Universidade Federal do Rio de Janeiro

Escola Politécnica

Departamento de Eletrônica e de Computação

Sistema de Monitoramento de Câmeras e Análise de Indicadores de Comportamento Humano a Estímulos Visuais do Comércio

Autor:

Helainy Ignácio de Almeida Torres

Orientador:

Prof. Jorge Lopes de Souza Leão, Doc. Ing.

Examinador:

Ana Carolina Ferraz Mendonça de Souza, D. Sc.

Examinador:

Aloysio de Castro Pinto Pedroza, Dr.

DEL

Agosto de 2013

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
Escola Politécnica - Departamento de Eletrônica e de Computação
Centro de Tecnologia, bloco H, sala H-217, Cidade Universitária
Rio de Janeiro - RJ CEP 21949-900

DEDICATÓRIA

Dedico esse trabalho a minha Mãe, Maria do Carmo, por ter acreditado em mim em todas as fases da minha vida.

AGRADECIMENTO

Quero agradecer a minha família, que sempre me incetivou e me apoiou durante a graduação.

Agradeço também aos muitos amigos, colegas e professores que participaram direta ou indiretamente do processo de conclusão do curso; pela força, apoio e coragem nos muitos e muitos momentos de desânimo.

Agradeço em especial ao Raphael, por ter sido um grande amigo e companheiro, pela paciência e compreensão nos diversos momentos em que tive que me ausentar do convívio social para me dedicar as provas, projetos e trabalhos.

E a ICE interactive por ter me aberto as portas quando tudo estava perdido.

"To indo.
Fiz coisas certas que deram errado
Fiz coisas erradas que deram certo
Fiz bem feito, ficou incompleto
Fiz mau feito, ficou perfeito
Fiz o que deu, faltou
Fiz o que não deu, sobrou
Fiz de conta que não gostei, foi bom
Fiz de conta que gostei, foi ruim.
Fiz e farei enquanto viver,
Resultado ... só depois vou saber."
J.Ruy

RESUMO

O crescimento significativo do poder de compra dos consumidores, gerado pela facilidade de parcelamento em cartões de crédito e carnês de compra, e pela redução de taxas e impostos sobre os produtos, trouxe ao mercado de shoppings e lojas de varejos um dinamismo que não se esperava.

É comum encontrarmos dados setoriais com informações desencontradas sendo divulgadas pela imprensa, associações e, inclusive, em relatórios utilizados pelas empresas deste setor, que frequentemente tomam decisões baseadas em informações pouco robustas. Esses dados frequentemente são calculados com base em estimativas ou generalizações que podem não corresponder à realidade específica de uma determinada loja.

O objetivo deste projeto é dar aos agentes econômicos do setor - lojistas, empreendedores e investidores - um retrato fiel do que acontece no cenário de sua loja através de uma ferramenta de análise de indicadores baseada em informações coletadas por meio de câmeras instaladas na própria loja. Esta ferramenta permitirá a esses agentes econômicos avaliar se sua estratégia de vendas está atingindo o público-alvo correto (identificada por gênero e faixa etária), auxiliando-os em tomadas de decisões estratégicas a partir do seu próprio histórico.

Palavras-Chave: Mercado de Consumo, Indicadores, Estratégia, Computação em nuvens, Google Web ToolKit, Google App Engine.

ABSTRACT

The signicant growth of the purchasing power of consumers, generated by installment facility by credit cards and purchase booklets, and reduction of fees and taxes on products, brought to malls market and retails stores a dynamism that was not expected.

It is common to find industry data with misinformation being disseminated by the press, associations, and even in reports used by companies in this sector, which frequently make decisions based on week information. These data are often calculated based on estimates or generalizations that may not correspond to the specific reality of a particular store.

The objective of this project is to give economic agents in the industry - retailers, entrepreneurs and investors - an accurate description of what happens in the scenario of their business through an indicator analysis tool based on information collected through cameras installed in the store. This tool will allow these economic agents evaluate if their sales strategy is reaching the correct target audience (identified by gender and age), assisting them in strategic decision making based on their own background.

Key-words: Consumer Market, Indicators, Strategy, Cloud computing, Google Web Toolkit, Google App Engine.

