, longitude e altitude, e ainda outras grandezas como a velocidade e intervalos de tempo.

Exemplo de funcionamento GPS através dos satélites na órbita terrestre como pode ser observado na Figura 1.

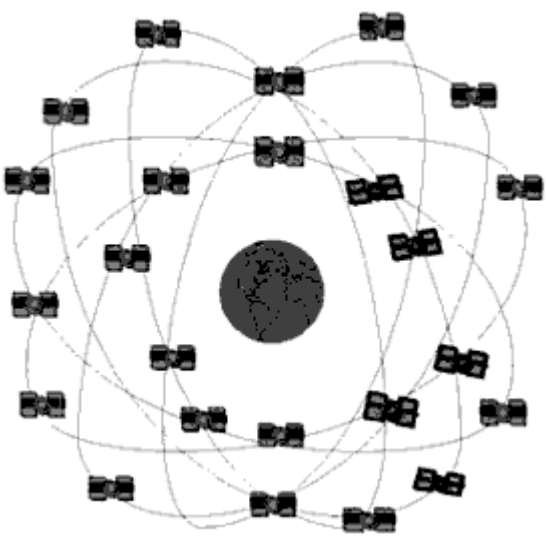


Figura 1– GPS

Fonte: MONICO, 2000, pp 23.

1.3.2 A-GPS

O AGPS (GPS Assistido) é uma versão melhorada do GPS e foi criada para localizar os satélites com mais rapidez e confiabilidade utilizando conexões de internet como 3G e GPRS para receber dados que auxiliam na geração das coordenadas.

O A-GPS primeiramente se comunica com uma antena de celular que por sua vez já possui a localização dos satélites retornando a resposta mais rapidamente.

1.3.3 Exemplo de Aplicações

Existem grandes empresas que desenvolveram aplicações móveis que utilizam geoposicionamento. Com a tecnologia GPS e o crescimento constante do número de dispositivos móveis, essas empresas vem utilizando esse cenário para construir tecnologias que ajudam no dia a dia das pessoas de todo o mundo.

É muito comum encontrar pessoas nas ruas utilizando diversos aplicativos e tornando seu uso uma rotina diária.

Dentre os diversos aplicativos que tiveram sucesso, pode-se destacar o Google Maps, Google Latitude, FourSquare, entre outros.

Google Maps

O Google Maps (GOOGLEMAPS, 2012) é uma tecnologia utilizada para apresentar tráfego ao vivo, rotas de trânsito e imagens no nível de rua utilizado por diversas empresas e websites na internet. Essa tecnologia surgiu em 2004 quando o Google comprou a empresa de mapeamento digital Keyhole.

Google Latitude

O Google Latitude (GOOGLELATITUDE, 2012) é uma ferramenta que permite que uma pessoa localize seus amigos em tempo real através de um sistema GPS. A aplicação exibe um mapa e a localização de cada amigo podendo existir interação entre eles através de mensagem instantânea. Foi lançado no Mercado em Fevereiro de 2009 com versões para *smartphones* com sistema operacional Android, Blackberry, Symbian S60 e Windows Mobile. A versão para iPhone foi lançada em Julho de 2009.

Em Fevereiro de 2011 o Google já possuía 10 milhões de usuários ativos utilizando sua plataforma Google Latitude (GOOGLELATITUDE, 2012).

FourSquare

O Foursquare (FOURSQUARE, 2012) é um aplicativo móvel que ajuda pessoas a encontrar os melhores lugares como hoteis, *shoppings* e restaurantes próximos a pessoa no momento.

Utilizando a tecnologia de GPS, o Foursquare permite salvar lugares onde tenha visitado, buscar por lugares de acordo com seus interesses e ver dicas de pessoas e amigos referentes a experiência obtida naquele lugar.

A primeira versão do Foursquare foi lançada em Fevereiro de 2008 e em fevereiro de 2012 já contava com mais de 25 milhões de pessoas utilizando sua plataforma móvel. (FOURSQUARE, 2012)

1.4 Ferramentas de Desenvolvimento para Dispositivos Móveis

Entre tantos dispositivos móveis e diversas linguagens de desenvolvimento, surge a dúvida entre qual linguagem escolher para desenvolver suas aplicações.

Para escolher a linguagem, deve-se pensar na evolução da aplicação, isto é, se a linguagem escolhida está em constante evolução, se há segurança e também se há material de suporte como wiki, comunidades e outros.

1.4.1 J2ME

O J2ME (*Java Plataform Micro Edition*) (JAVAME, 2012), também conhecido com JAVA ME e um ambiente para executar aplicações em JAVA para dispositivos móveis. Ele foi criado especificamente para dispositivos com baixa memória, vídeo e capacidade de processamento limitados.

O J2ME apresenta suporte para uma amigável e flexível Interface Gráfica com Usuário (GUI), um modelo de segurança robusto, um conjunto de protocolos de redes embutidos (JAVAME, 2012). O JAVA continua presente em aplicações móveis, mas também tem forte atuação em sistemas para Internet. Sua principal vantagem é a portabilidade, isto é, sistemas desenvolvidos em JAVA podem ser executados em qualquer dispositivo.

1.4.2 Android

O Android (ANDROID, 2012) é uma plataforma para tecnologia móvel completa *open source*(com um sistema operacional, um *middleware* (camada intermediária de aplicação) e aplicativos) desenvolvido pela *Open Handset Alliance* liderado pela Google Inc.

Foi apresentado em Novembro de 2007 e é desenvolvida com base no Kernel GNU Linux.

Na plataforma Android o desenvolvimento de aplicativos pode ser feito utilizando um kit de desenvolvimento (SDK) que disponibiliza as ferramentas e APIs necessárias para desenvolver aplicações para a plataforma Android (como por exemplo Eclipse/MotoDev) utilizando a linguagem Java (LECHETA, 2009). Os testes das aplicaçõespodem ser simuladas através do Android Simulator.

Os programas em Java são executados na máquina virtual otimizada para dispositivos móveis denominada Dalvik.Possui suporte para diversas funcionalidades de *hardware*, dentre as principais pode-se destacar o suporte para câmera, GPS, bússola, acelerômetro, Bluetooth, WiFi, 3G, suporte para diversos formatos de áudio e vídeo, para banco de dados, entre outros (LECHETA, 2009).

1.4.3 iOS

O sistema operacional iOS (Iphone OS) da Apple Inc é um dos mais populares pois revolucionou a forma do usuário interagir com um aparelho celular e em Dezembro de 2012 está na versão 6.0 (APPLE iOS, 2012)

Devido ao grande sucesso do iOS combinado com o poderoso hardware do iPhone, a Apple está crescendo no mercado móvel se tornando a terceira maior vendedora de celulares no mundo com 8,7% de todas as vendas globais tendo crescimento de 128,4% em base anual (IDC, 2012).

O iOS foi desenvolvido exclusivamente para o iPhone, mas devido ao grande sucesso, também é utilizado no *tablet* iPad.

2. DESENVOLVIMENTO DE APLICAÇÕES PARA IOS

Aplicativos nativos para iOS são desenvolvidos em uma linguagem de programação chamada Objective-C e recebem suporte de uma biblioteca chamada Cocoa Touch, sendo que grande parte de seu trabalho será feito pela IDE chamada Xcode (FAIRBAIRN *et al.* 2012).

Neste capítulo será apresentado as tecnologias relacionadas para o desenvolvimento de aplicativos nativos para o sistema operacional iOS.

2.1 Objective-C

O Objective C é uma linguagem de programação orientada a objetos baseada em Smalltalk, sendo utilizada para desenvolvimento em dispositivos que utilizam o sistema operacional iOS ou MacOS.

Segundo Fairbairn (2012), esta linguagem foi construída sobre a linguagem C e possui uma sintaxe bem diferenciada das principais linguagens de programação existentes, pois, o modelo de programação orientada a objetos tem por base o envio de mensagens a objetos, diferente de linguagem como C++ e Java, que chama métodos diretamente em um objeto.

O Objective-C permite a implementação de lógica condicional e criação de loops, mas não oferece suporte inerente à interação com o usuário, ao acesso de recursos em rede ou à leitura de arquivos. Para isso a Apple incluiu no SDK um conjunto de bibliotecas de suporte conhecidas coletivamente como Cocoa Touch.

Para desenvolver aplicativo iOS na linguagem Objective-C, utiliza-se a IDE Xcode, disponível apenas para plataforma MacOS.

Classe

As classes em Objective-C são definidas em duas partes: arquivo de cabeçalho e arquivo de implementação.

Um arquivo cabeçalho (.h) é uma interface que declara os métodos e propriedades da classe e define as heranças. Nessa interface deve-se declarar os elementos essenciais para execução programa como a inclusão de frameworks e classes auxiliares.

A Figura 2 ilustra estrutura básica de um arquivo de cabeçalho.

```
@interface NomeDaClasse: SuperClasse
{
        Declaração_de_variáveis;
}

Declaração_de_métodos;

@end
```

Figura 2 – Estrutura de classe em Objective-C - arquivo de cabeçalho

Na Figura 3 é possível observar um exemplo, neste foi criado uma interface MinhaClasse que extende a classe NSObject, dentro das chaves defini-se as variáveis e fora das chaves, definos o encapsulamento das variáveis utilizando a diretiva @property e os métodos.

```
@interface Pessoa: Object
{
NSString *nome;
NSString *endereco;
}

@property (nonatomic, retain) NSString *nome;
@property (nonatomic, retain) NSString *endereco;

// Métodos
-(void)realizaSqrt:(int)valor;
+(BOOL)verificaAcao;

@end
```

Figura 3 – Exemplo de classe em Objective-C - arquivo de cabeçalho

Fonte: Marzullo, 2012 pp 26

A declaração de @property foi criada para facilitar o encapsulamento das varíaveis. Ao declarar um @property basta utilizar o @syntetize na classe que implementação, e a classe cuidará dos métodos Getters e Setters deixando o código fonte mais fácil de ser compreendido.

O acesso as variáveis pode ser modificado utilizando as seguintes opções:

- **readwrite:** Define que o acesso a variável será de leitura e escrita. Esta propriedade já é definida como padrão ao criar os @property.
- **readonly:** Definindo essa propriedade, a variável será apenas de leitura.
- **assign:** É utilizado em valores escalares e especifica que o método seter trate a atribuição de forma simples.
- **retain:** Definindo essa propriedade, o método fará a atribuição do novo valor ao objeto, mais antes, ele armazenará uma referência do objeto sendo possível recuperar o valor armazenado anteriormente.
- **copy:** Ao invés de criar uma referência com o "retain", o copy envia uma mensagem avisando o objeto original que o mesmo deve ser liberado. Ao utilizar essa propriedade é necessário implementar o protocolo NSCopying.
- **nonatomic:** Especifica que o objeto não poderá ser processado em ambiente de múltiplos processadores (não atômico).

Arquivo de implementação é um arquivo .m que implementa a classe de cabeçalho (.h).

A Figura 4 ilustra a estrutura básica de implementação do arquivo de implementação.

```
@interface NovaClasse
//Definição_de_métodos;
@end
```

Figura 4 - Estrutura de classe em Objective-C - arquivo de implementação

Todos os métodos declarados na classe de cabeçalho deverão ser implementados na classe de implementação e todas as variáveis que foram encapsuladas devem utilizar a diretiva @synthesize para que os métodos *getters* e *setters* sejam ativados.

Na Figura 5 é possível observar um exemplo de implementação de arquivo de implementação.

```
@implementation Pessoa

@synthesize nome;
@synthesize endereco;

// Métodos
-(void)realizaSqrt:(int)valor{
       //implementação do método

}
+(BOOL)verificaAcao{
//implementação do método
}

@end
```

Figura 5 - Exemplo de classe em Objective-C - arquivo de implementação

Fonte: Marzullo, 2012 pp 26

A Figura 6 ilustra a sintaxe de instanciação de objetos.

```
Pessoa *p = [[Pessoa alloc] init];

p.nome = @"Lara";

p.endereco = @"Rua das Nuvens";

...
}
```

Figura 6 - Exemplo de instanciação em Objective-C - arquivo de implementação

Fonte: Marzullo, 2012 pp 26

Método

Métodos em Objective-C são declarados seguindo a seguinte estrutura:

[tipoAcesso] ([tipoDeRetorno]) nomeDoMetodo : ([tipoParametro]) variável

- tipoAcesso: pode ser identificado com um traço(-) indicando que é um método de instância, ou pode ser identificado com um sinal de mais(+) quando for uma instância de classe, ou seja, não é necessário instanciar a classe para invocar o método.

- tipoDeRetorno: tipo de variável que o método vai retornar como por exemplo NSString, BOOL entre outros.
- nomeDometodo: deve ser informado o nome da chamada do método. Recomenda-se a inserção de nomes que representem oque o método fará como boas práticas de programação.
- tipoParametro: Caso o método possua uma assinatura, devemos informar nessa propriedade.
- variável: Nome da variável que será passada com referência para ser utilizada dentro da estrutura do método.

A Figura 7 demonstra um exemplo de implementação de método.

```
-(NSString)verificaMedia:(int)nota{
       if(nota>=7)
              return @"Você esta aprovado";
       else
              return @"Você esta reprovado";
}
```

Figura 7 - Exemplo de método em Objective-C

2.2Arquitetura iOS

A arquitetura do iOS é semelhante encontrada no Mac OSX, está divida em 4 camadas: Cocoa Touch, Media, Core Services e Core OS (APPLE 2012; MILANI, 2012).

A Figura 8 é demonstrado a arquitetura do iOS.

Figura 8 - Camadas da plataforma iOS

Fonte: MILANI, 2012 pp 15

Camada CORE

A camada CORE é responsável pelo gerenciamento dos recursos do dispositivo como bateria, luminosidade, memória, segurança, comunicação com *hardware* externo e outras funções.

Camada CORE SERVICES

A camada CORE SERVICES cuida das funcionalidades relacionadas ao telefone como o controle das chamadas recebidas, trocar mensagens de texto SMS, iCloud *Storage* e também controla protocolos de comunicação de rede e banco de dados SQLITE (SQLITE, 2012).

Camada MEDIA

A camada MEDIA cuida de todos os efeitos sonoros produzidos pelo iPhone e também da parte gráfica que utiliza OpenGL ES ou Quartz, sendo ambas utilizadas para a criação de aplicações gráficas como jogos.

Camada COCOA Touch

A camada Cocoa Touch é um conjunto de *frameworks* para desenvolvimento do iOS e construído com foco na interface do usuário e otimização das aplicações.

Cocoa usa o padrão de projeto Model-View-Controller (MVC) como é possível observar na Figura 9, ondeo modelo (*model*) encapsula os dados da aplicação, a visão (*views*) exibe e edita os dados e o controlador (*controller*) gerencia a lógica entre os dois.

O objetivo de separar as responsabilidades desta maneira é criar um aplicativo que é mais fácil de projetar, implementar e manter (COCOA, 2012; MARZULLO, 2012).

Segundo Marzullo (2012), no padrão de projeto Model-View-Controller a interação com o usuário é feita por intermédio da visão, para informar ao controlador o que o usuário fez alguma ação temos dois elementos os *outlets* (canal de

comunicação entre controlador e a visão (componente)) e os alvos (um ponto no controlador onde à visão pode notificá-lo de uma interação).

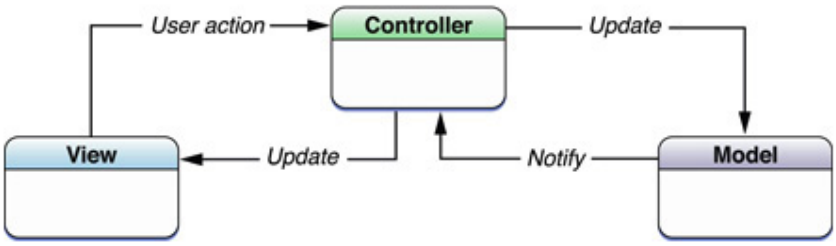


Figura 9 – MVC
Fonte: Apple Core Audio, 2012

Dos diversos *frameworks* inclusos, dos principais *frameworks* utilizados em aplicativos para iPhone, é possível destacar o Foundation Kit, UIKit, entre outros (FAIRBAIRN, 2012).

O Core Location é utilizado para determinar a localização geográfica do dispositivo. Fornece a localização através de Latitude e Longitude.

O Core Animation é utilizado para reenderizar a parte gráfica da aplicação com efeitos avançados de composição. Com o Core Animation é possível criar interfaces dinâmicas sem a necessidade de se programar em baixo nível como utilizando a API Gráfica OpenGL.

O Core Audio foi projetado para lidar com as necessidades de áudio do aplicativo fornecendo alto desempenho com baixa latência. É uma camada que fica acima da camada de abstração de *hardware* (HAL).

Core Data: É um *framework* utilizado para persistir os dados no iPhone e está disponível para iOS 3.0 ou superior

O Foundation Kit é uma coleção de classes de estruturas de dados, recursos de rede, entrada e saída de arquivos, data, hora e funções de tratamento de strings.

Core UIKit é um *framework* projetado para auxiliar no desenvolvimento de GUIs com animações elaboradas. Trabalha a parte visual do aplicativo, portanto, permite a interação do usuário através de toques e também controla a câmera do dispositivo, como é possível observar na Figura 10.