SIGLAS

AJAX - Asynchronous Javascript and XML

API - Application Programming Interface

BigTable - Local de persistência de objetos do App Engine

CaaS - Communication as a Service

CSS - Cascanding Style Sheets

DaaS - Desenvolvimento como um Serviço

DHTML - Dynamic HTML

DOM - Document Object Model

Gadgets - Software ou serviço que pode ser agregado a um ambiente maior

GAE - Google Append Engine

GQL - Google Query Language

GUI - Grafical User Interface

GWT - Google Web Toolkit

HTML - HyperText Markup Language

IaaS - Infrastructure as a service

ID - Indentificador de um objeto ou dado

IDE - Ambiente de Desenvolvimento Integrado

JDBC - Java Database Connectivity

JDO - Java Data Objects

JPA - Java Persistence API

JRE - Java Runtime Environment

JSF - Java Server Faces

JSNI - Java Script Native Interface

JSP - Java Server Pages

LAN - Local Area Networks

NUC - Next Unit of Computing

OTS - Opportunity to see

OTB - Opportunity to buy

PaaS - Plataforma as a Service

PHP - Personal Home Page

RIA - Rich Internet Application

RMI - Remote Method Invocation

RPC - Remote Method Call

SaaS - Software as a service

SAD - Sistemas de Apoio à Decisão

SQL - Sequencial Query Language

URL - Uniform Resource Locator

WAN - Wide Area Networks

WLAN - Wide Local Area Networks

Widgets - Componentes de interface gráfica

W3C - World Wide Web Consortium

XML - eXtensible Markup Language

Sumário

1 Introdução **1**

1.1 Tema 1

1.2 Delimitação 1

1.3 Justificativa 2

1.4 Objetivos 2

1.5 Metodologia 3

1.6 Materiais 3

1.7 Descrição 4

2 Visão Geral do Sistema **5**

2.1 Horus - Analytics da Vida Real 5

2.2 O Uso Estratégico de Sistemas da Informação 7

3 Fundamentação Teórica **9**

3.1 Computação em Nuvem 9

3.2 Google Application Engine 12

3.2.1 Sandbox 14

3.2.2 BigTable 15

3.3 Google Web Toolkit 17

3.4 Highcharts 18

4 Arquitetura Interna do Software **19**

4.1 Cliente 20

4.2 Servidor 21

4.3 Comunicação Cliente-Servidor 22

4.4 Comunicação Servidor-Banco de Dados 23

4.5 Dashboard - Highcharts 26
4.6 Transmissão de Dados para a Nuvem 27
4.7 Monitoramento 27

5 Funcionamento do Sistema **30**
5.1 Instalação do Sistema 31
5.2 Envio de Dados 32
5.3 Monitoramento do Sistema 33
5.3.1 Monitoramento Local 33
5.3.2 Monitoramento na Nuvem 34
5.4 Definição dos usuários e permissões 35
5.5 Definição dos dados e das métricas 36
5.6 Utilização do Analytics 38
5.7 Análise dos dados 40
5.7.1 Exemplo 1 40
5.7.2 Exemplo 2 41
5.7.3 Conclusões 41

6 Resultados **43**
6.1 Velocidade de resposta para uma quantidade massiva de dados 44
6.2 Sincronismo entre os sistemas de monitoramento 44
6.3 Compatibilidade com navegadores e dispositivos 45
6.4 Clareza de usabilidade 45
6.5 Resposta do cliente 45

7 Conclusões **47**

8 Projetos Futuros **48**

Lista de Figuras

2.1 Interface do dashboard para o usuário final: análise de audiência separada por faixa etária 6
2.2 Diagrama de blocos do sistema Horus 7
2.3 Análise comparativa entre dois períodos 8

3.1 Ilustração de uma nuvem servindo aplicativos a diversos dispositivos . 10
3.2 Camadas de Arquitetura de Serviços 11
3.3 Console de administração do app engine da Google 13

4.1 Menus de cadastro e tela de cadastro de câmeras 21
4.2 Entitades e espaço ocupado na aplicação 22
4.3 Modelo Entidade-Classes 24
4.4 Requisições do monitoramento 29

5.1 Tela inicial da aplicação final 30
5.2 Logitech QuickCam Pro 9000 e Next Unit of Computer respectivamente 31
5.3 Visualização do monitoramento automático na nuvem 34
5.4 Usuários 35
5.5 Audiência classificada por gênero e OTS 37
5.6 Funil de análises métricas 38
5.7 Visualização de vitrines nas lojas 39
5.8 Funcionalidades da Tela de Dashboard 39

Lista de Tabelas

3.1 O custo dos recursos de computação do App Engine do Google 13
3.2 Relacionamento entre entidades 16

4.1 Comunicação cliente-servidor através de servlets 23

Capítulo 1

Introdução

1.1 Tema

O tema do trabalho é uma plataforma para a visualização de dados demográficos (gênero e faixa etária) da audiência de mídias, e a análise do desempenho do conteúdo exibido por elas através de métricas baseadas nesses dados, para saber se estão alcançando o público-alvo esperado da maneira esperada.

Por mídia entende-se qualquer meio utilizado para chamar a atenção do público de forma visual, seja um banner, um telão ou uma vitrine. Por audiência entende-se todo o público impactado por essa mídia quando olha para ela.

Através desta plataforma, um comerciante ou qualquer outro interessado pode avaliar se estão atraindo o público-alvo desejado (em termos de gênero e faixa etária) e também tomar decisões estratégicas com base nos dados e métricas apresentados. Por exemplo: ao verificar que o público mais impactado pela sua vitrine é o de mulheres jovens, um lojista pode adequar a sua estratégia ou até a sua equipe de vendas para melhor atender a esse público.

1.2 Delimitação

O trabalho consiste em prover uma interface gráfica de visualização de indicadores gerados pelo processamento de imagens de câmeras localizadas na vitrine da loja. Também será gerado um sistema automático de monitoramento do funcionamento dessas câmeras para garantir a integridade do sistema. O trabalho não

inclui o processamento das imagens das câmeras, o seu papel é permitir que qualquer usuário consiga fazer análises e relátorios gráficos com os indicadores oferecidos e janelas de tempo variáveis.

Para esse projeto, as mídias serão as vitrines de lojas, o conteúdo exibido serão os itens expostos nelas, e o público serão as pessoas que passam em frente a essas vitrines.

Para garantir a confiabilidade do sistema todas as Empresas, Lojas e Usuários são cadastradas para limitar o acesso aos dados de pessoas não autorizadas.

Os dados utilizados no desenvolvimento do trabalho serão gerados artificialmente pois o objetivo não é avaliar o sistema de processamento de imagens e sim a forma como os dados são apresentados: com clareza, rapidez e confiabilidade. E também para garantir a privacidade dos dados dos clientes.

1.3 Justificativa

A sociedade conteporânea é extremamente consumista. Porém, com o excesso de produtos oferecidos no mercado, é dificil para um comerciante analisar se sua estratégia de vendas está atingindo o público-alvo desejado.

Preços baixos, luzes, enfeites, cores podem causar impactos postivos ou negativos.

É importante ter uma resposta rápida a impactos negativos e/ou um histórico de estratégias baratas e acessíveis de sucesso.

Apesar do projeto ter sido desenvolvido para o varejo o sistema pode ser adaptado para estudos de comportamento humano em outros segmentos, como bens de consumo e digital signage.

1.4 Objetivos

O objetivo geral é, então, prover uma ferramenta que permita: (1) Visualizar através de gráficos a variação do comportamento do público ao longo do tempo; (2) Visualizar métricas e indicadores de desempenho da sua vitrine; (3) Garantir que as câmeras de captura de dados estão funcionando corretamente, e; (4) Garantir que os dados foram processados e armazenados no banco de dados para futuras análises.

1.5 Metodologia

A obtenção de dados se dá através da análise de imagens de câmeras posicionadas na vitrine da loja. Essas imagens são processadas por um software proprietário, de um parceiro da ICE Interactive, que produz como resposta o gênero e a faixa etária da pessoa que ele detectou. Essas informações associadas ao tempo de visualização, data e hora são armazenadas num servidor na nuvem, o Google App Engine (GAE). No momento da análise, o usuário escolhe a forma como esses dados serão agrupados e processados; construindo indicadores que permitam avaliar tempo e/ou quantidade de visualização agrupada por grupos de sexo e faixa etária.