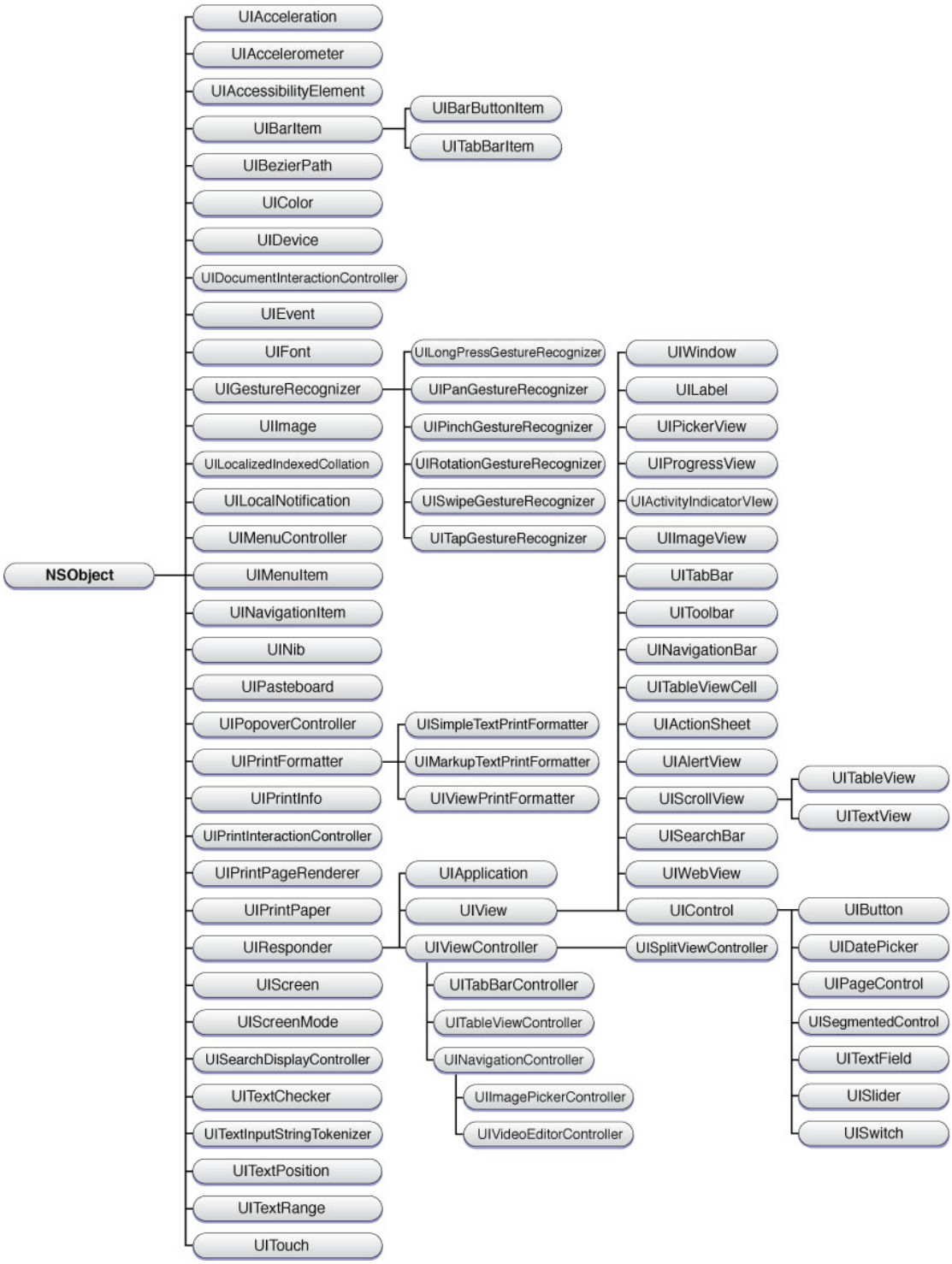


Figura 10 – UIKit

Fonte: Apple UI Kit 2012

2.3 Xcode

O Xcode é o ambiente de desenvolvimento integrado (IDE) projetado para o desenvolvimento de aplicações para iOS e Mac. O IDE Xcode inclui um editor de código fonte que pode mostrar erros em ambos, sintaxe e lógica, e até sugere correções e um editor de interface do usuário (Interface Builder) (MILANI, 2012).

Segundo Milani (2012), o Xcode combina a potência de um ambiente de desenvolvimento baseado em Unix com uma interface simples, elegante e poderosa possibilitando a criação de aplicações mais rapidamente, porém, o mesmo fica limitado para desenvolvedores que possuam um computador da Apple.

Uma ferramenta que compõem o Xcode é emulador de iPhone e um emulador de iPad, chamado de iOS Simulator que permite o teste das aplicações, entretanto os recursos que dependentes de hardware (por exemplo: acelerômetro e o giroscópio) não estão disponíveis.

A Figura 11 ilustra a IDE Xcode.

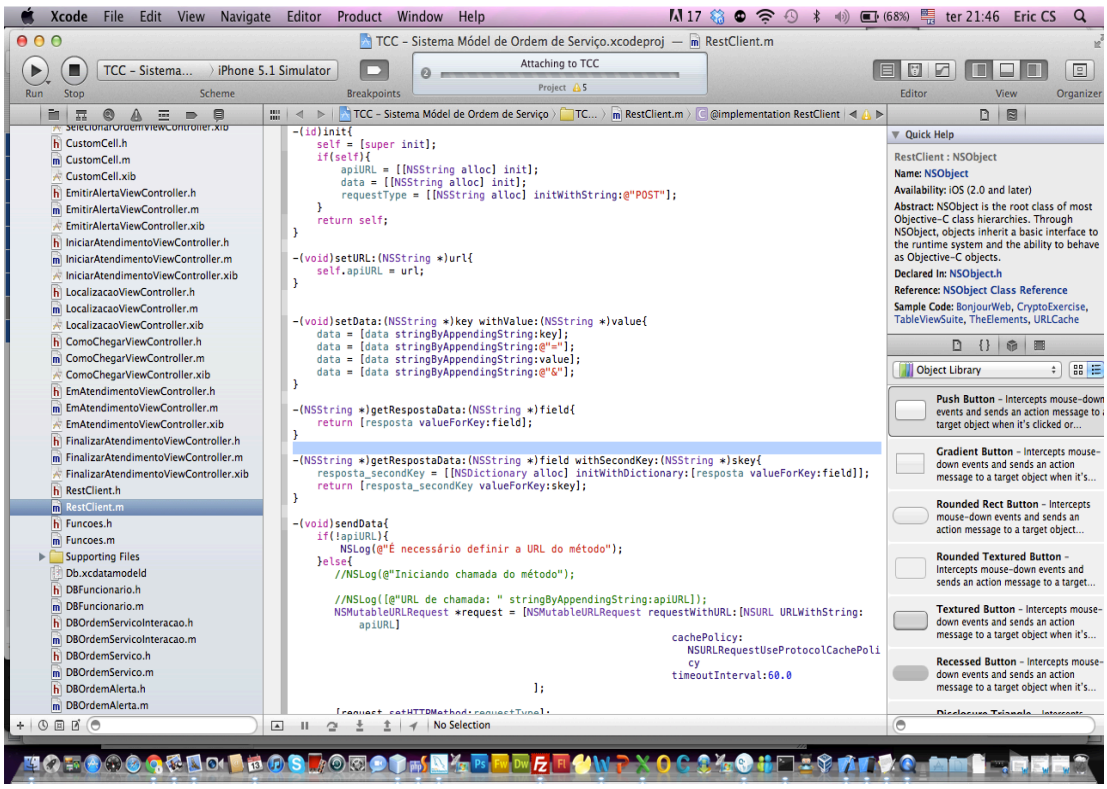


Figura 11 - Interface do Xcode

2.4Interface Builder

O Interface Builder permite criar interfaces para iOS sem escrever código, isto é, apenas arrastando botões, tabelas e outros componentes para a interface. É possível inserir os itens, definir seus atributos e relação entre eles.

A interface de usuário para um aplicativo iOS é armazenada em um arquivo de formato ".xib". O arquivo .xib é um formato baseado em XML, que são carregados quando a aplicação é iniciada possibilitando que o framework Cocoa use essas informações para criar os botões em tempo de execução.

Segundo Fairbairn (2012), é normal encontrar referência a arquivos ".nib" em vez de ".xib", o arquivo de extensão ".nib" representa um formato binário mais antigo, que resulta em arquivos menores e com maior velocidade de processamento. O Xcode converte automaticamente arquivos *.xib para o formato *.nib.

A Figura 12 ilustra o Interface Builder na IDE X-Code.

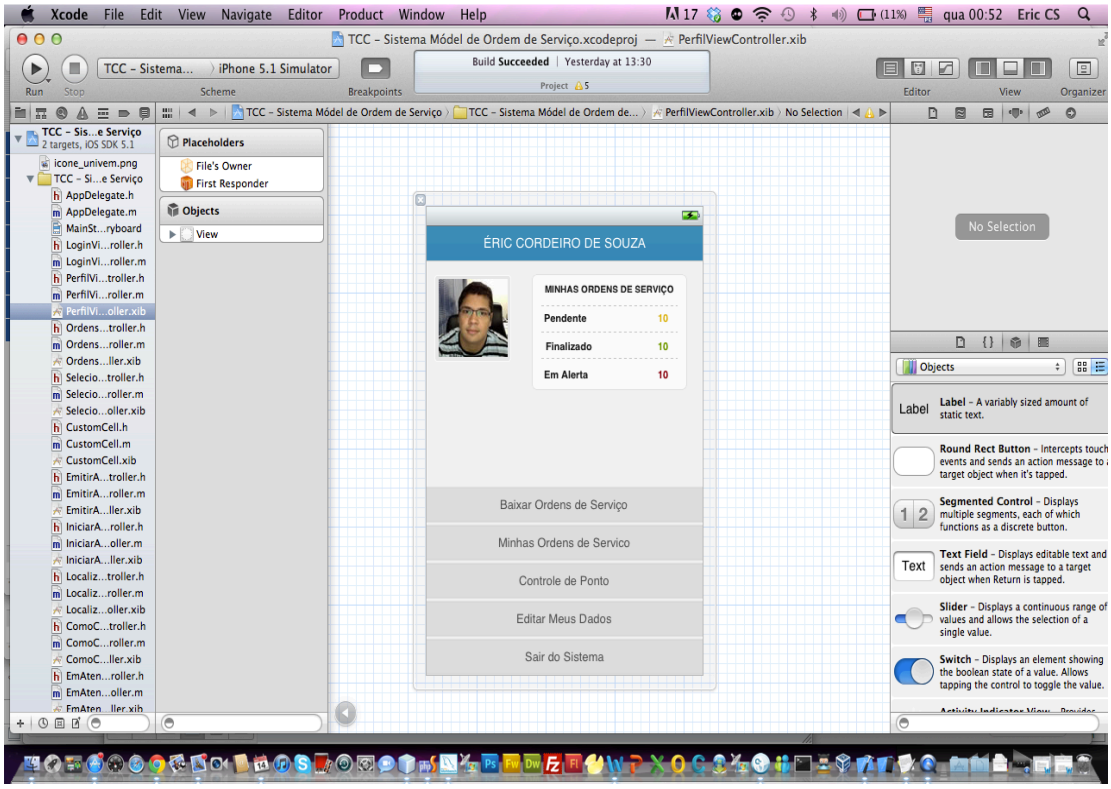


Figura 12 - Interface Builder na IDE X-Code

2.5 Licenças de desenvolvimento

Segundo Milani (2012), para o desenvolvimento de aplicativos para iOS existem três modos de interagir com as aplicações e licenças. Para publicar na AppStore será necessário passar por um processo de aprovação das regras da Apple.

- Sem nenhuma licença comercial: poderá utilizar o Xcode e testar somente no iOS Simulator. Além disso, não poder instalar as aplicações em seus dispositivos físicos reais e nem publicá-las na AppStore (MILANI, 2012).
- Licença comercial Standard: A licença Standard permite a publicar na AppStore e a instalação nos dispositivo móvel real ao custo de US$ 99 anuais (MILANI, 2012).
- Licença comercial Enterprise: mesmos benefícios da licença Standard, porém, permite habilitar mais de um desenvolvedor a fim de utilizá-la ao custo de US$ 299 anuais (MILANI, 2012).

3. DESENVOLVIMENTO DO APLICATIVO

Neste capítulo será apresentado detalhes de implementação das aplicações propostas.

Diversas empresas atualmente possuem funcionários que passam a maior parte do tempo prestando serviços externos. Esta empresa não tem controle sobre as tarefas que seus funcionários executam e nem sobre a carga horária de trabalho.

Além destes problemas, de acordo com a Portaria nº 3.626, de 13/nov./91 do Ministério do Trabalho e Previdência Social (MTPS, 2012), dispõe que quando a jornada de trabalho for executada integralmente fora do estabelecimento do empregador, o horário de trabalho constará também de ficha, papeleta ou registro de ponto, que ficará em poder do empregado. Dessa forma, em uma reclamatória trabalhista a empresa não tem prova a seu favor para comprovar a efetiva jornada de trabalho do funcionário que poderá apresentar uma ou duas testemunhas (normalmente é um colega que também foi despedido) e afirmar que trabalhava além do horário contratado, sendo exigidas horas extraordinárias.

Com isso, a criação de um sistema móvel de ordem de serviço pode amenizar esses efeitos, ajudando o administrador da empresa a ter ciência da localização do seu funcionário em tempo real, qual cliente está sendo atendido e qual a carga horária de trabalho desse funcionário.

Para atender esse cenário, foram desenvolvidos os seguintes sistemas:

- Sistema Web: Sistema que será utilizado pelo Administrador da empresa para gerenciar funcionários, clientes e ordens de serviço.
- Sistema Móvel: Sistema que será utilizado pelo funcionário da empresa que executará tarefas externamente.
- Serviço de integração: Um serviço baseado no estilo de arquitetura REST utilizado para integração entre a aplicação móvel e a aplicação web.

A Figura 13 ilustra a interação entre as aplicações e o ambiente proposto de uma maneira geral.

Figura 13 – Ambiente proposto para o sistema móvel de ordem de serviço

3.1 Aplicação Web

Com o constante crescimento da internet, diversas empresas estão migrando suas soluções para sistemas *web*. Este novo modelo de sistema possui maior disponibilidade, pois os servidores estão conectados a links redundantes, menos custo com infraestrutura e podem ser acessados de qualquer dispositivo com conexão e navegador de internet.

O sistema *web* foi desenvolvido utilizando a linguagem PHP, com framework CakePHP, seguindo o padrão MVC com banco de dados MySql.

O PHP é uma linguagem orientada a objetos, está na versão 5.4.3 (PHP, 2012).

A interface *web* foi criada a partir de conceitos de RIA (*Rich Internet Applications*) utilizando o tema GeboAdmin, disponível para comercialização. O tema GeboAdmin apresenta uma interface padrão, em inglês, sendo, a mesma, traduzida e adaptada para o framework CakePHP e modificada com novas funcionalidades de interface para garantir uma boa experiência do usuário seguindo conceitos de usabilidade.

O sistema *web* foi desenvolvido com as funcionalidades divididas em módulos: cliente, funcionário, ordem de serviço e relatórios. Os módulos e suas respectivas funcionalidades serão descritas na sequência.

Cliente

Permite que o administrador cadastre, edite e atualize registros de clientes no sistema. Também é possível visualizar informações de contato do cliente, serviços utilizados e localização de sua residência através de um mapa. O sistema automaticamente detecta as coordenadas (latitude e longitude) do endereço informado no momento do cadastro através da API do Google Maps.

Funcionário

Permite que o administrador cadastre, edite ou atualize as informações de seus funcionários. É possível adicionar funcionários a partir de cadastro de cliente evitando assim duplicidade de informações no banco de dados.

A funcionalidade Localizar Funcionário permite que o administrador saiba a localização de seu funcionário sendo possível visualizar através de um mapa.

Serviços

Permite que o administrador adicione, edite e atualize serviços que sua empresa disponibiliza aos seus clientes.

A funcionalidade Serviços para Clientes permite adicionar serviços contratados ao cadastro do cliente podendo futuramente prestar atendimento a este serviço através de uma ordem de serviço.

Na funcionalidade Ordem de Serviço o administrador poderá atribuir ordens de serviço para seus funcionários informando o cliente a ser atendido, a data e horário que será feito o atendimento e detalhes sobre a necessidade do atendimento.

A funcionalidade Alerta de Ordem de Serviço permite que o administrador seja notificado quando um funcionário, por algum motivo, não puder prestar atendimento para um cliente. Ao receber a notificação, o administrador pode reagendar a ordem de serviço e entrar em contato com o cliente.

Relatório

Relatório Tempo em Trânsito: Permite que o administrador visualize quanto tempo está sendo gasto em trânsito pelo funcionário para chegar até a residência do seu cliente.

Relatório Tempo em Atendimento: Permite que o administrador visualize o tempo utilizado no atendimento de seus clientes.

Relatório Controle de Ponto: Permite visualizar quantas horas de serviço um determinado funcionário trabalhou por mês e o horário de início e término da sua jornada de trabalho diária.

Relatório Ordem de Serviço: Exibe a quantidade de ordens de serviço atendidas no mês.

Banco de Dados

O sistema gerenciador de banco de dados utilizado na aplicação é o MySql. O Mysql é um gerenciador de banco de dados relacional de alto desempenho muito utilizado em aplicações *Web*.

Para diminuir a complexidade do desenvolvimento, algumas rotinas foram implementadas diretamente no banco de dados através de Triggers e Procedures minimizando a possibilidade de erro no processo e maximizando a performance.

As seguintes funcionalidades foram implementadas através de Triggers:

1) A tabela "ordem_servico_interacaos": Essa tabela armazenará todas as interações da ordem de serviço, isto é, todo o histórico do atendimento de uma ordem de serviço.

- A trigger "finaliza_interacao" é executada a cada atualização da tabela (AFTER UPDATE) e seta automaticamente o tempo gasto em trânsito e o tempo gasto no atendimento do cliente calculando o espaço do tempo entre os check-ins.

Esta trigger também atualiza o relatório de horas diárias do funcionário para que o administrador possa saber quanto tempo de atendimento ou quanto tempo em trânsito o usuário utilizou em um determinado dia.

A trigger também atualiza o status da ordem de serviço para "finalizado" ou "reagendar" de acordo com a informação da interação.

2) Tabela "relatorio_hora_dias": Esta tabela tem como objetivo armazenar o tempo que o funcionário esteve em trânsito e em atendimento em um determinado dia.

- A trigger "insere_atualizarelatoriomes" é executada após a inserção de dados na tabale relatorio_hora_dias. Esta trigger executa uma procedure que atualiza a tabela "relatorio_hora_mes".