O sistema que salva os dados processados é executado automaticamente em horários programados, para permitir o agrupamento dos dados e minimizar seu envio para a nuvem, já que o GAE, que é um sistema pago, cobra pelo fluxo de *bits*. Por essa razão um sistema de monitoramento de dados e um de monitoramento de câmeras ambos automáticos foram desenvolvidos para garantir remotamente a integridade dos dados. Caso alguma anomalia seja encontrada os administradores serão notificados (atualmente por e-mails) e entidades de erros são criadas no banco de dados.

1.6 Materiais

As câmeras utilizadas em testes são propriedade da empresa ICE SOFTWARES LTDA, situada na encubadora de empresas da COPPE, a licença do *software* de processamento de imagens também pertence à mesma, embora os dados provenientes dessas câmeras não foram utilizados para o desenvolvimento desse projeto a fim de preservar as empresas e lojas que contratarem os serviços.

Os dados utilizados no desenvolvimento dessa monografia serão gerados artificialmente.

O Google App Engine (GAE) disponibiliza uma cota diária limitada para desenvolvedores que será suficiente para o desenvolvimento desta monografia e o *Highcharts*, componente utilizado para geração de gráficos do dashboard também é permitido para desenvolvimento de aplicações em fase de testes e só será cobrado quando esta aplicação estiver no mercado.

A ICE SOFTWARE LTDA arcará com os custos dessas ferramentas para sua aplicação final comercial.

1.7 Descrição

O capítulo 2 trás uma introdução geral ao sistema e o seu papel como ferramenta de apoio gerencial em um ambiente comercial.

O capítulo 3 descreve as bases teóricas das tecnologia utilizadas na implementação do sistema. A arquitetura interna da aplicação principal e das aplicações complementares que constituem o Sistema Horus serão detalhadas no capítulo 4.

O capítulo 5, faz uma visão de 360° do sistema, desde a montagem do *hardware* necessário, a instalação e a execução dos aplicativos auxiliares para manter o sistema funcionando, à utilização básica do sistema e exemplos de análises práticas do dia-a-dia.

O capítulo 6, fala sobre os testes aos quais a ferramenta foi avaliada e seus resultados.

A conclusão está no capítulo 7 e os projetos e pespectivas futuras finalizam a monografia no capítulo 8.

Capítulo 2

Visão Geral do Sistema

2.1 Horus - Analytics da Vida Real

O sistema Horus, que foi desenvolvido ao longo desse projeto, é um aplicativo web, que permite que através do processamento de imagens das câmeras situadas na vitrine das lojas, analisar o comportamento das pessoas que circulam em frente à loja.

As combinações de eventos como passar em frente, olhar, parar e entrar na loja são transformadas em indicadores que, associados às informações de gênero e faixa etaria, traduzem reações comportamentais a uma determinada estratégia de marketing.

Esses eventos são capturados por uma webcam e processados num minicomputador NUC (Next Unit of Computer) localizado junto a vitrine da loja, de forma estratégica para não ser percebido e influenciar nas análises. Os dados processados são enviados via conexão 3G na forma da classe de dados denominada CIndicator[1] e armazenados na nuvem da Google.

Para o usuário final, a aplicação é apresentada na forma de dashboard (Figura 2.1), que permite que ele escolha os indicadores que deseja visualizar, a janela temporal da análise, e também a exibição dos dados separados de acordo com gênero ou faixa etária. Também é possível que seja feita uma comparação gráfica entre dois períodos escolhidos pelo usuário, a fim de avaliar o histórico de comportamento da audiência.

[1] Como será melhor detalhado no capítulo 4

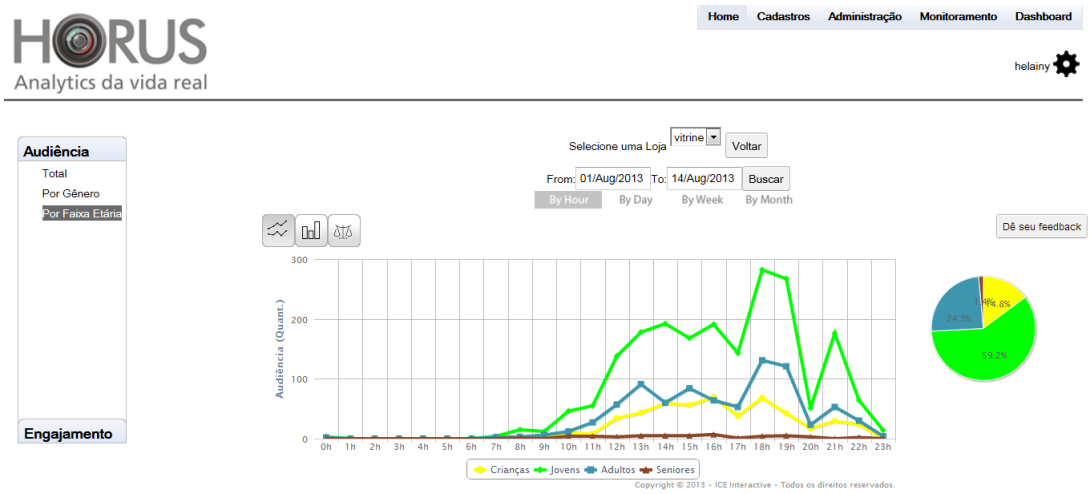

Figura 2.1: Interface do dashboard para o usuário final: análise de audiência separada por faixa etária

Complementando o sistema há mais duas aplicações: A primeira é responsável pelo monitoramento do sistema, verificando se as câmeras e computadores estão funcionando corretamente. A segunda é responsável pelo envio dos dados para o sistema Horus. Ambos são automáticos, funcionando independente de ações humanas externas, exceto em casos onde seja detectado um problema (câmera sem sinal, computador travado, falta de acesso à internet, imagem da câmera obstruída, etc.)

O sistema é proprietário da empresa ICE Interactive, situada na encubadora da Coppe, na qual sou estagiária, permitindo-me o desenvolvimento do projeto. Na Figura 2.2 podemos visualizar o diagrama de blocos do sistema em uma visão geral.

O processamento da imagem capturada e a análise de eventos não são abordadas nesse projeto apenas o seu processamento na forma de indicadores, transmissão de dados, monitoramento e a interface de análise do usuário .