- A trigger "atualiza_atualizarelatoriomes" é executada após cada atualização na tabela relatório_hora_dias. Esta trigger tem funcionamento similar a trigger de inserção, porém, ela remove o tempo anterior da tabela "relatorio_hora_mes" e executa a procedure para atualizar os dados.

3) Tabela ordem_servico_alertas: Esta tabela tem como objetivo inserir os alertas nas ordens de serviço, isto é, quando por algum motivo o funcionário não comparecer ao atendimento, esta tabela será populada com a justificativa.

- A trigger "atualiza_ordem" é executada a cada inserção na tabela ordem_servico_alertas e tem como objetivo mudar o status da ordem de serviço para "Alerta". Dessa forma o administrador verá que foi emitido um alerta e tomará as decisões necessárias.

A Figura 14 ilustra o Diagrama Entidade/Relacionamento da aplicação *web* em questão.

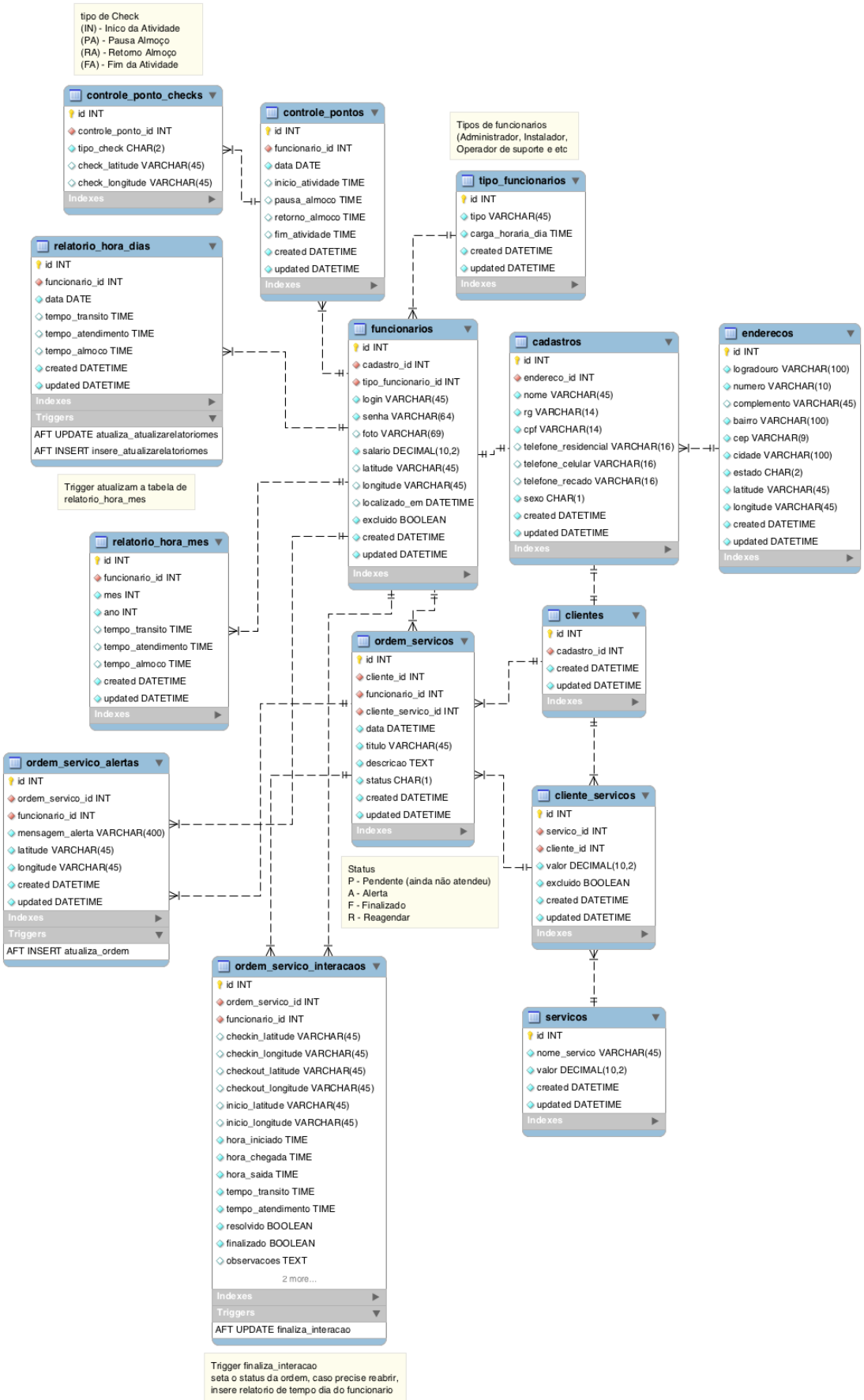


Figura 14 - Modelagem DER

3.2 Integração do Sistema

Para garantir a interoperabilidade entre as diferentes plataformas *WEB* e iOS, foi desenvolvido uma aplicação utilizando o estilo de arquitetura em REST.

O REST (*Representational State Transfer*/Transferência do Estado Representacional) foi criado a partir de uma tese de doutorado de Roy Fielding, um dos principais autores do protocolo HTTP (FIELDING, 2000).

REST proposto nesse projeto utiliza o cabeçalho HTTP como envelope de metadados e os métodos são definidos no próprio cabeçalho. Os métodos principais são GET e POST. As informações transferidas estão no formato JSON.

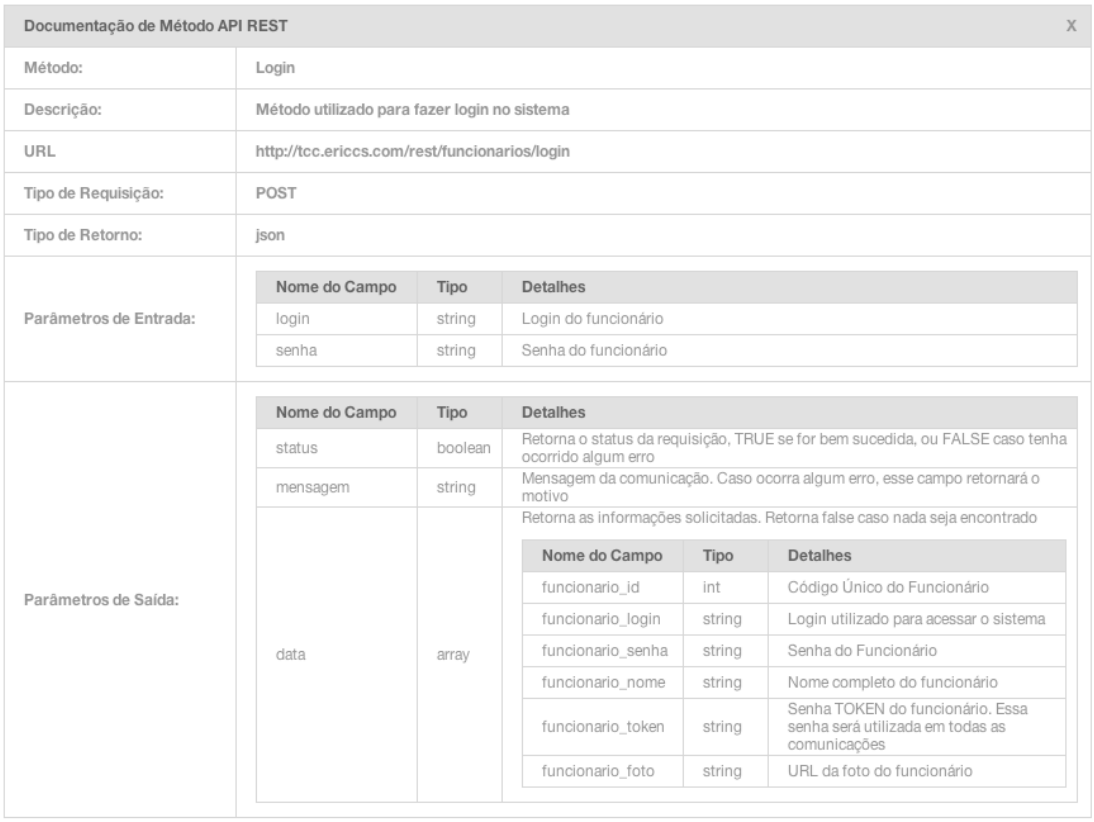
O servidor REST será consumido pelo sistema móvel. A Tabela 2 apresenta os métodos estão disponíveis

Tabela 2 – Métodos Rest

Nome do Método	Descrição
Login	Método utilizado para fazer login no sistema
BaixarOrdens	Método utilizado para fazer resgatar a lista de ordens de serviço que o funcionário terá que executar no dia
AtualizarCadastro	Método utilizado para atualizar o cadastro do funcionário
OrdemIniciar	Método utilizado para iniciar a ordem de serviço
OrdenCheckIn	Método utilizado iniciar o atendimento (quando o funcionário chegar na residência do cliente
OrdemCheckout	Método utilizado para finalizar uma orde de serviço
OrdemAlerta	Método utilizado emitir uma alerta (quando não for possível prosseguir com o atendimento)
ControlePonto	Método utilizado para atualizar o controle de ponto
CheckInLocalização	Método utilizado para atualizar a localização do funcionário

Para todos os métodos disponíveis pelo sistema de integração há uma documentação completa de como acessar os parâmentos de entrada, as informações de retorno, entre outras informações.

A Figura 15 ilustra a documentação dos métodos de integração.



Documentação de Método API REST X

Método:	Login
Descrição:	Método utilizado para fazer login no sistema
URL	http://tcc.ericcs.com/rest/funcionarios/login
Tipo de Requisição:	POST
Tipo de Retorno:	json

Parâmetros de Entrada:

Nome do Campo	Tipo	Detalhes
login	string	Login do funcionário
senha	string	Senha do funcionário

Parâmetros de Saída:

Nome do Campo	Tipo	Detalhes
status	boolean	Retorna o status da requisição, TRUE se for bem sucedida, ou FALSE caso tenha ocorrido algum erro
mensagem	string	Mensagem da comunicação. Caso ocorra algum erro, esse campo retornará o motivo
data	array	Retorna as informações solicitadas. Retorna false caso nada seja encontrado

Nome do Campo	Tipo	Detalhes
funcionario_id	int	Código Único do Funcionário
funcionario_login	string	Login utilizado para acessar o sistema
funcionario_senha	string	Senha do Funcionário
funcionario_nome	string	Nome completo do funcionário
funcionario_token	string	Senha TOKEN do funcionário. Essa senha será utilizada em todas as comunicações
funcionario_foto	string	URL da foto do funcionário

Figura 15 – Documentação dos métodos de integração

3.3Aplicação Móvel

Com o crescimento da computação móvel, cada vez mais empresas estão adaptando suas tecnologias para este novo cenário devido a grande flexibilidade que o mesmo proporciona.

As aplicações móveis podem ser *stand-alone*, isto é, funcionam sem a dependência de outras aplicações, ou podem depender de dados de outras aplicações que podem ser transmitidos por serviços *web*.

O sistema móvel foi desenvolvido para o *smartphone* iPhone e tem como objetivo auxiliar o funcionário na execução das ordens de serviço, fazer controle de ponto e transmitir informações sobre a localização do funcionário.

Para garantir a disponibilidade dos dados, não dependendo exclusivamente de conexão de Internet, foi criado um banco de dados SQLite no iPhone. O aplicativo consultará informações no servidor de integração utilizando conexão de Internet e

salvará os dados no dispositivo para que seja consultado quando a rede estiver indisponível.

A Figura 16 representa a Modelagem do Banco de dados SQLite para o iPhone.

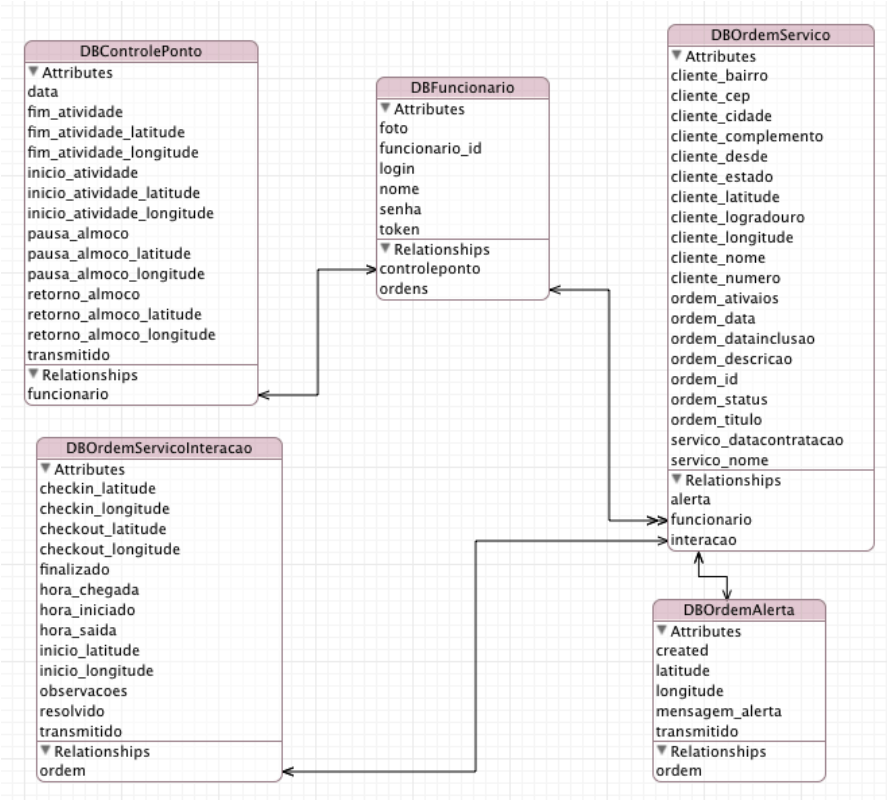


Figura 16 Modelagem Banco de Dados para iOS

Para facilitar a comunicação com o servidor REST, foi desenvolvida uma classe em Objective-C. Esta classe está preparada para receber os dados do servidor REST em JSON e converter os dados para ser utilizado no iOS.

A Figura 17 ilustra a implementação classe "RestClient.m" que poderá ser usada para consultar qualquer servidor REST.

```
#import "RestClient.h"

@implementation RestClient
@synthesize
apiURL,resposta,resposta_secondKey,erro,erro_msg,data,requestType;

//construtor
-(id)init{
    self = [super init];
    if(self){
        apiURL = [[NSString alloc] init];
        data =   [[NSString alloc] init];
        requestType = [[NSString alloc] initWithString:@"POST"];
        self.erro_msg = [[NSString alloc] initWithString:@""];
```

```
    }
    return self;
}

-(void)setURL:(NSString *)url{
    self.apiURL = url;
}

-(void)setData:(NSString *)key withValue:(NSString *)value{
    data = [data stringByAppendingString:key];
    data = [data stringByAppendingString:@"="];
    data = [data stringByAppendingString:value];
    data = [data stringByAppendingString:@"&"];
}

-(NSString *)getRespostaData:(NSString *)field{
    return [resposta valueForKey:field];
}

-(NSString *)getRespostaData:(NSString *)field withSecondKey:(NSString
*)skey{
    resposta_secondKey = [[NSDictionary alloc] initWithDictionary:[resposta
valueForKey:field]];
    return [resposta_secondKey valueForKey:skey];
}

-(void)sendData{
    if(!apiURL){
       self.erro_msg = @"É necessário setar a URL para comunicação";
    }else{
      //inicia a chamada do método
NSMutableURLRequest *request = [NSMutableURLRequest requestWithURL:[NSURL
URLWithString:apiURL]

cachePolicy:NSURLRequestUseProtocolCachePolicy

timeoutInterval:60.0];
       [request setHTTPMethod:requestType];
       [request setHTTPBody:[NSData dataWithBytes:[data UTF8String]
length:[data length]]];
       [request setValue:@"application/x-www-form-urlencoded"
forHTTPHeaderField:@"Content-Type"];

       NSError *tmperro;
       NSData *returnData = [NSURLConnection sendSynchronousRequest:request
returningResponse:nil error:&tmperro];
self.erro = tmperro;
       if(!erro){
           NSLog(@"Execução bem sucedidade... serializando...");
           resposta = [NSJSONSerialization JSONObjectWithData:returnData
options:kNilOptions error:&tmperro];
           self.erro = tmperro;
       }else {
           self.erro_msg = @"Ocorreu um erro na comunicação";
       }

    }
}
@end
```

Figura 17classe "RestClient.m"

As seguintes funcionalidades estão disponíveis na aplicação móvel.

1. **Login no Sistema:** Método utilizado para acessar o sistema. Deverá ser inserido o login e senha do funcionário. Para garantir que o login funcione independente de conexão de internet, o processo do login é validado da seguinte maneira (Figura 18):

a) O funcionário informa seus dados de acesso
b) O sistema inicia a comunicação com o serviço REST
c) Caso encontre o funcionário e os dados sejam válidos, o dispositivo móvel receberá uma instância atualizada do cadastro do funcionário, salvará no banco de dados e fornecerá acesso ao sistema.
d) Caso o serviço REST esteja indisponível, por ausência de conexão de dados ou indisponibilidade do serviço, o sistema fará uma segunda tentativa de login, porém, utilizará a base de dados do iPhone para consulta. Se os dados forem válidos, o sistema é liberado para uso.
e) Caso os dados informados não sejam válidos nas duas consultas (Serviço REST e base de dados do iPhone), o sistema exibirá um alerta.