O processamento da imagem é feito por por um software de uma empresa parceira da ICE Interactive, seu método de análise não pôde ser divulgado por questões empresariais. Outros softwares foram avaliados para o processamento das imagens mas este foi o que apresentou maior grau de confiabilidade nos testes executados, gênero: 86% e faixa etária: 75%, levando-se em consideração que o processamanto é realizado em alguns minutos. Processamento mais complexo e mais confiável podem ter uma reposta mais lenta e isso depende da necessidade do cliente.

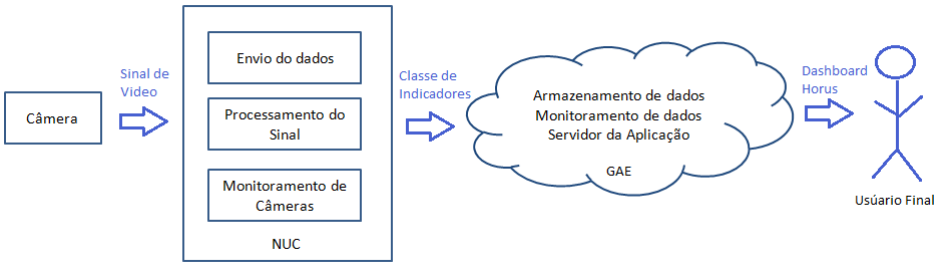

Figura 2.2: Diagrama de blocos do sistema Horus

2.2 O Uso Estratégico de Sistemas da Informação

As organizações em geral usam sistemas de informação para dar apoio às suas decisões. No caso de mercado de shoppings e lojas de varejo, essas decisões têm como objetivo aumentar as suas vendas e se destacar em um cenário de alta competitividade.

No varejo, a forma mais comum de exposição de produtos é através de vitrines. A decoração da vitrine normalmente é feita por um profissional qualificado; luzes, cores, faixas de preços e diversos apetrechos podem ser utilizados para atrair o público, normalmente levando em consideração datas comemorativas ou períodos do ano.

O uso bem-sucedido de sistemas da informação nesse setor envolve a identificação de áreas decisivas para o sucesso na escolha da estratégia de chamar a atenção do público-alvo para a sua loja. Numa visão mais geral, esses sistemas poderiam ser utilizados para analisar setores de maior fluxo de pessoas e sua classificação por gênero e faixa etária pode permitir a instalação de novas lojas voltadas para esse setor específico. [37]

O sistema Horus veio integrar uma solução já existente, uma tecnologia de identificação e contagem de faces, a um mercado carente e em constante mudança. A dinâmica do processo permite aos agentes econômicos - lojistas, empreendedores e investidores - uma análise quantitativa e também comparativa de sua estratégia numa interface de fácil acesso e compreensão (Figura 2.3) sem a necessidade de aguardar relatórios longos, algumas vezes de qualidade questionável e talvez atrasados, já que nesses casos “tempo é dinheiro”.

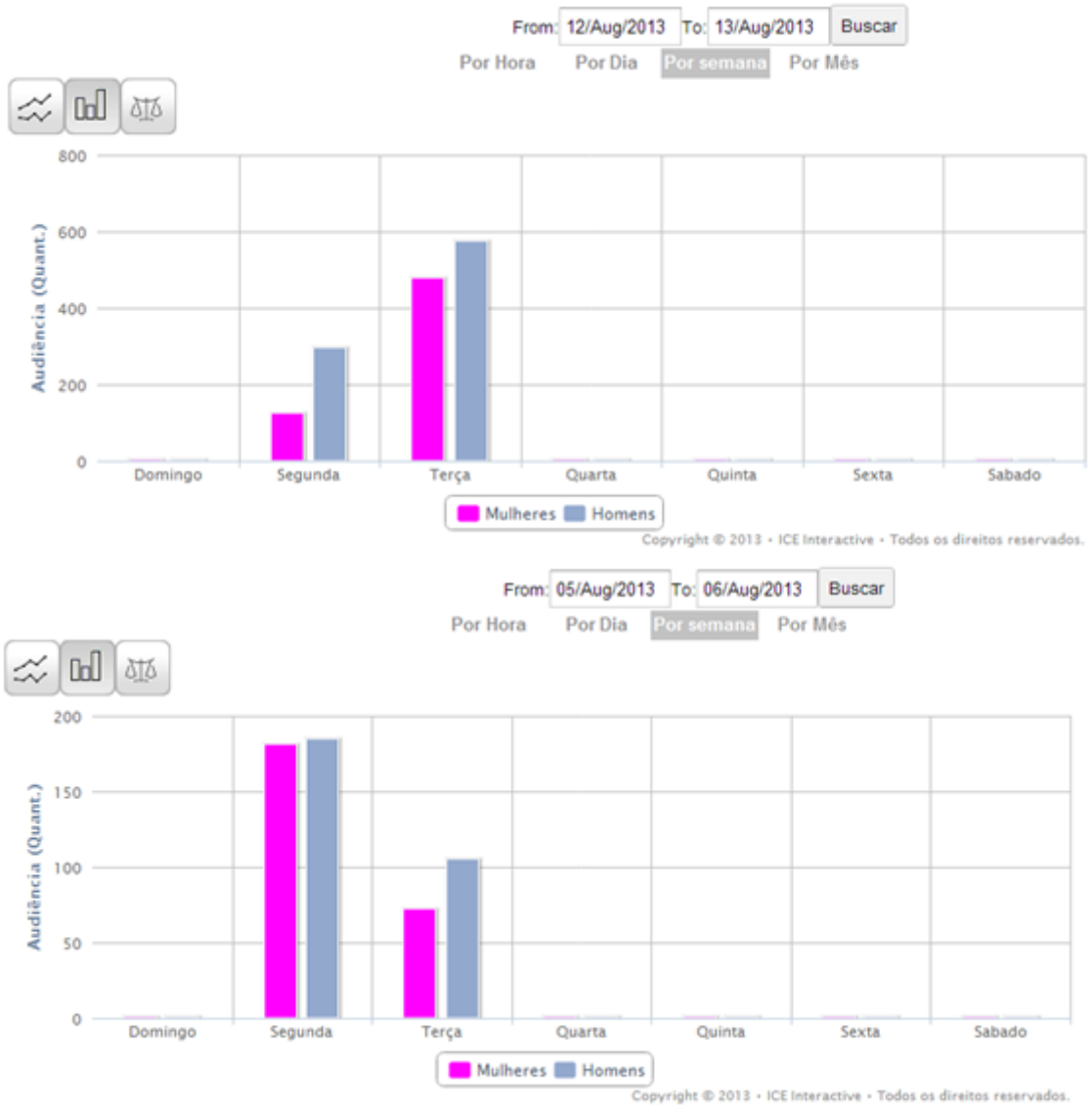

Figura 2.3: Análise comparativa entre dois períodos

Capítulo 3

Fundamentação Teórica

Esse capítulo traz uma breve abordagem das técnicas e tecnologias que foram utilizadas no desenvolvimento desse projeto. Dentre elas podemos destacar a computação em nuvem, que toma parte neste projeto através do serviço do Google App Engine (GAE). Vamos analisar as suas características e suas diferenças em relação à maioria dos sistemas de nuvem, o fato de não ser IaaS (Infraestrutura como um serviço) e nem SaaS (Software como um Serviço), e seu sistema de armazenamento de dados (Bigtable) que introduz conceitos de banco de dados orientado a colunas.

Também vamos falar da API Google Web Toolkit (GWT), utilizada na criação da interface com o usuário, de fácil utilização e compatibilidade com diversos navegadores e sobre a biblioteca Highcharts utilizada na visualização dos gráficos do dashboard.

3.1 Computação em Nuvem

A computação em nuvem é uma tendência recente da tecnologia cujo objetivo é proporcionar serviços de Tecnolgia da Informação (TI) sob demanda com pagamento baseado no uso e armazenamento de dados. Ela veio substituir a necessidade de construir infraestruturas de TI complexas, onde usuários têm que realizar instalação, configuração e atualização de sistemas de software. Em geral, os recursos de computação e hardware correm o risco de ficar obsoletos, e muitas empresas optam por terceirizar esses o serviço para evitar problemas e riscos.

Na computação em nuvem os recursos de TI são fornecidos como serviços,

permitindo que usuários os acessem sem necessidade do conhecimento da tecnologia utilizada. Dessa forma usuários e empresas passam a acessar os serviços sob demanda.

Para os desenvolvedores de software, que movem suas aplicações para a "nuvem" a vantagem que o modelo fornece é similar à do usuário final: quando um software é desenvolvido, o aplicativo é vendido como um produto a ser instalado no computador do usuário, ou seja, devem ser criadas versões para diversos sistemas operacionais e infra-estruturas. Com o modelo em "nuvens", basta que o desenvolvedor crie uma aplicação, teste e rode a aplicação na plataforma que escolher e disponibilizar na "nuvem" para que usuários a utilizem. A Figura 3.1 ilustra este cenário.

A ideia da computação em nuvem consiste em que tudo é tratado como serviço disponibilizado ao usuário, podendo ele ser um desenvolvedor ou usuário final.

Figura 3.1: Ilustração de uma nuvem servindo aplicativos a diversos dispositivos

Muitos serviços de computação em nuvem são oferecidos pelo modelo chamado *Utility Computing*, o qual é definido como um pacote de recursos computacionais medidos e cobrados de forma semelhante aos serviços de utilidade pública, como eletricidade, água, telefone ou banda larga. Este modelo prevê uma otimização dos investimentos feitos pelos clientes nos recursos de hardware, diminuindo a necessidade de grandes investimentos iniciais, possibilitando o aluguel de mais recursos à medida que forem necessários, podendo ainda alguns provedores de serviços se ajustarem dinamicamente às necessidades de demanda, para momentos de pico, evitando

que recursos fiquem ociosos a maior parte do tempo.

Atualmente os serviços de computação em nuvem possuem três categorias bem definidas de acordo com os recursos e o modo que estes recursos são disponibilizados. Estes modelos são importantes, pois eles definem um padrão arquitetural para soluções de computação em nuvem.

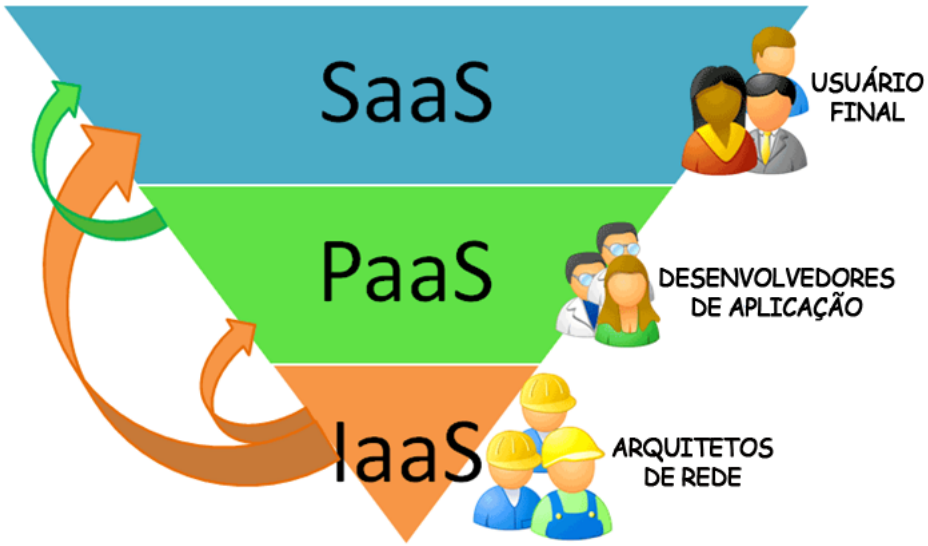

Figura 3.2: Camadas de Arquitetura de Serviços

- Software como um Serviço (SaaS): O modelo de SaaS proporciona sistemas de software com propósitos específicos que são disponíveis para os usuários por meio da Internet e acessíveis a partir de vários dispositivos do usuário por meio de uma *interface thin client* como um navegador Web. No SaaS, o usuário não administra ou controla a infraestrutura subjacente, incluindo rede, servidores, sistema operacional, armazenamento ou mesmo as características individuais da aplicação, exceto configurações específicas. [27]