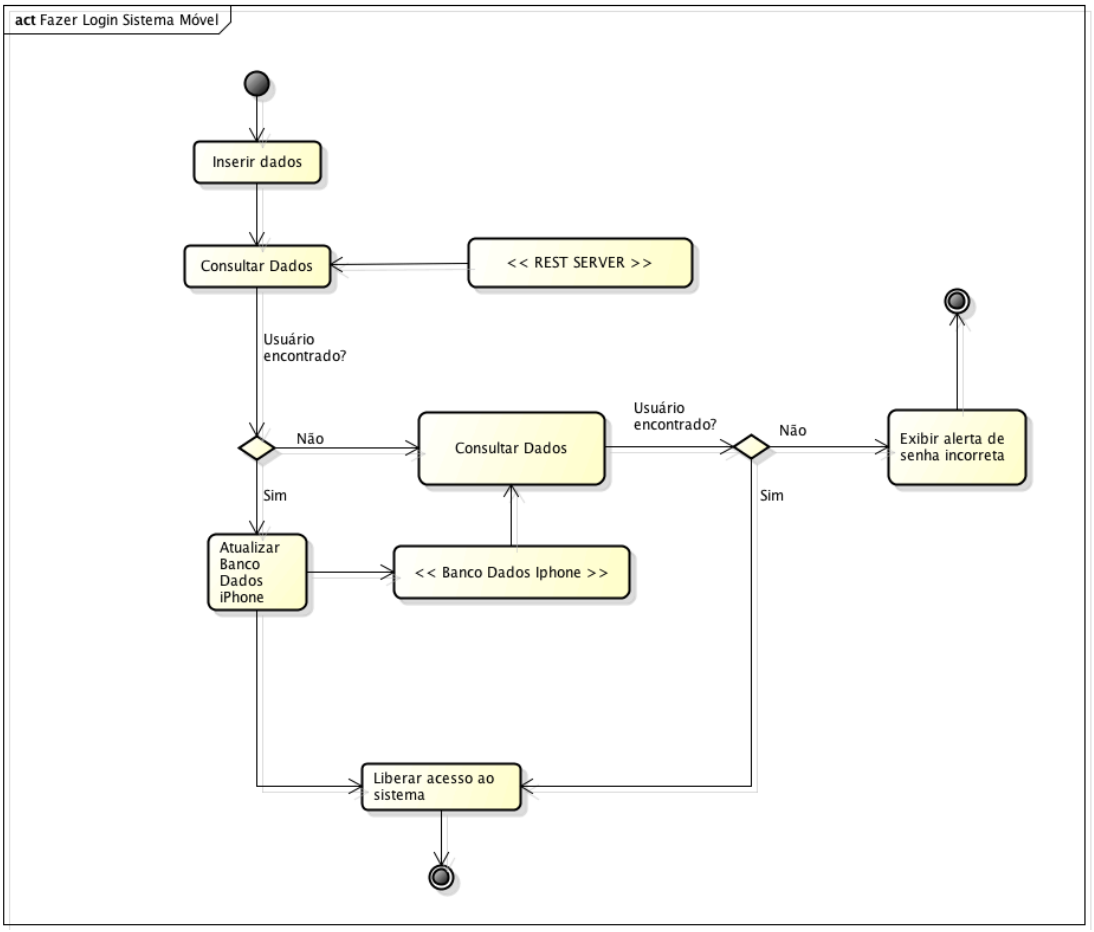


Figura 18 Diagrama do Método Login

2. **Baixar Ordens de Serviço:** Método utilizado para fazer o download das ordens de serviço que deverão ser executadas no dia. Esta funcionalidade inicia a comunicação com o serviço REST, recebe os dados e armazena no banco de dados do iPhone para garantir que as ordens de serviço possam ser consultadas independente de conexão de internet.

3. **Controle de Ponto:** Esta funcionalidade está divida em 4 ações e tem como objetivo inserir registros no controle de ponto do funcionário junto com sua localização.

a) Ação 1: Inicio de Atividade – Insere no controle de ponto a hora que o funcionário iniciou suas atividades.

b) Ação 2: Iniciar Almoço: Insere no controle de ponto a hora que o funcionário parou suas atividades para iniciar seu horário de almoço.

c) Ação 3: Retornar do almoço: Insere no controle de ponto a hora que o funcionário retornou do almoço para iniciar suas atividades.
d) Ação 4: Finalizar Trabalho: Ultimo registro no controle de ponto. Insere a hora que o funcionário encerrou sua jornada de trabalho diária.

4. **Editar Meus Dados:** Permite que o usuário atualize sua senha e seu login. As informações serão transmitidas para o serviço REST e serão armazenadas na base de dados do iPhone.

5. **Minhas Ordens de Serviço:** Esta funcionalidade está dividida em várias etapas e tem como objetivo executar a ordem de serviço.

a) O funcionário seleciona a ordem de serviço desejada.
b) O funcionário visualiza os detalhes da ordem de serviço podendo inicia-la ou emitir um alerta.
c) Ao iniciar a ordem de serviço, a aplicação enviará as coordenadas (check-in) para o servidor REST, armazenará os dados no banco de dados SQLite e exibirá informações do endereço do cliente.
d) Caso o funcionário chegue no endereço do cliente, o sistema fará um check-in para armazenar a hora de chegada e as coordenadas e exibirá detalhes completos da ordem de serviço. Caso o funcionário não chegue no local, poderá emitir um alerta.
e) Ao chegar no local, o funcionário poderá fazer o atendimento ou também poderá emitir um alerta caso não tenha material necessário ou ocorra alguma outra situação não planejada.
f) Ao finalizar o atendimento, o funcionário insere detalhes do mesmo e o sistema automaticamente faz o último check-in informando a hora da finalização de atendimento e as coordenadas de localização. Também é atualizada a instância da ordem de serviço.

A Figura 19 ilustra o fluxo da funcionalidade ordem de serviço.

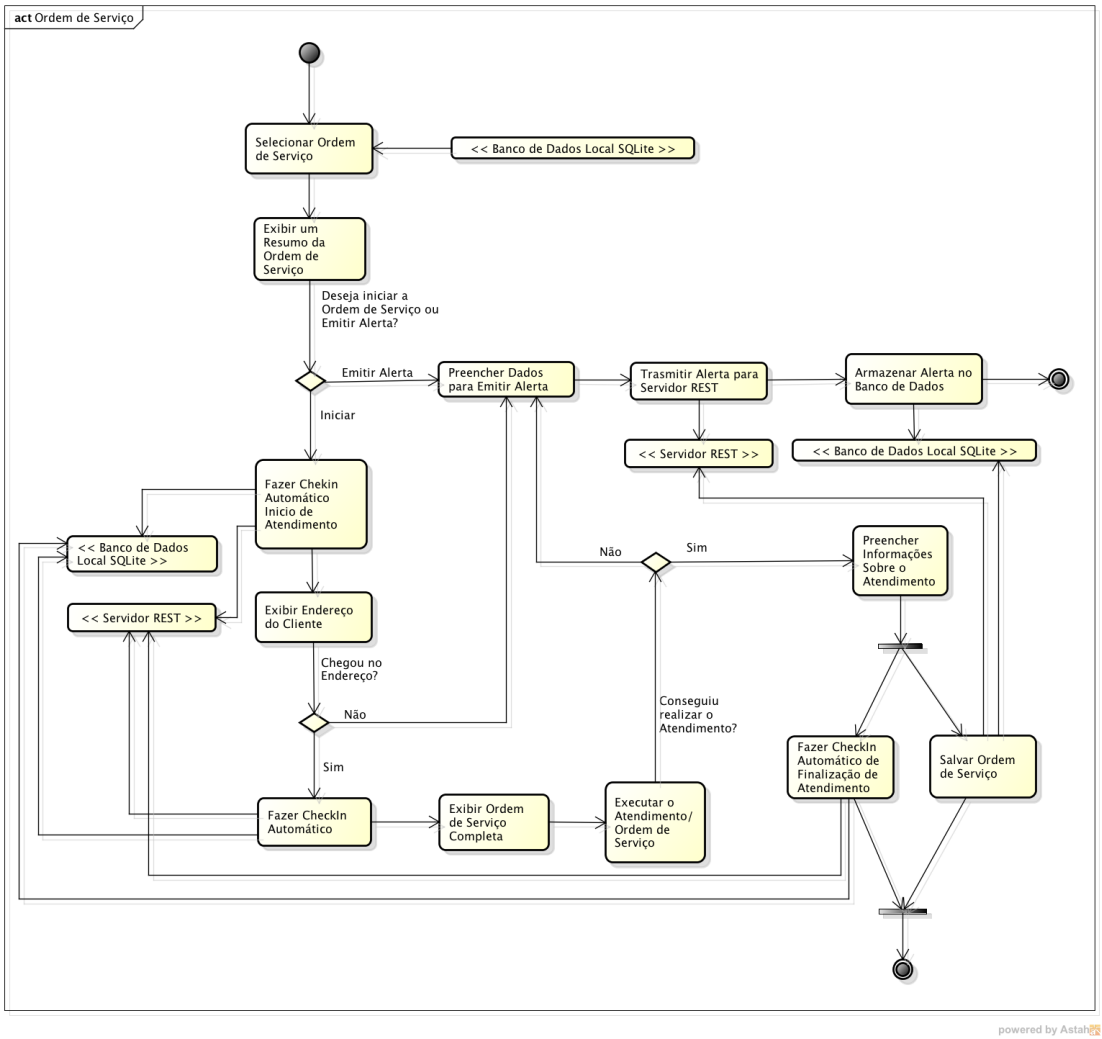


Figura 19 Diagrama do fluxo da Ordem de Serviço

6. **Funcionalidade Sincronizar Dados**: Esta funcionalidade é executada em segundo plano a cada 5 minutos e tem como objetivo transmitir informações que não foram transmitidas por erro de conexão de internet.

7. **Funcionalidade Check-In Localização**: Esta funcionalidade é executada em segundo plano e tem como objetivo fazer um ckeck-in da localização a cada 5 minutos. Desta forma o administrador da empresa poderá saber onde seu funcionário está sem que haja interação do funcionário com alguma funcionalidade da aplicação.

3.3.1 Geolocalização no iPhone

O *framework* Core Location fornece ao aplicativo a localização do dispositivo através de coordenadas. Este framework utiliza a tecnologia GPS, WPS *(Wireless Provisioning Services*) ou torre de celular para fazer a triangulação do sinal.

O sistema operacional iOS se responsabiliza em determinar qual tecnologia será utilizada de acordo com a disponibilidade.

Para obter uma localização no iOS, é necessário que sua classe implemente métodos da interface CLLocationManagerDelegate. Estes métodos utilizam um objeto do tipo CLLocationManager que traduzirá as mensagens e indicará as propriedades para o Core Location.

Para que a aplicação possa receber a localização é necessário que o usuário permita acesso a este recurso. É possivel definir uma mensagem para usuário utilizando a propriedade *purpose* em uma instância CLLocationManager.

Para definir a precisão na localização, utiliza-se a propriedade *desiredAccurancy* da classe CLLocationManager. Essa propriedade pode ser personalizada setando as seguintes constantes:

- KCLLocationAccuracyBestForNavigation: Utiliza GPS para identificar a localização fornecendo uma localização mais precisa, porém, utiliza alto consumo de energia do dispositivo.
- KCLLocationAccuracyBest: Possui alta precisão, mas também utiliza alto consumo de energia do dispositivo.
- KCLLocationAccuracyNearestTenMeters: Esta constante fornece uma precisão aproximadamente de 10 metros e utiliza alto consumo de energia do dispositivo.
- KCLLocationAccuracyHundredMeters: Esta constante tem a precisão de aproximadamente 100 metros. O consumo de energia é médio e utiliza WPS para identifcar a localização.
- KCLLocationAccuracyKilometer: Esta constante tem precisão de aproximadamante 1km. Por utilizar a torre de celular para identificar

a localização, possui um baixo consumo de energia.

- KCLLocationAccuracyThreeKilometers: Esta constante tem a precisão de aproximadamente 3km, utiliza a torre de celular e utiliza baixo consumo de energia.

Para aplicações que necessitam transmitir localização frequentemente, o CLLocationManager disponibiliza a propriedade *distanceFilter*. Esta propriedade é numérica e represtanta à distância em metros que o dispositivo precisa se mover para retransmitir a localização. Isso garante que a aplicação não fique transmitindo a localização a todo tempo gerando uso desnecessário da bateria.

Ao desenvolver a aplicação utilizando o Core Location pode-se optar qual tipo de serviço será utilizado sendo eles:

- Standard Location Services: Este serviço utiliza as funcionalidades de Wi-fi e GPS do celular e tem como vantagem a precisão na localização, mas tem suas desvantagens como lentidão para gerar as coordenadas, consome mais energia e não é executada em segundo plano, isto é, caso a aplicação seja minizada ou fechada o serviço para de funcionar.
- Significant Change Location Services: Este serviço tem como vantagem a velocidade na hora de gerar as coordenadas e também é possivel a execução em segundo plano.

A Figura 20 ilustra a classe GPS.

```
#import "Gps.h"

@implementation Gps

@synthesize locationManager;
@synthesize mapView;
@synthesize latitude;
@synthesize longitude;

-(id)init{
    self = [super init];
    if(self){
        locationManager.purpose = @"Permitir utilização do recurso GPS";
        locationManager = [[CLLocationManager alloc] init];
        locationManager.delegate = self;
        locationManager.distanceFilter = 100;
        locationManager.desiredAccuracy = kCLLocationAccuracyBest;
        [locationManager startUpdatingLocation];
    }
    return self;
}

-(void)getCoordenadas{
```

```
}

-(void)locationManager:(CLLocationManager *)manager didUpdateToLocation:(CLLocation
*)newLocation fromLocation:(CLLocation *)oldLocation{

    latitude = [[NSString alloc] initWithFormat:@"%g",
newLocation.coordinate.latitude];
    longitude = [[NSString alloc] initWithFormat:@"%g",
newLocation.coordinate.longitude];

}

@end
```

Figura 20 classe Gps.m

4. RESULTADOS

Os resultados desse trabalho foi uma aplicação móvel, uma aplicação *web* e um serviço REST para integrar as aplicações. Como resultado, será apresentado o uso da aplicação em exemplos práticos demonstrados pelas imagens a seguir.

Sistemas Web

Login:

Para acessar o sistema, o administrador deverá acessar a URL http://tcc.ericcs.com e inserir seus dados como é possível observar na Figura 21.

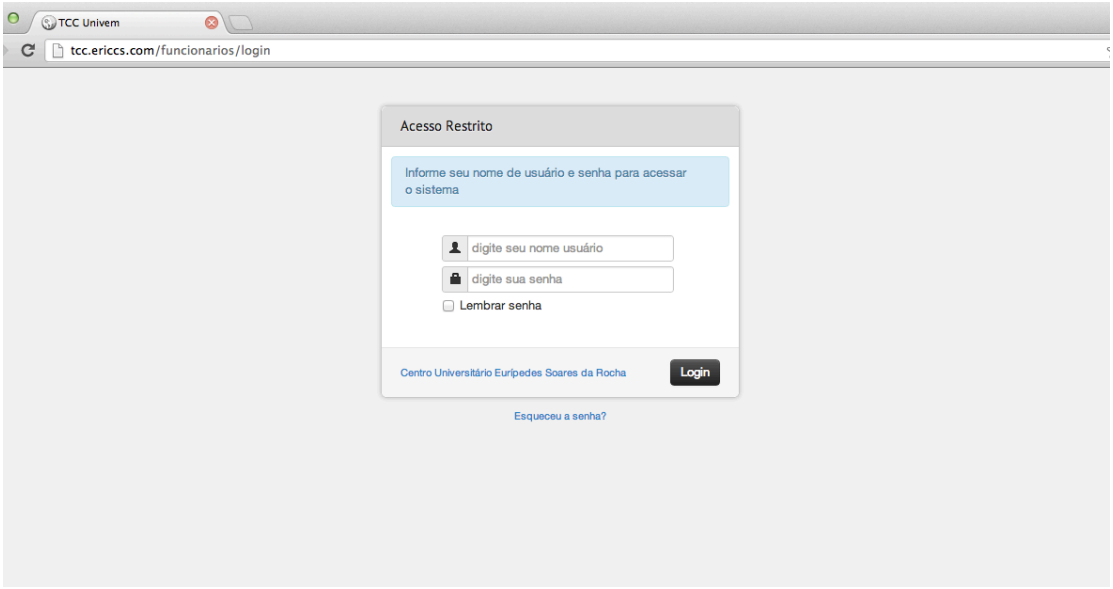


Figura 21 - Login

A Figura 22 ilustra o painel de bordo da aplicação *web*

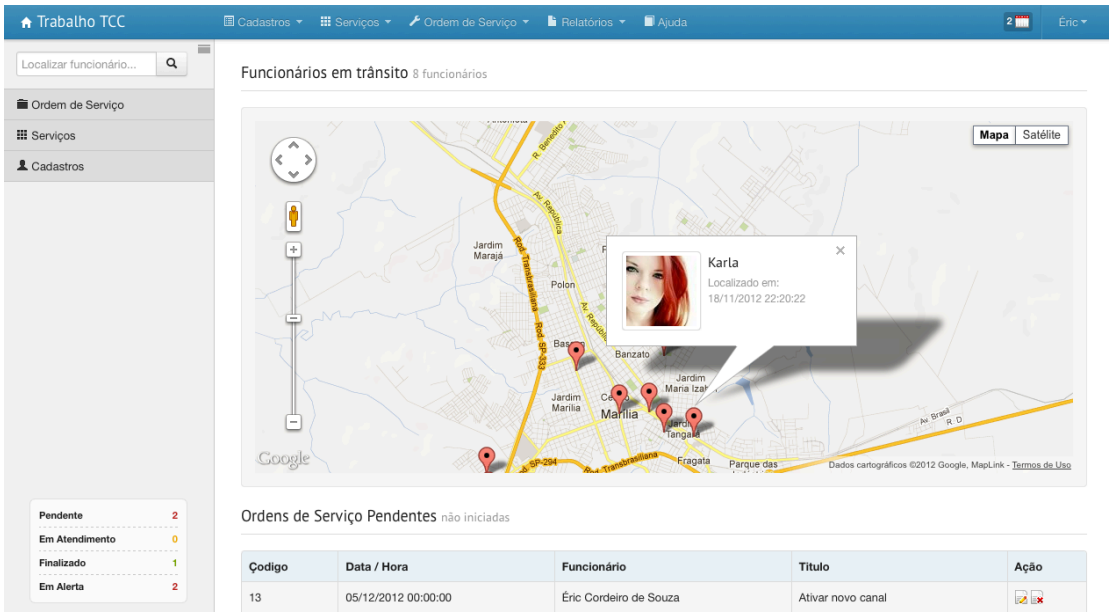


Figura 22 – Painel de bordo da aplicação *web*

Cliente

A aplicação *web* permite ao administrador listar todos os clientes (Figura 23), realizar buscas sobre um cliente específico, visualizar (Figura 25) e editar (Figura 24) suas informações ou cadastrar um novo cliente (Figura 26). Também é possível visualizar a localização da residência do cliente através de um mapa.

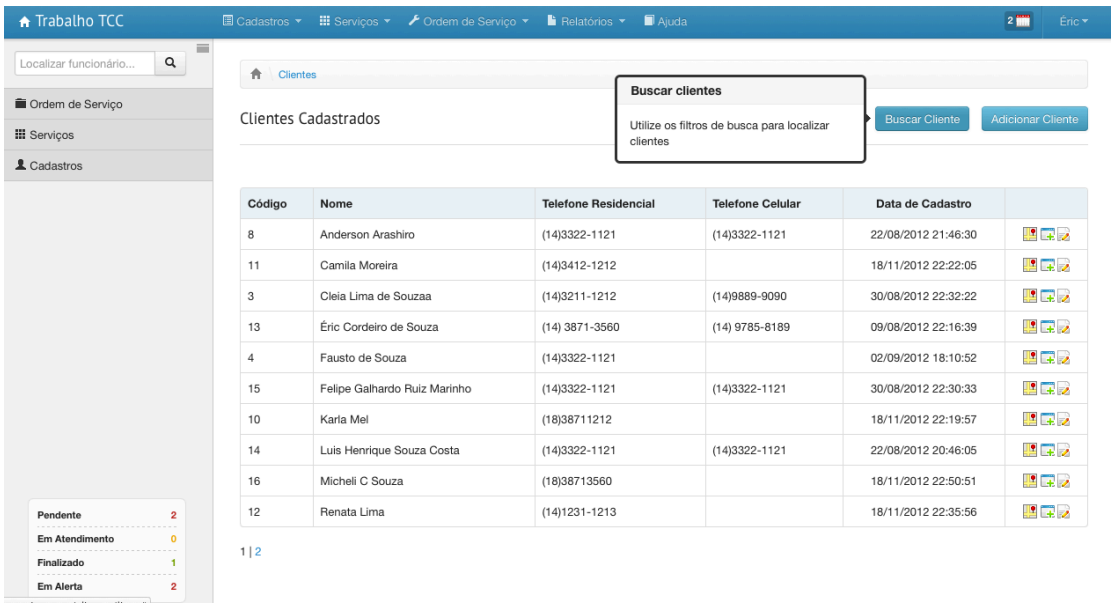


Figura 23 - Lista de clientes

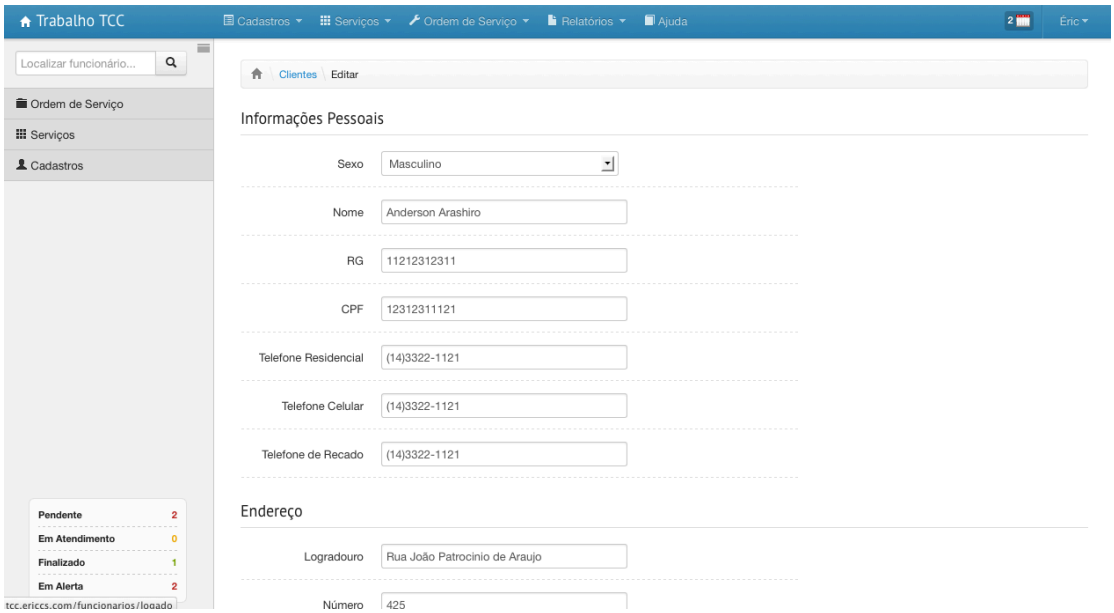


Figura 24 - Editar Cliente

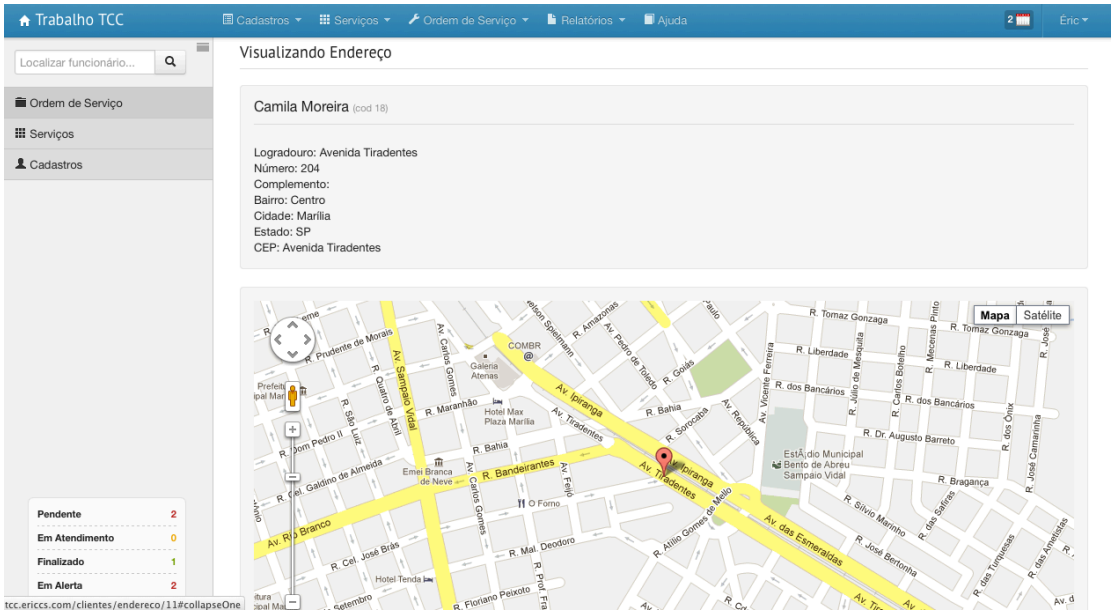


Figura 25 - Visualizar Cliente

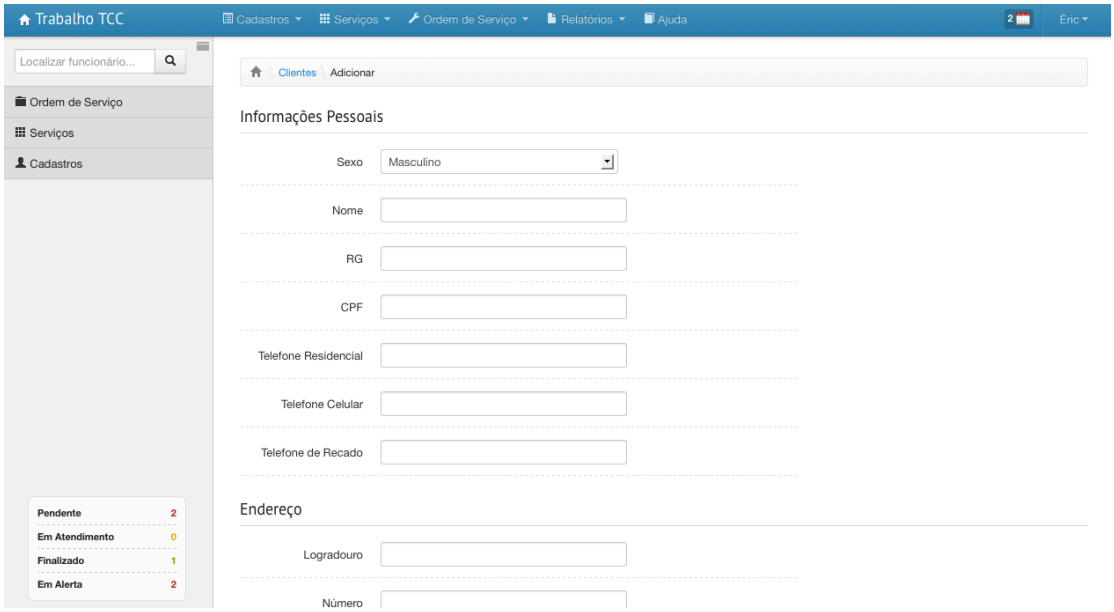


Figura 26 - Novo cliente

Funcionário

A aplicação *web* permite ao administrador listar todos os funcionários (Figura 27), realizar buscas sobre um funcionário específico (Figura 28), visualizar e editar suas informações (Figura 30), cadastrar um novo funcionário (Figura 31) e localizar seu funcionário visualizando através de um mapa ().

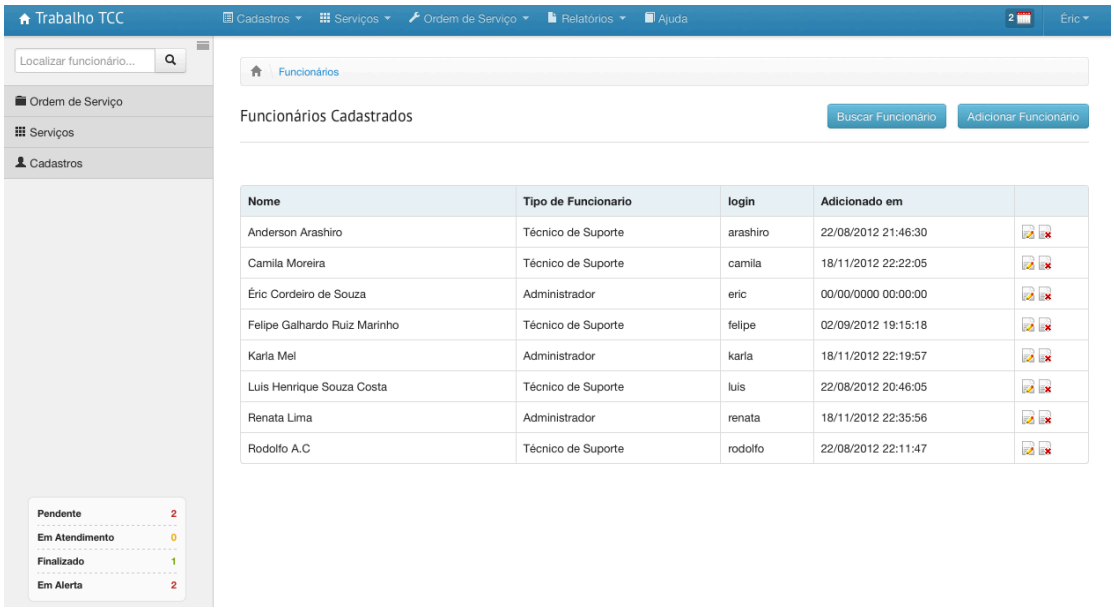


Figura 27 - Lista de funcionário

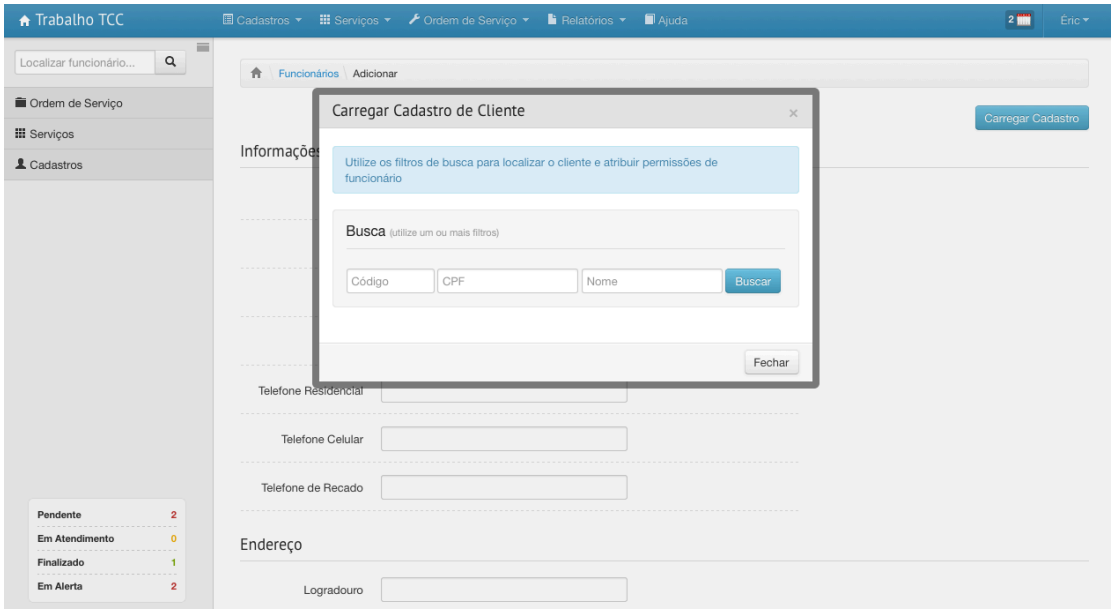


Figura 28 – Buscar funcionários

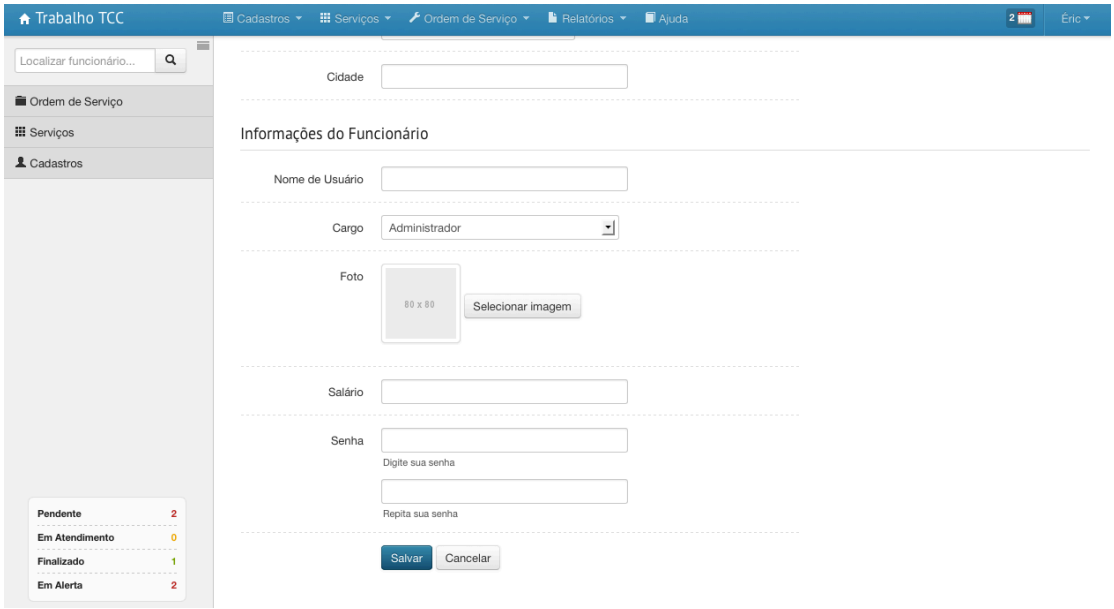


Figura 29 – Novo funcionário

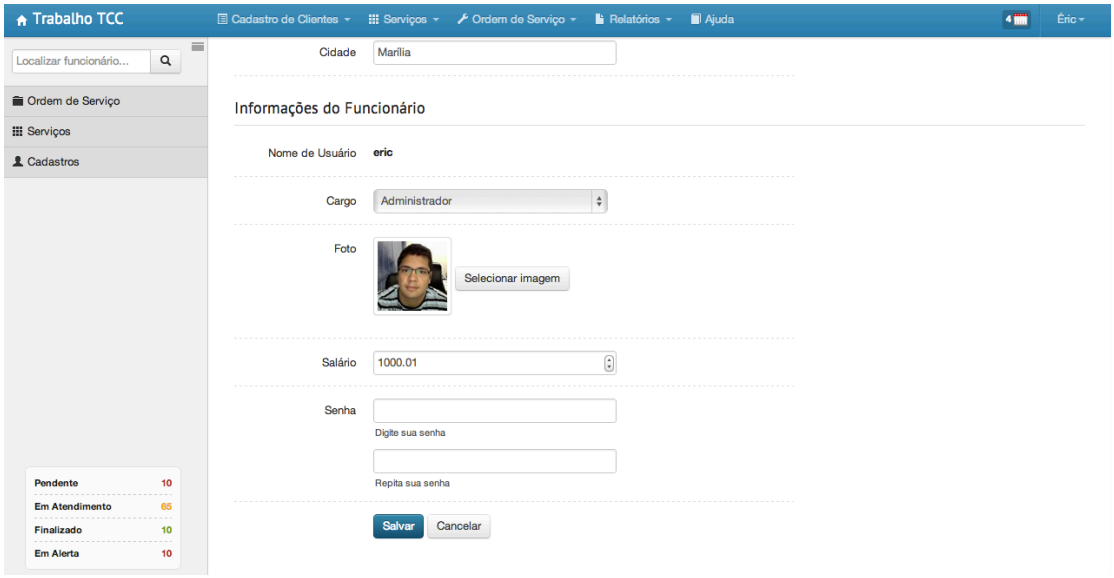


Figura 30 – Editar funcionário

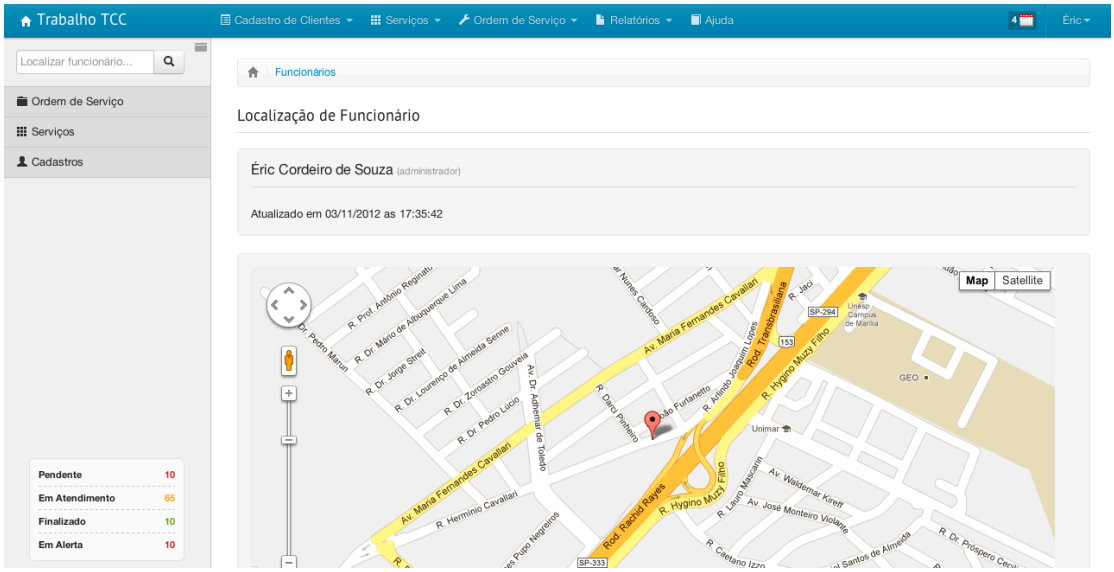


Figura 31 – Localização do funcionário

Serviços

A aplicação *web* permite ao administrador listar todos os serviços disponibilizados pela sua empresa e realizar buscas sobre um serviço específico (Figura 32), visualizar/editar suas informações (Figura 33) ou cadastrar um novo serviço (Figura 34). Também é possível adicionar serviços para os clientes (Figura 35).