- Plataforma como um Serviço (PaaS): O modelo de PaaS fornece sistema operacional, linguagens de programação e ambientes de desenvolvimento para as aplicações, auxiliando a implementação de sistemas de software. Assim como no SaaS, o usuário não administra ou controla a infraestrutura subjacente, mas tem controle sobre as aplicações implantadas e, possivelmente, as configurações de aplicações hospedadas nesta infraestrutura. [27]

- Infraestrutura como um Serviço (IaaS): A IaaS torna mais fácil e acessível o fornecimento de recursos, tais como servidores, rede, armazenamento e outros recursos de computação fundamentais para construir um ambiente de aplicação sob demanda, que podem incluir sistemas operacionais e aplicativos. Em geral, o usuário não administra ou controla a infraestrutura da nuvem, mas tem controle sobre os sistemas operacionais, armazenamento, aplicativos implantados e, eventualmente, seleciona componentes de rede, tais como firewalls.[27]

3.2 Google Application Engine

O Google foi um dos primeiros a implantar uma plataforma em "nuvem" aberta para desenvolvimento de aplicativos. O Google Application Engine, App Engine ou GAE foi um dos pioneiros na implantação da estrutura PaaS, pois fornecia infra-estrutura para os usuários e uma plataforma de desenvolvimento. Dessa forma usuários passaram a crias aplicativos para a nuvem e não utilizar apenas os já fornecidos pela mesma alavancando a estrutura como uma plataforma de hospedagem.[1]

Com o Google App Engine, pode-se criar aplicativos web nos mesmos moldes de sistemas escaláveis dos aplicativos do Google. Os aplicativos implantados no Google App Engine são fáceis de criar, manter e escalar à medida que seu tráfego e armazenamento de dados precisam crescer. Uma vez implantado o sistema no App Engine, os clientes já poderão acessar e usar o aplicativo. Esse processo facilita bastante o desenvolvimento de aplicações, pois a lógica pode ser testada e atualizada a qualquer momento. O Google disponibiliza também um dashboard como ferramenta de análise de sua aplicação, como mostra a Figura 3.3.

O Google é conhecido por ser altamente confiável e por sua infraestrutura de alto desempenho. Com o Google App Engine, é possivel beneficiar-se dos 10 anos de experiência e conhecimento que o Google possui em executar sistemas altamente escaláveis e orientados para o desempenho. As mesmas políticas de segurança, privacidade, escalabilidade e proteção de dados para os seus aplicativos se aplicam a todos os aplicativos do Google App Engine. O Google oferece um leque de API's para acessos a seus serviços, para que a nuvem da possa ser utilizada em diversos

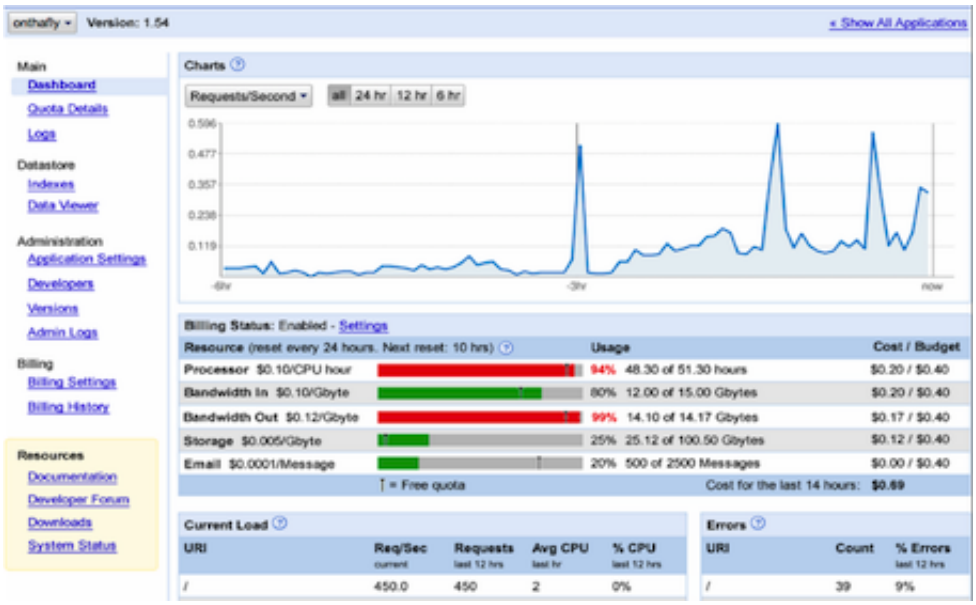

Figura 3.3: Console de administração do app engine da Google

sistemas.

O Google App Engine sempre será gratuito para iniciar uma aplicação, afirma o Google, até o limite de 500 MB de armazenamento e 5 milhões de acessos de página mensais. Quando este limite for atingido, é possível ativar o faturamento através do console de administração do aplicativo, pagando apenas pelos recursos consumidos. A Tabela 3.1 apresenta esses valores (em dólares americanos). [10]

Além disso, cada conta de desenvolvedor pode conter até dez aplicativos.

Tabela 3.1: O custo dos recursos de computação do App Engine do Google

Recurso	Unidade	Custo unitário
Largura de banda de saída	gigabytes	$ 0,12
Largura de banda de entrada	gigabytes	$ 0,10
Tempo de CPU	Horas da CPU	$ 0,10
Dados armazenados	gigabytes por mês	$ 0,18
Destinatários de e-mail	destinatários	$ 0,0001

Fonte: BILLING [10]

O Google Application Engine possui suporte a aplicativos criados com o uso das linguagens de programação Python, Go e Java, e, através dessa última, várias

outras baseadas na máquina virtual Java. O App Engine permite o uso de parte das bibliotecas padrões da linguagem Java. Para diminuir o efeito dessas limitações, foi criada uma série de serviços que facilitam o desenvolvimento e execução de determinadas tarefas. [1]

Os aplicativos criados e disponibilizados a partir do Google App Engine podem ser acessados a partir do domínio livre no padrão appspot.com oferecido pelo serviço, adicionado ao identificador da aplicação escolhido na hora da sua criação, ficando da forma: identificador.appspot.com. (ex: http://www.testehorus.appspot.com/ domínio criado para o estudo de caso).