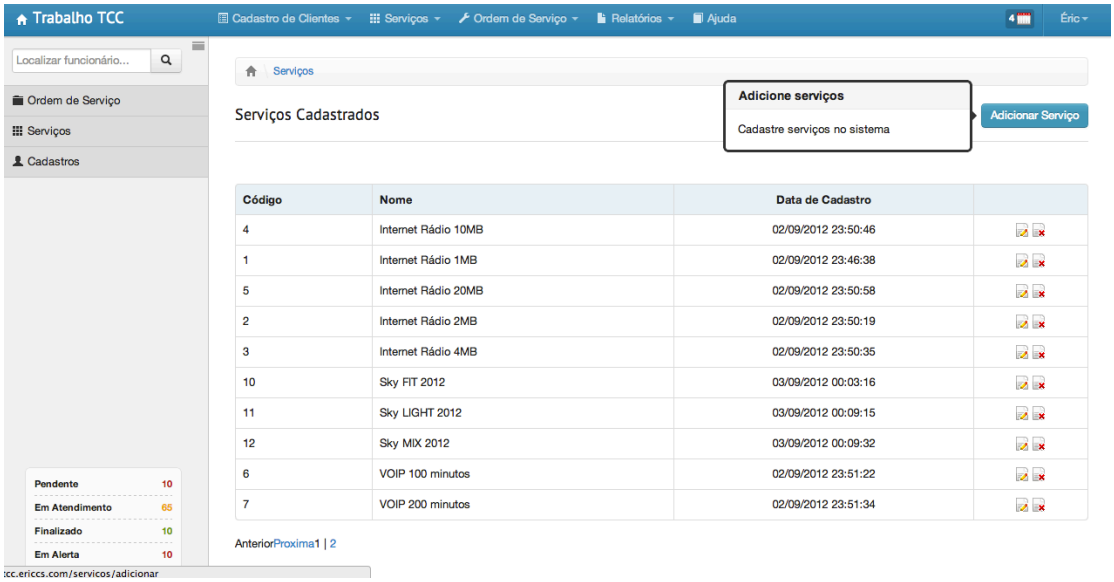


Figura 32 - Lista de Serviços

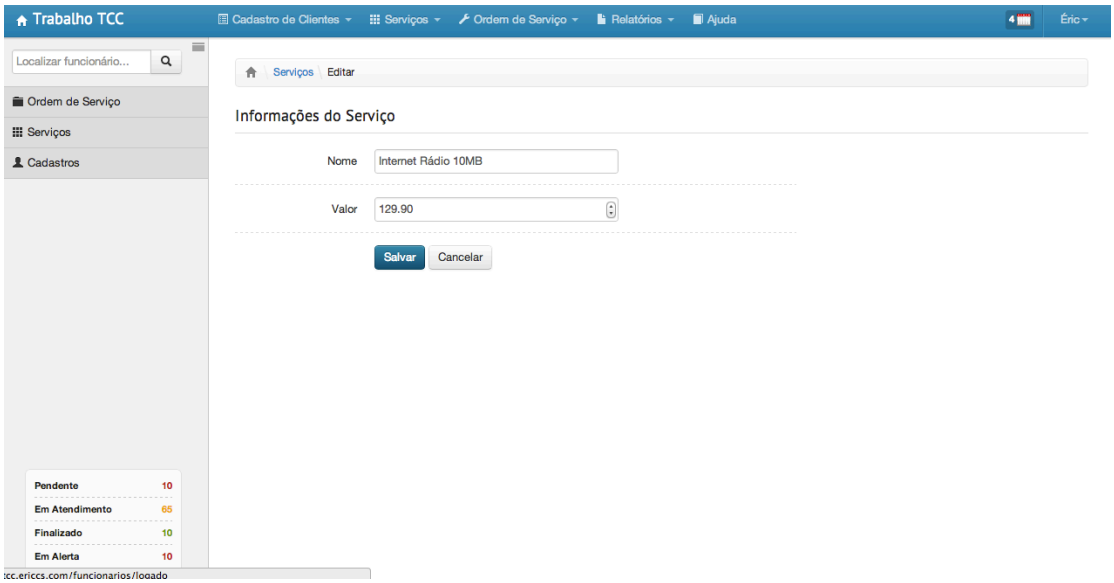


Figura 33 – Editar Serviço

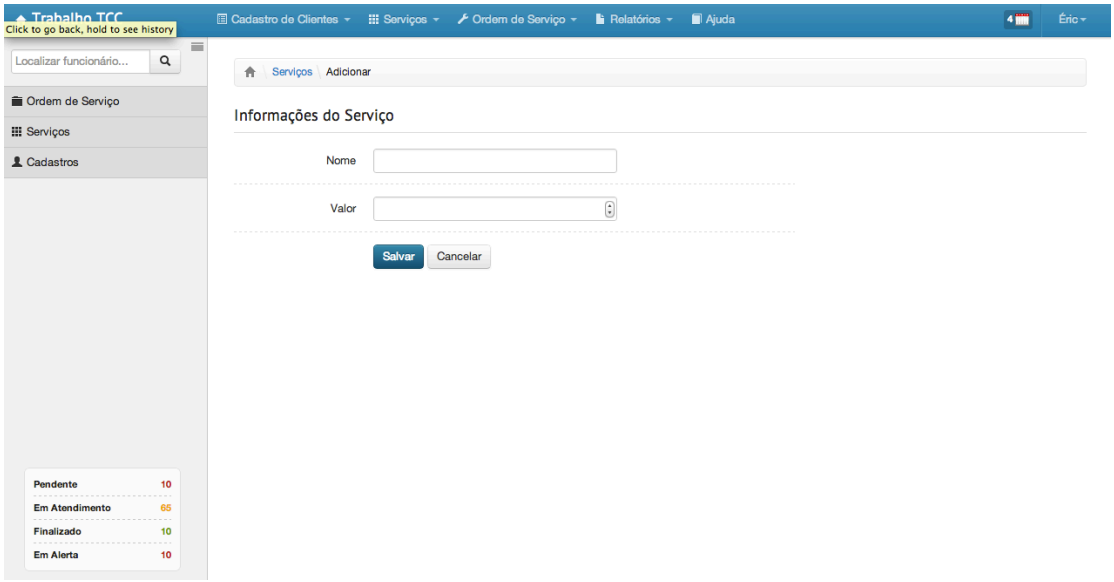


Figura 34 – Novo Serviço

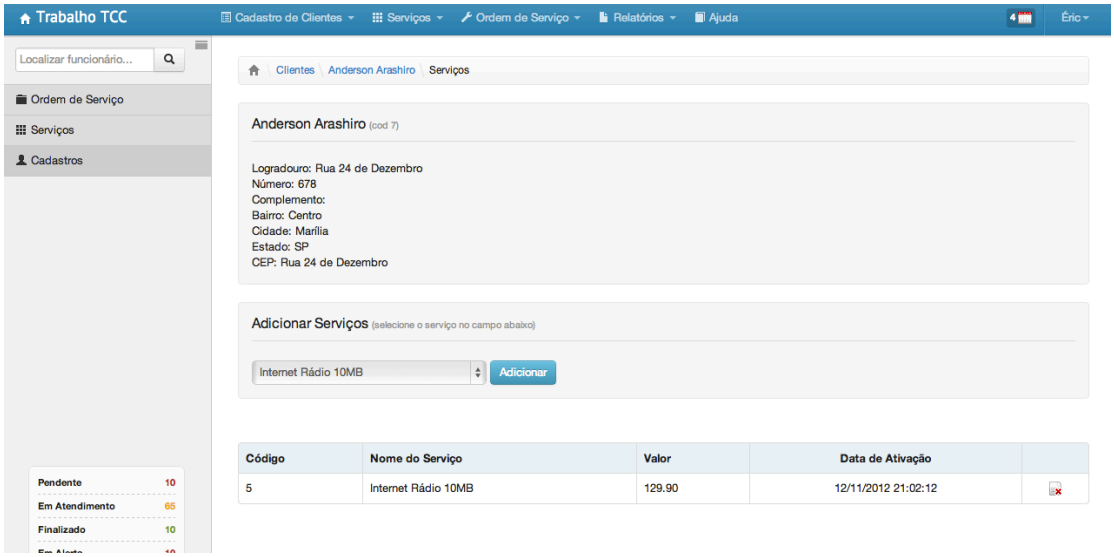


Figura 35 – Adicionar serviço ao cliente

Ordem de Serviço

A aplicação *web* permite ao administrador listar todas as ordens de serviços cadastradas e realizar buscas sobre uma ordem de serviço específica (Figura 36), visualizar e editar suas informações (Figura 37) ou cadastrar uma nova ordem de serviço (Figura 38).

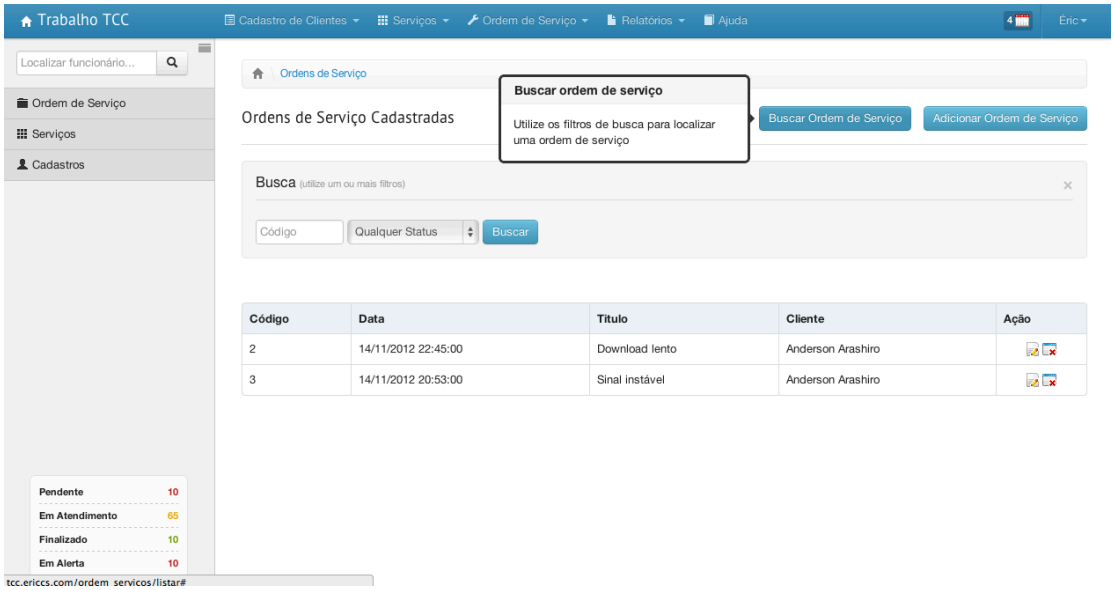


Figura 36 – Lista de ordem de serviços

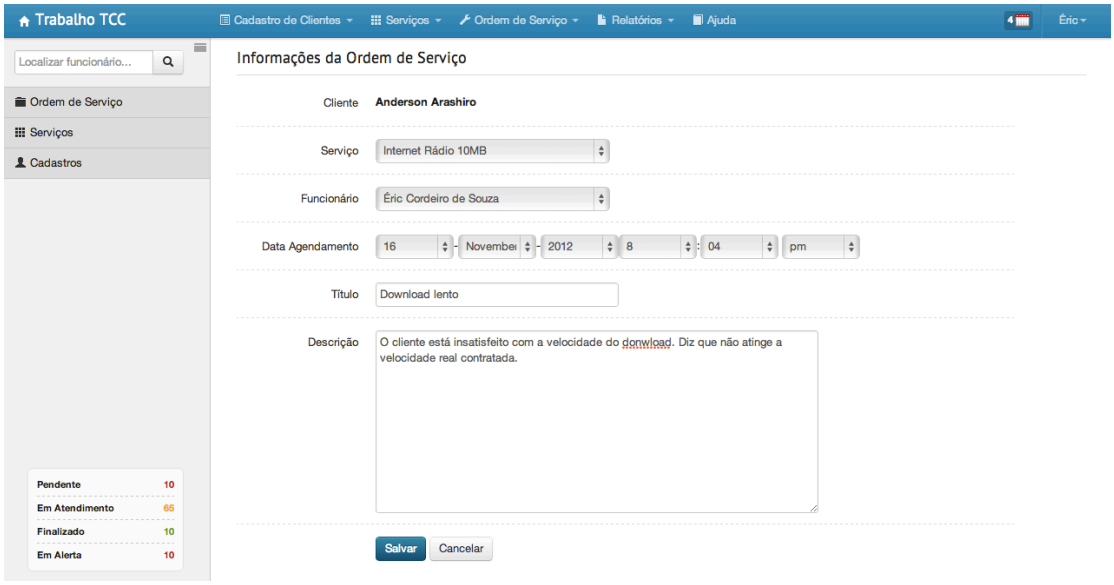


Figura 37 – Editar ordem de serviço

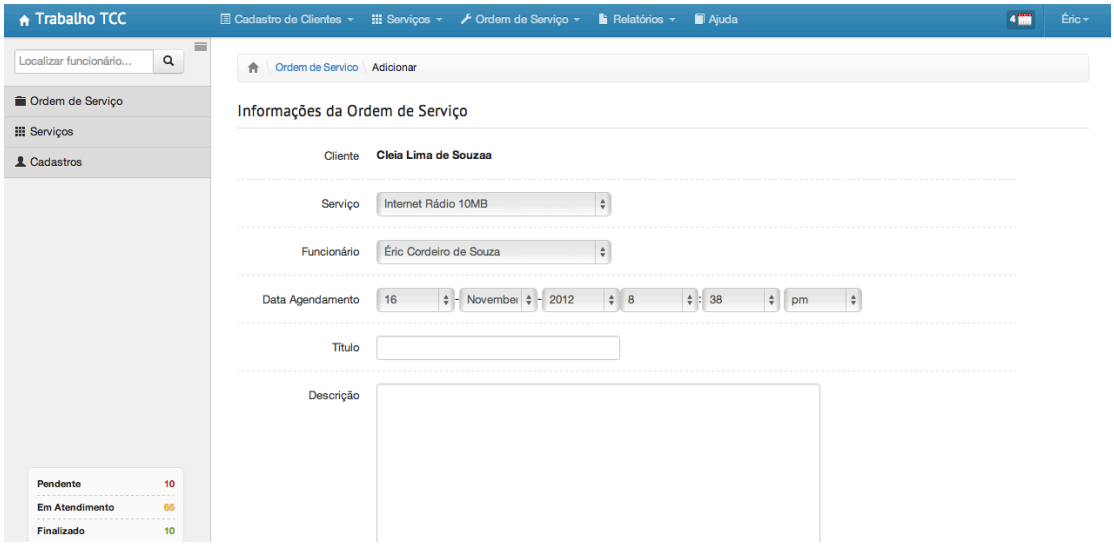


Figura 38 – Nova ordem de serviços

Relatórios

A aplicação *web* permite ao administrador visualizar relatórios de tempo em trânsito, tempo em atendimento, controle de ponto e ordens de serviço.

Relatório Tempo em trânsito (Figura 39; Figura 40): a aplicação *web* permite visualizar o tempo gasto em trânsito em um determinado dia, permite filtrar por dia e por funcionário desejado e permite exibir detalhes mostrando o tempo em trânsito gasto em cada ordem de serviço.

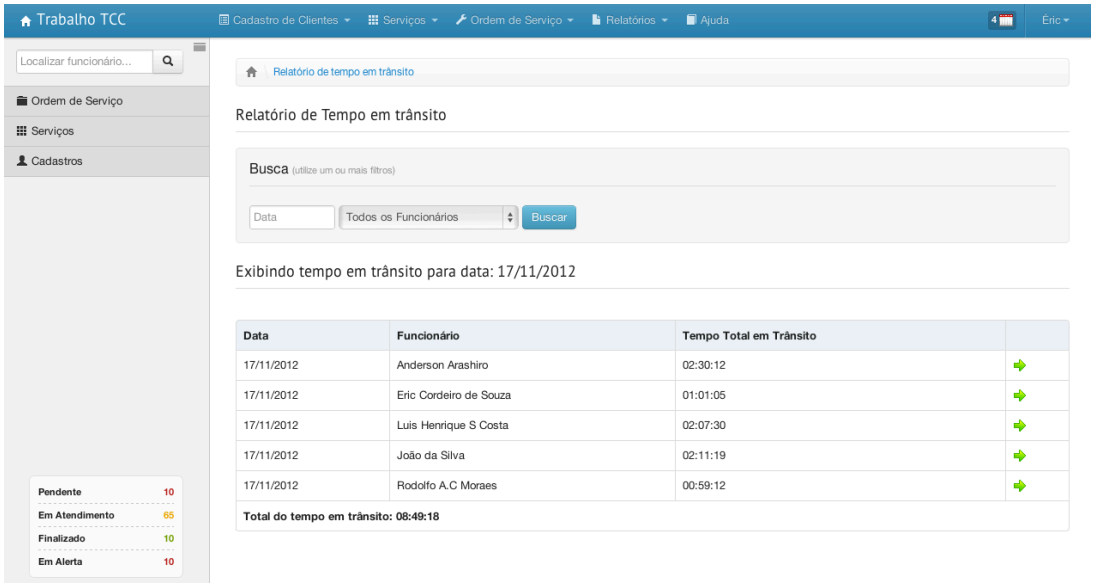


Figura 39 - Relatório de tempo em trânsito busca

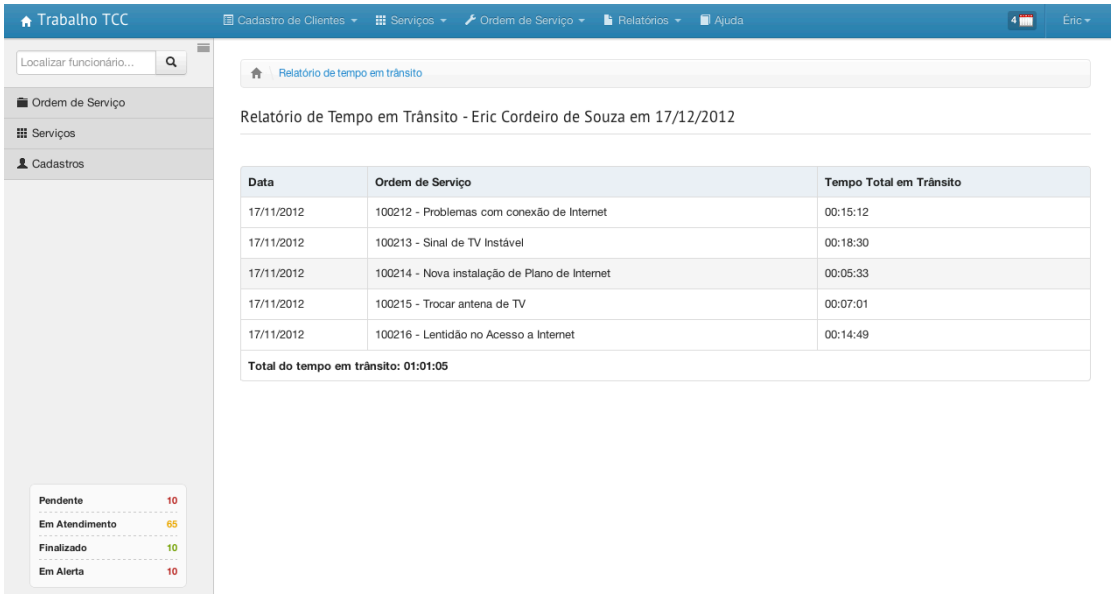


Figura 40 – Relatório de tempo em trânsito

Relatório Tempo em atendimento (Figura 41; Figura 42): a aplicação *web* permite visualizar o tempo de atendimento gasto num determinado dia, permite filtrar por dia e por funcionário desejado e permite exibir detalhes mostrando o tempo de atendimento gasto em cada ordem de serviço.

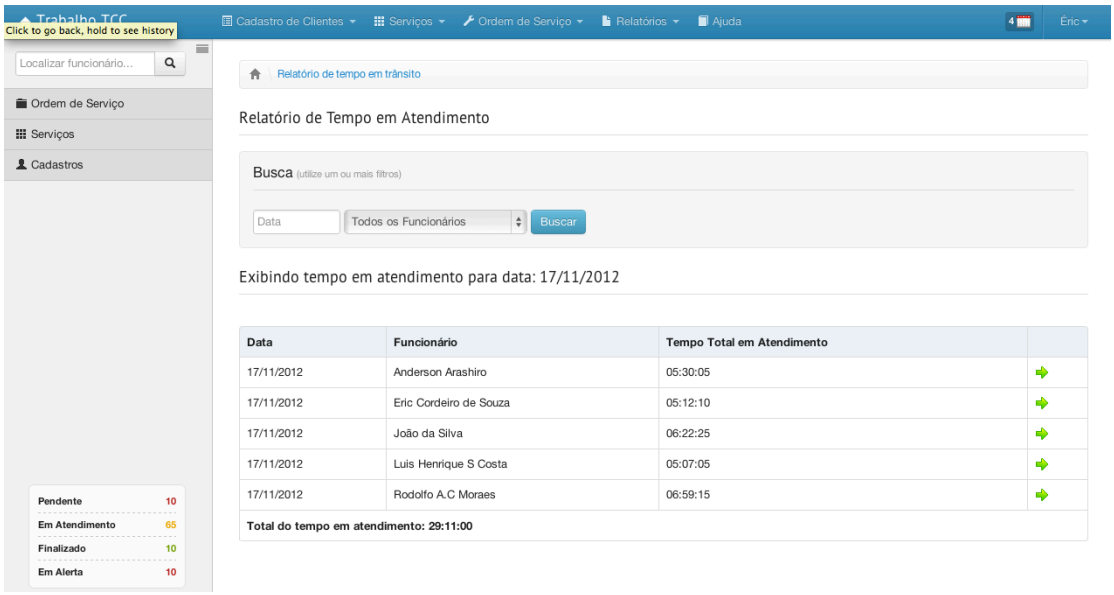


Figura 41 - Relatório de tempo em atendimento busca

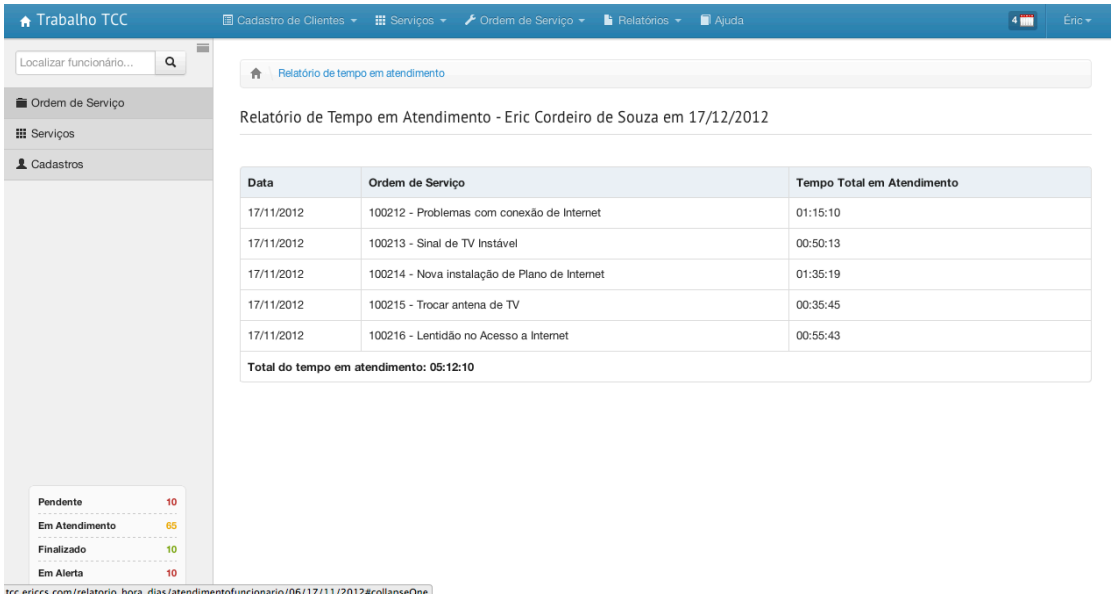


Figura 42 - Relatório de tempo em atendimento

Relatório Controle de Ponto (Figura 43; Figura 44): a aplicação *web* permite visualizar o controle de ponto de um determinado mês, permite filtrar por mês e por funcionário desejado e permite exibir detalhes mostrando todos os horários de ponto de um funcionário em um determinado mês.

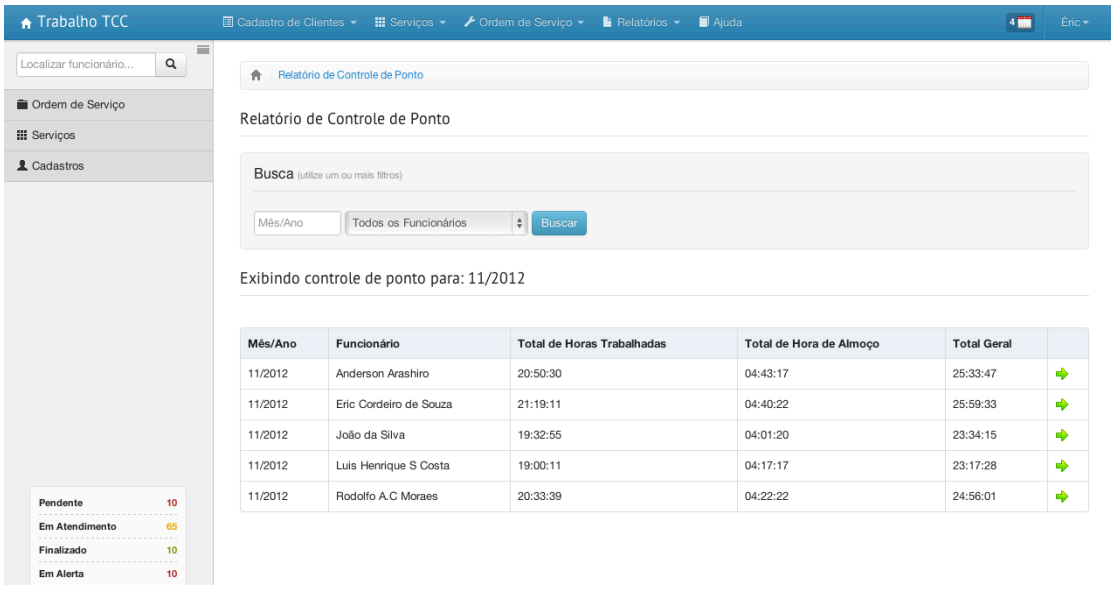


Figura 43 - Relatório de controle de ponto busca

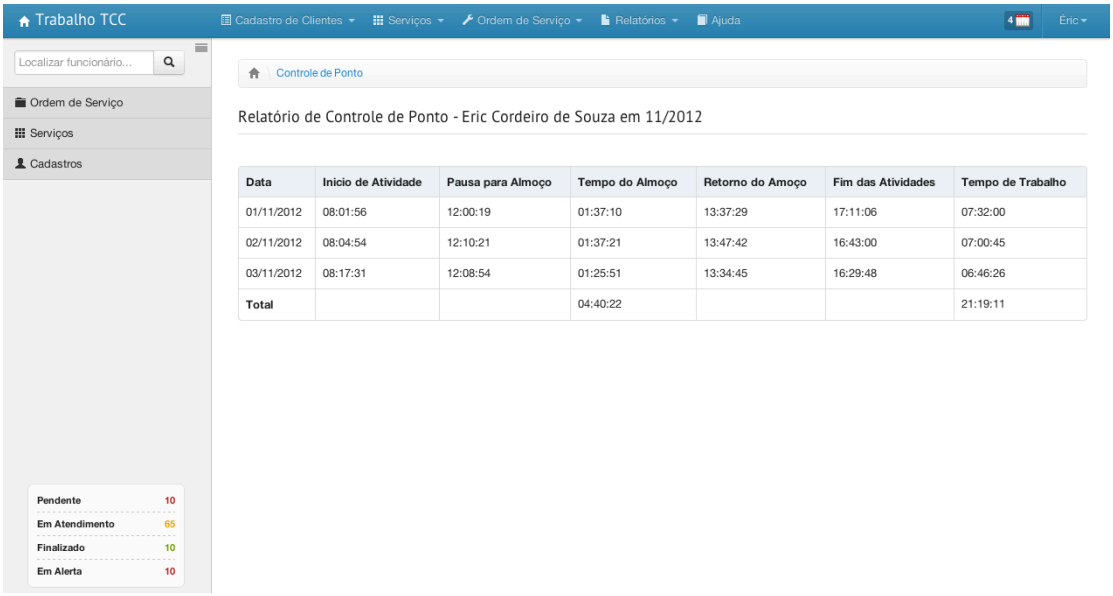


Figura 44 – Relatório de controle de ponto

Relatório Ordens de Serviço (Figura 45): Como é possível visualizar na figura abaixo, o sistema *web* permite visualizar o relatório anual de ordem de e permite filtrar por ano.

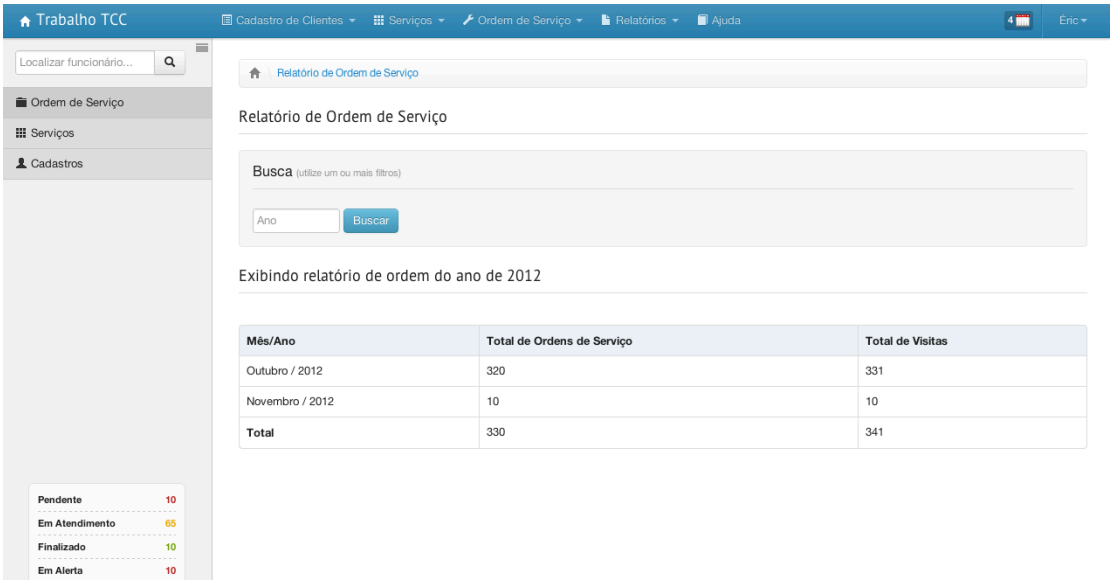


Figura 45 - Relatório de ordens de serviço

Sistema Móvel

Para visualizar o sistema móvel, é necessário executar a aplicação no iOS Simulator

Login

A Figura 46 ilustra o login na aplicação móvel que permite o acesso mediante a autenticação de usuário e senha.

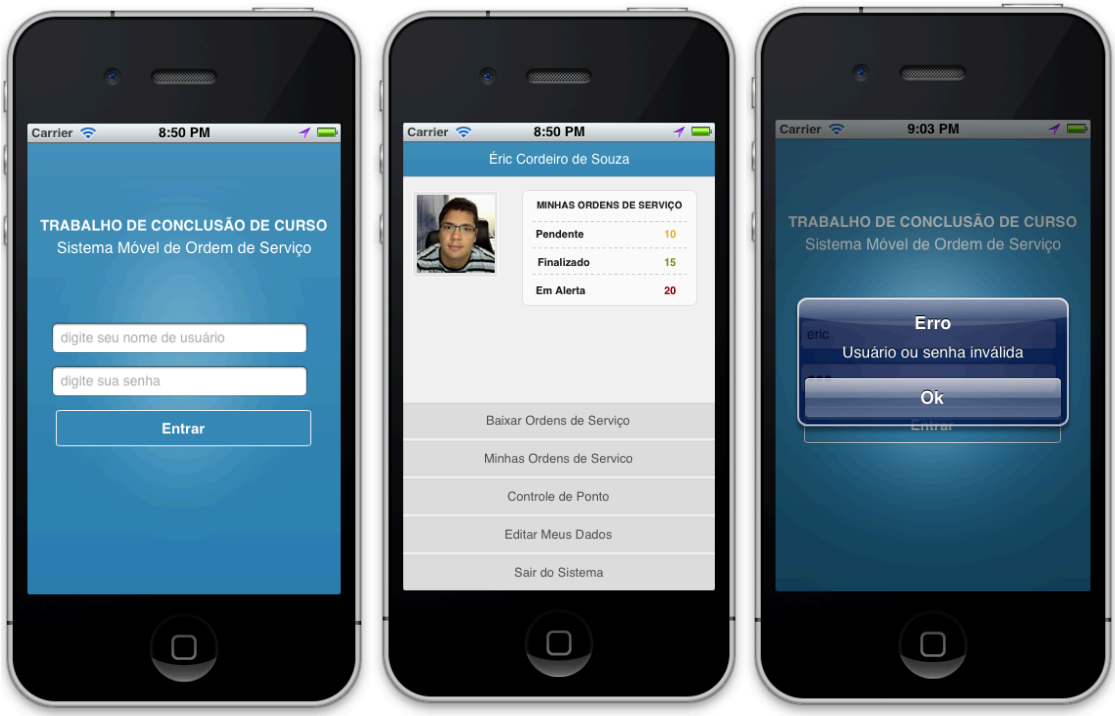


Figura 46- (A) Tela de Login; (B) Em caso de sucesso tela principal da aplicação; (C) Em caso da falha será exibida uma mensagem de erro.