3.2.1 Sandbox

O Application Engine, implementa de forma inovadora o paradigma de computação em nuvem, em que o armazenamento e processamento do aplicativo são distribuídos entre os clusters do Google, e necessita de um ambiente virtual seguro para cada aplicativo. Este ambiente, chamado de sandbox, disponibiliza acesso limitado ao sistema operacional. Esta limitação possibilita que o Google App Engine distribua as solicitações de web da aplicação entre diversos clusters, podendo iniciar ou interromper os servidores para atender às demandas de tráfego.[4]

Para que isso seja possível algumas limitações são impostas ao aplicativo. Dentre elas podemos citar as mais relevantes segundo o Google:[19]

1. O aplicativo pode acessar outros computadores na internet somente através de solicitações feitas utilizando o protocolo HTTP ou HTTPS por meio dos serviços de obtenção de URL e de e-mail. Outros computadores podem conectar-se à aplicação somente fazendo solicitações HTTP ou HTTPS nas portas padrão;

2. Não é permitido que a aplicação grave no sistema de arquivos. Apenas é permitida a leitura de arquivos enviados juntamente com o código da aplicação. Deve-se utilizar o sistema de armazenamento de dados, cache de memória ou outros serviços do Google App Engine para todos os dados que devam ser persistidos durante as solicitações;

3. É executado o código da aplicação somente em resposta a uma solicitação da

web ou uma tarefa programada (Cron Job), devendo retornar uma resposta em no máximo 30 segundos em ambos os casos. Não é permitida à aplicação gerar subprocessos (Threads) ou executar código após a resposta.

3.2.2 BigTable

O sistema de armazenamento de dados do App Engine utiliza uma arquitetura distribuída que apesar de ser consistente, não é um banco de dados relacional, embora tenha recursos similares ao mesmo. Suas características exclusivas exigem uma maneira diferente de gerenciar e projetar dados. Esse banco é denominado Bigtable [1].

BigTable é um banco de dados distribuído designado para o gerenciamento de grandes quantidades de dados estruturados. Suporta massas de dados na casa dos Petabytes através de milhares de servidores espalhados pelo mundo[41].

Os dados de uma aplicação podem estar distribuídos em dezenas de máquinas e em diferentes localidades, possivelmente em diferentes lugares ao redor do mundo.

Armazenar dados em um ambiente distribuído pode acarretar grandes complicações para o desenvolvimento da aplicação, porém graças ao Google App Engine esta tarefa não é de preocupação do desenvolvedor, pois a API disponibilizada gratuitamente pelo Google se encarrega de realizar toda a distribuição, replicação e carga de dados e ainda oferece um poderoso mecanismo de consulta e transações.

Realizando uma comparação com o modelo de banco de dados tradicional (banco de dados relacional), os tipos seriam as tabelas, as entidades seriam as linhas, enquanto as propriedades seriam os campos ou colunas de uma tabela [19].

A API de armazenamento de dados conta com uma linguagem de consulta parecida com o SQL (Structured Query Language), o GQL (Google Query Language).

O método de armazenamento de dados no Google App Engine é responsável por armazenar e executar consultas sobre objetos de dados conhecidos como entidades.

Uma propriedade pode ser referência de outra entidade através de sua “Key”. Na tabela 3.2 exemplificamos essa relação: Entity key é o valor da chave que relaciona as entidades e Decoded entity key sua decodificação. Ou seja, uma entidade

Decoded entity key	$CCompany : id = 4 > CStore : id = 2 > CVitrine : id = 9$
Entity key	$agxzfnRGGluZyB0ZXJzDAsSBkNTdG9yZRgCDA$

Tabela 3.2: Relacionamento entre entidades

conhece a sua "familia" de entidades.[1]

NoSQL

Atualmente existem diversos bancos de dados NoSQL desenvolvidos principalmente por empresas que possuem grandes quantidades de dados, exemplos são o Google e a Amazon, que desenvolveram respectivamente a BigTable e o DynamoDB. Diferentes bancos de dados NOSQL possuem características peculiares, porém todos possuem a característica em comum de serem banco de dados não-relacionais.

No modelo convencional de banco de dados, uma informação está dividida em várias tabelas, reduzindo assim a redundância de informações, porém quando a quantidade de registros é muito grande, a busca se torna lenta. Já no NoSQL, os registros encontram-se em geral agrupados, isso melhora a velocidade de busca, e em contrapartida, aumenta a repetição de informações. O modelo não-relacional, trabalha de forma semelhante a arrays, nele existem coleções que podem ser agrupadas de diversas maneiras como, por exemplo, por colunas e linhas, como é o caso da BigTable.

Na questão de escalabilidade, a maior vantagem do NoSQL está no fato de não precisar se preocupar em dividir as relações entre tabelas na hora de adicionar mais um servidor.

Todas as técnicas utilizadas para escalar bancos de dados relacionais são utilizas nos não-relacionais, porém o desenvolvedor não as percebe. [49]

GQL

O GAE datastore utiliza uma chave como identificador único para atributos como linha e pode ser acessado com a linguagem de consulta Google Query Language (GQL), um subconjunto da linguagem SQL. A linguagem GQL possui suporte a seleção, ordenação e alguns operadores. A instrução de seleção pode ser realizada

em uma única tabela, ou seja, não suporta junções, assim como associações ou chaves compostas em decorrência do modelo de dados utilizado.

Dessa forma, associações devem ser implementadas na camada de aplicação. Segue abaixo um exemplo de consulta simples permitida pelo GQL. Nota-se a semelhança com o SQL.

```
SELECT * FROM CIndicator-Type
```

3.3 Google Web Toolkit

O Google Web Toolkit(GWT) é um kit de desenvolvimento de aplicações web do Google, que permite desenvolver interfaces Ricas para Internet (RIA - Rich Internet Application), escrevendo código fonte na linguagem de programação Java, compilado e gerando uma saída em código JavaScript. Aplicações Ricas para Internet são Aplicações Web que tem funcionalidades de softwares tradicionais do tipo Desktop. RIA típicos injetam todo o processamento da interface para o navegador da internet, que assincronamente dispara pedidos de processamento para o servidor de aplicação, com tecnologia JavaScript e XML, o AJAX.[2]

A formulação do GWT foi elaborada para a utilização da plataforma GAE, pois a padronização da estrutura do projeto facilita escalar a aplicação desenvolvida e é uma alternativa ao JSF para o desenvolvimento orientado a componentes.