Baixar Ordem de Serviço

Conforme é possível observar na Figura 47, a aplicação móvel permite que o funcionário faça o download de suas ordens de serviço que deverão ser executadas no dia.

Figura 47 - Baixar Ordem de Serviço

Ordem de Serviço

Conforme é possível observar na Figura 48, a aplicação móvel permite que o funcionário execute uma ordem de serviço ou emita um alerta caso não seja possível executá-la.


Carrier
8:51 PM
ORDENS DE SERVIÇO
Problema no sinal
Internet instável
Instalação Sky
Erro na conexão
Reparto na Antena
Remover Equipamento
Sem acesso a internet
Instalação Wireless
Configuração de Rede
Lentidão no acesso a internet
Troca de Roteador

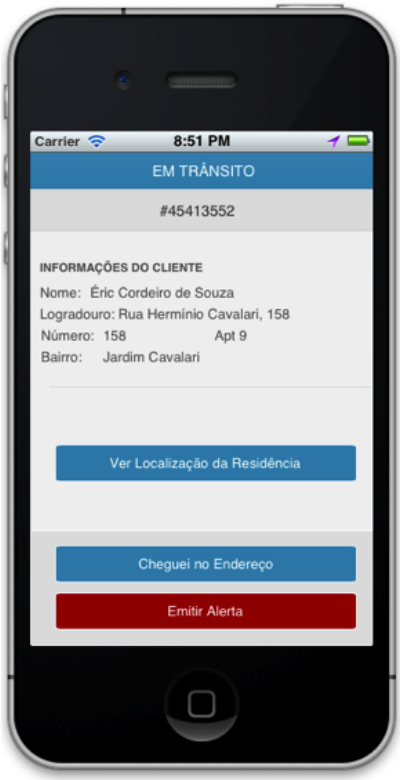
(A)


Carrier
8:51 PM
INICIAR ORDEM DE SERVIÇO
#45413552
INFORMAÇÕES DO CLIENTE
Nome: Éric Cordeiro de Souza
Cliente desde: 10/12/2012
INFORMAÇÕES DO SERVIÇO
Serviço: Sky Plus 2012
Data da contratação: 10/12/2012
INFORMAÇÕES DA ORDEM DE SERVIÇO
Data de Inclusão: 10/12/2012
Título: Problemas no Sinal da TV
Iniciar Atendimento / Ver Endereço
Emitir Alerta

(B)


Carrier
8:51 PM
EMITIR ALERTA
#45413552
INFORMAÇÕES DA ORDEM DE SERVIÇO
Cliente: Éric Cordeiro de Souza
Data de Inclusão: 10/12/2012
Título: Problemas no Sinal da TV
Emitir Alerta

(C)


Carrier
8:51 PM
EM TRÂNSITO
#45413552
INFORMAÇÕES DO CLIENTE
Nome: Éric Cordeiro de Souza
Logradouro: Rua Hermínio Cavalari, 158
Número: 158 Apt 9
Bairro: Jardim Cavalari
Ver Localização da Residência
Cheguei no Endereço
Emitir Alerta

(D)

(E) (F)

(G)

Figura 48 – (A) Lista de ordem de serviços; (B) Iniciando uma ordem de serviço; (C) Emitindo um alerta; (D) Verificando endereço; (E) Iniciando atendimento; (F) Finalizando atendimento; (G) Mensagem de sucesso.

Controle de Ponto

Conforme é possível observar na Figura 49, a aplicação móvel permite que o funcionário insira registros no controle de ponto.

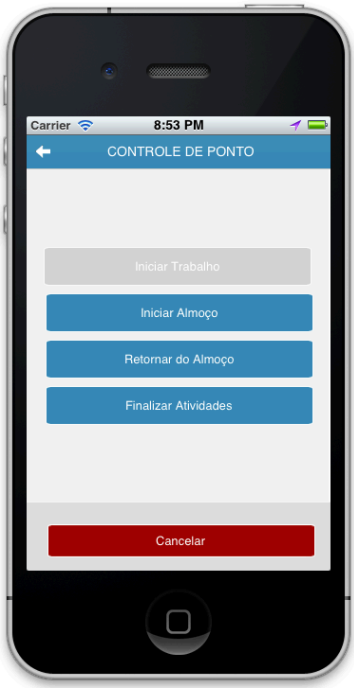


Figura 49 - Controle de Ponto

Editar dados de acesso

Conforme é possível observar na Figura 50, a aplicação móvel permite que o funcionário edite seu login e senha de acesso.

Figura 50 – Editar dados de acesso

4.1 Análise dos Resultados

Para avaliação das aplicações propostas nesse trabalho, foi definido uma metodologia de avaliação da qualidade da aplicação *web* e da aplicação móvel com base nos seguintes critérios:

1) Usabilidade: Conjunto de características que definem a capacidade que a aplicação possui de ser legível, atraente e navegável envolvendo aspectos de interface, estilo e simplicidade.

2) Funcionalidade: Conjunto de características que definem a capacidade da aplicação efetuar recuperação, inclusão e alteração de informação.

3) Eficiência: Conjunto de características que definem a capacidade da aplicação com relação a performance.

4) Confiabilidade: Conjunto de características que definem a capacidade da aplicação com relação a recuperação de erros, processamento correto dos links e validação de dados.

O conceito utilizado para avaliação consiste em ruim, regular e bom.

Aplicação Web: Para avaliar a aplicação *web* com base nos critérios citados, a aplicação foi disponibilizada para cinco usuários. Os resultados podem ser observados na Tabela 3.

Tabela 3 – Resultado da avaliação aplicação *web*

Critéiro	Usuário 1	Usuário 2	Usuário 3	Usuário 4	Usuário 5
Usabilidade	Bom	Regular	Regular	Bom	Bom
Funcionalidade	Bom	Bom	Bom	Bom	Bom
Eficiência	Bom	Bom	Bom	Bom	Regular
Confiabilidade	Bom	Bom	Bom	Bom	Bom

Aplicação Móvel: Para avaliar a aplicação Móvel, com base nos critérios, 5 pessoas utilizaram a aplicação. Os resultados podem ser observados na Tabela 4.

Tabela 4 - Resultado da avaliação aplicação móvel

Critéiro	Usuário 1	Usuário 2	Usuário 3	Usuário 4	Usuário 5
Usabilidade	Bom	Bom	Bom	Bom	Bom
Funcionalidade	Bom	Bom	Bom	Bom	Bom
Eficiência	Bom	Regular	Bom	Bom	Regular
Confiabilidade	Bom	Bom	Bom	Bom	Bom

5. CONCLUSÃO

O objetivo desse trabalho foi desenvolver uma aplicação de ordem de serviço para que o administrador tenha controle sobre seus funcionários que trabalham externamente, sabendo sua localização, a carga horária de trabalho e as tarefas que eles executam.

O desenvolvimento foi composto por uma aplicação móvel para o *smartphone* iPhone, uma aplicação *web* para o administrador da empresa e um serviço de integração (baseado em REST) entre as aplicações.

Com esse estudo, foi possível perceber o grande crescimento das tecnologias móveis que sem dúvida são a nova tendência tecnológica para aplicações que precisam de mobilidade.

Em relação à implementação na plataforma iOS, foi identificado no aprendizado da linguagem Objective-C que a linguagem segue uma sintaxe muito diferenciada das principais linguagens existentes e que para adquirir o aprendizado é necessário recorrer a sites americanos e assistir vídeo aulas em inglês pois não há muito material em português para o estudo.

Também foi possível perceber como o mercado de aplicações móveis está aquecido, e que há grandes casos de sucesso com aplicações não muito complexas, mais utilizadas para entretenimento.

Em relação à aplicação *web*, foi possível perceber o grande número de frameworks disponíveis para implementar na linguagem PHP e que esta linguagem é muito utilizada para desenvolvimento *web*. Há muito material em português para ajudar no aprendizado além de contar com documentações do framework CakePHP e do PHP em português.

Em relação ao serviço de integração (REST), foi possível perceber que as empresas estão aderindo a esse tipo de tecnologia devido a velocidade na comunicação e simplicidade.

Em relação ao dispositivo móvel, foram identificados problemas de conectividade de rede, porém todos os problemas foram solucionados através de lógicas implementadas no próprio dispositivo persistindo a informação e retransmitindo posteriormente.

5.1 Trabalhos Futuros

As aplicações desenvolvidas possibilitam que o administrador tenha um controle sobre os funcionários que executam tarefas fora da empresa, mas este trabalho poderá evoluir recebendo novas funcionalidades deixando o software mais inteligente.

Das funcionalidades futuras, foi possível identificar:

- Implementação de Nível de Acesso: O sistema foi modelado para ter vários tipos de funcionários (Administradores, Operadores, Atendentes). O próximo passo é implementar uma ACL para controlar o acesso por tipo de usuário.
- Acesso a clientes: O sistema poderá liberar acesso ao cliente. Dessa forma, o cliente interage com um atendente via chat. O chat automaticamente iniciará uma ordem de serviço e ao finalizar o atendimento via chat, o atendente poderá agendar essa ordem de serviço para que seja resolvida através de uma visita ao cliente ou poderá encerrar a mesma.
- Feed-back de atendimento: Ao finalizar uma ordem de serviço, o sistema móvel poderia notificar um departamento de controle de qualidade. O departamento poderia ligar para o cliente e fazer uma análise da satisfação do cliente.
- Alertas Inteligentes: Quando o usuário emite um alerta, o sistema envia a notificação para o administrador reagendar o atendimento. Futuramente seria possível deixar a aplicação mais inteligente, ao incluir um alerta, o sistema poderia localizar outro funcionário mais proximo da residência do cliente e atribuir a ordem para ele.
- Traçar rota: O sistema móvel exibe a localização da residência do cliente através de um mapa. Futuramente é possível fazer com que a aplicação exiba a rota de trânsito necessária para chegar na residência do cliente através do mapa ou de texto. Para que isso seja possível, é necessário implementar métodos da API do Google Maps.

- Mensagem instantânea: É possível implementar um chat no sistema *web* que seja integrado com um chat no sistema móvel. Dessa forma, a empresa reduzirá custos com telefonia.

REFERÊNCIA BIBLIOGRAFICA

ANDROID, Disponível em: http://developer.android.com/tools/index.html acessado em Novembro 2012

ANATEL, Agencia nacional de telecomunicações: Número de habilitações de aparelhos celulares, Disponível em http://www.anatel.gov.br/Portal/exibirPortalNoticias.do?acao=carregaNoticia&codigo=24913

ANATEL, Número de acesso a internet móvel - Disponível em http://sistemas.anatel.gov.br/SMP/Administracao/Consulta/AcessosPrePosUF/graficoRegiao.asp?mes=7&ano=2012

APPLE iOS - http://www.apple.com/br/ios/

Apple Introduces X-Code, 2012 – Disponível em https://www.apple.com/pr/library/2003/06/23Apple-Introduces-Xcode-the-Fastest-Way-to-Create-Mac-OS-X-Applications.html

Apple Core Audio, http://developer.apple.com/library/ios/#documentation/MusicAudio/Conceptual/CoreAudioOverview/WhatisCoreAudio/WhatisCoreAudio.html#//apple_ref/doc/uid/TP40003577-CH3-SW1

Apple UI KIT, http://developer.apple.com/library/ios/#documentation/uikit/reference/UIKit_Framework/Introduction/Introduction.html#//apple_ref/doc/uid/TP40006955-CH1-SW2

CERT BR – Segurança em dispositivos móveis: Disponível em http://cartilha.cert.br/dispositivos-moveis/

CHARLAND ANDRE, LEROUX BRIAN. 2011. Mobile application development: web vs. native.Commun. ACM 54, 5 (May 2011), 49-53. DOI=10.1145/1941487.1941504 http://doi.acm.org/10.1145/1941487.1941504

COCOA - https://developer.apple.com/technologies/mac/cocoa.html

COULOURIS, George F. Sistemas Distribuídos: conceitos e projeto. 4ª ed. Porto Alegre: Bookman, 2007

Core Animation 2012, https://developer.apple.com/library/mac/#documentation/cocoa/conceptual/coreanimation_guide/Articles/WhatisCoreAnimation.html#//apple_ref/doc/uid/TP40004689-SW1

FAIRBAIRN, Christopher K. Objective-C fundamental / Christopher K. Fairbairn, Johannes Fahrenkrug, Collin Ruffenach ; [tradução Rafael Zanolli]. -- São Paulo : Novatec Editora ; London, NY : Manning Publications, 2012.

FIELDING, Roy Thomas Fielding, Architectural Styles and the Design of Network-based Software Architectures, Tese de Doutorado, University of california, Irvine, 2000, Disponível em:

http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

FLING, B. Mobile Design and Development. 1st ed. California: O'Reilly, 2009. ISBN 0596155441.

FOURSQUARE. Disponível em <http://pt.foursquare.com>. Acessado em: 15 novembro. 2012.

GOOGLEAPPENGINE. Disponível em <http://code.google.com/intl/pt-BR/appengine/>. Acessado em: 15 novembro. 2012.

GOOGLECODE. Disponível em <http://code.google.com>. Acessado em: 15 novembro. 2012.

GOOGLEMAPS. Disponível em <http://www.google.com.br/mobile/maps/>. Acessado em: 15 novembro. 2012.

GOOGLELATITUDE. Disponível em <http://www.google.com.br/intl/ptBR_ALL/mobile/latitude/>. Acessado em: 15 novembro. 2012.

IBM Worklight, Disponível em: http://www-01.ibm.com/software/mobile-solutions/worklight/features/ acessado em Novembro 2012.

IDC, Disponível em: http://www.idc.com/getdoc.jsp?containerId=prUS23297412 acessado em Fevereiro 2012.

JAVAME. Disponível em <http://www.oracle.com/technetwork/java/javame/index.html> Acessado em: 15 novembro. 2012.

LECHETA, Ricardo R. Google Android: Aprenda a Criar Aplicações Para Dispositivos Móveis com Android SDK. Novatec, 2009.

MARZULLO Fabio, iPhone na Prática. Novatec, 2012.

MILANI, André Programando para iPhone e iPad : aprenda a construir aplicativos para o iOS / André Milani. -- São Paulo : Novatec Editora, 2012.

MONICO, J.F.G. 2000. Posicionamento pelo NAVSTAR-GPS: descrição, fundamentos e aplicações. São Paulo: Editora UNESP, p287.

MTPS Disponível em: Ministério do Trabalho e da Previdência Social - Portaria nº 3.626, de 13/nov/91 http://www010.dataprev.gov.br/sislex/paginas/66/MTPS/1991/3626.htm acessado em Fevereiro 2012.

PHP, Php.net Disponível em: http://www.php.net/downloads.php#v5 acessado em Fevereiro 2012.

PHONEGAP, Disponível em: http://phonegap.com/ acessado em Novembro 2012

SQLite, Disponível em: http://www.sqlite.org/ acessado em Fevereiro 2012.

TITANIUM, http://www.appcelerator.com/platform/titanium-sdk/ acessado em Novembro 2012

ANEXO 1 - QUESTIONÁRIO

Nome: _____

Telefone: ( ) _____ - _____

Idade: _____

Em relação a utilização do sistema web, assinale a alternativa que melhor representa a sua opinião quanto ao uso do sistema

Em relação a Usabilidade			
	ruim	regular	bom
Compreensibilidade do sistema	( )	( )	( )
Mecanismos de ajuda e retorno	( )	( )	( )
Aspectos de Interface	( )	( )	( )
Legibilidade (Clareza no conteúdo apresentado)	( )	( )	( )
Navegabilidade	( )	( )	( )

Em relação a Funcionalidade			
	ruim	regular	bom
Mecanismo de Busca	( )	( )	( )
Autenticação do sistema	( )	( )	( )
Segurança	( )	( )	( )
Compatibilidade de navegadores	( )	( )	( )

Em relação a Eficiência			
	ruim	regular	bom
Performance	( )	( )	( )
Acessibilidade	( )	( )	( )

Em relação a Confiabilidade			
	ruim	regular	bom
Validação de dados nos formulários	( )	( )	( )
Processamento corretos dos links: Os links apontam para o local correto	( )	( )	( )
Auxilio para correção de erro	( )	( )	( )

Em relação aos critérios avaliados			
	ruim	regular	bom
Usabilidade	( )	( )	( )
Funcionalidade	( )	( )	( )
Eficiência	( )	( )	( )
Confiabilidade	( )	( )	( )

Em relação a utilização do sistema móvel, assinale a alternativa que melhor representa a sua opinião quanto ao uso do sistema

Em relação a Usabilidade			
	ruim	regular	bom
Compreensibilidade do sistema	( )	( )	( )
Mecanismos de ajuda e retorno	( )	( )	( )
Aspectos de Interface	( )	( )	( )
Legibilidade (Clareza no conteúdo apresentado)	( )	( )	( )
Navegabilidade	( )	( )	( )

Em relação a Funcionalidade			
	ruim	regular	bom
Autenticação do sistema	( )	( )	( )
Segurança	( )	( )	( )

Em relação a Eficiência			
	ruim	regular	bom
Performance	( )	( )	( )
Acessibilidade	( )	( )	( )

Em relação a Confiabilidade			
	ruim	regular	bom
Validação de dados nos formulários	( )	( )	( )

Processamento corretos dos botões: Os botões apontam para o local correto	( )	( )	( )
Auxilio para correção de erro	( )	( )	( )

Em relação aos critérios avaliados			
	ruim	regular	bom
Usabilidade	( )	( )	( )
Funcionalidade	( )	( )	( )
Eficiência	( )	( )	( )
Confiabilidade	( )	( )	( )