A ferramenta trabalha com o código dividido em duas partes distintas, uma com código a ser executado pelo cliente e outra com código a ser executado pelo servidor através de chamadas assíncronas feitas pelo código no cliente. O código Java destinado a ser executado pelo cliente é automaticamente compilado pelo Google Web Toolkit para código JavaScript capaz de ser executado por diversos navegadores, deixando a cargo do compilador questões de compatibilidade entre navegadores e de otimização.

O próprio Google utiliza esta ferramenta em vários de seus aplicativos: o Gmail, a rede social G+ e o conjunto de ferramentas como processadores de texto e planilhas do Google Docs.

O GWT pode ser utilizado por qualquer aplicativo Java para web e possui integração com o Google Application Engine, sendo distribuído juntamente com seu

plug-in para Eclipse o Googlipse.

Desde seu lançamento na conferência JavaOne em maio de 2006, o GWT vem evoluindo passando da versão 1.0 para a 2.5 atualmente. [48]

Uma desvantagem do GWT atualmente é que suas páginas não podem ser indexadas pelos mecanismos de busca, uma vez que elas são geradas dinamicamente. Para contornar o problema é usado cloaking, que é uma técnica utilizada por webmasters para entregarem conteúdos diferentes de uma mesma URL para visitantes específicos do site.

O GWT possui três componentes básicos: o compilador Java-to-JavaScript, o emulador Java Runtime Environment (JRE) e a bliblioteca de interface gráficas com o usuário (GUI).

3.4 Highcharts

Highcharts é uma biblioteca de gráficos escritos em JavaScript puro, oferecendo gráficos intuitivos e interativos para seu site ou aplicação web. Highcharts atualmente suporta linha, spline, área, área-spline, coluna, barra, pizza e tipos de gráfico de dispersão.

GWT Highcharts é uma biblioteca de código aberto disponível gratuitamente que fornece uma abordagem completa e elegante recurso para incluir Highcharts dentro de um aplicativo GWT usando puro código Java (incluindo GWT bibliotecas widget como SmartGWT ou Ext GWT).[47]

Essa biblioteca é livre para desenvolvimento de aplicações não comerciais. Nesse projeto todos os gráficos foram desenvolvidos usando a mesma por ser compatível com todos os navegadores modernos, incluindo o iPhone / iPad e Internet Explorer a partir da versão 6, não necessitando de plugins Java ou Flash no lado do cliente e ainda ser muito bem documentada.

Capítulo 4

Arquitetura Interna do Software

A vantagem do GWT é que toda a aplicação é escrita em Java durante a implantação, sendo que as partes que vão rodar no cliente são traduzidas para JavaScript garantindo a execução nos navegadores disponíveis.

A abordagem do GWT, entretanto, tem uma desvantagem, pois como o código Java é transformado em código JavaScript para executar no navegador, apenas algumas classes do Java estão disponíveis para uso (em sua maioria classes dos pacotes "java.lang" e "java.util") e funcionalidades avançadas da linguagem (como acesso a bancos de dados) têm que ser feitas utilizando o suporte a invocação remota de métodos do GWT, que faz a ponte entre a aplicação no navegador e o código Java no servidor.

Por essa razão, a aplicação principal de interação com o usuário, pode ser dividida em duas partes:

- Cliente - Responsável pela interação direta do cliente através da interface. Responsável por capturar as configurações de loja/vitrine e período de interesse do cliente e por mostrar os resultados retornados pelo servidor.

- Servidor - Responsável pela interação na nuvem e pelo processamento dos dados consultados. O servidor recebe do cliente o período e a loja/vitrine de interesse, consulta os dados na nuvem, agrupa baseado na forma de visualização temporal escolhida pelo cliente e envia os dados para a interface do cliente.

4.1 Cliente

O GWT funciona através de módulos, que podem ser adicionados a qualquer página HTML, bastando incluir a referência para o javascript dentro do módulo onde serão criados 2 pacotes principais: o pacote client e o pacote server.

Dentro do pacote client ficam os EntryPoints, que são pontos de entrada do módulo onde está definido o método onModuleLoad(), executado quando o mesmo é carregado na página.

Logo no inicio do projeto, pouco se conhecia sobre essa estrutura e na versão 01 do sistema todas as telas e componentes eram adicionados ao mesmo módulo, gerando uma página pesada de carregar, inicializando componentes e variáveis nem sempre usados e com uma estrutura de difícil manutenção, chegando o arquivo ter 4436 linhas de código.

Visando simplicidade e modularidade da interface, o sistema foi todo desenvolvido utilizando componentes básicos do GWT puro e componentes específicos foram criados para a aplicação, facilitando a reutilização dos mesmos e garantindo uma padronização das telas.

A tela principal possui apenas um menu de opções, que são carregadas a partir das permissões associadas ao usuário durante o login. Cada permissão desse login gera uma opção no menu associada a uma widget, uma janela totalmente independente com um conjunto de componentes adequados a sua função. Essas widgets são criadas e recriadas no lado do cliente e os dados necessários na mesma são invocados através de servlets ao servidor.

A vantagem de dividir as funcionalidades, ou telas, do sistema em widgets associadas a permissões do usuário é permitir criar perfis de usuários, ou seja, denominar responsabilidades específicas aos usuários: administração, cadastros, monitoramentos, análise de dados, auditoria e até combinações das mesmas. As lojas que podem ser visualizadas por um usuário também são um tipo de permissão associada ao perfil. Dessa forma, é possivel ter dois gerentes, na mesma empresa, responsáveis por lojas diferentes estimulando competições entre equipes, análise de gestão e estratégia de vendas.

Na Figura 4.1 podemos ver todo o menu de cadastros do sistema todas as telas de cadastros possuem a mesma aparência, para manter um padrão e melhorar

Figura 4.1: Menus de cadastro e tela de cadastro de câmeras

o intendimento e a usabilidade do sistema.

4.2 Servidor

Como o GWT possui uma limitação do uso de bibliotecas Java no lado do cliente, uma solução simples, é a invocação remota de serviços implementados no servidor (através de um Servlet) que pode então se utilizar de toda a expressividade e APIs do Java para retornar os dados processados para o cliente. Para criar um serviço que vai ser invocado, precisamos inicialmente definir a interface do serviço. Essa interface deve obrigatoriamente estender a interface de marcação com.google.gwt.user.client.rpc.RemoteService e os seus métodos são automaticamente definidos como métodos "invocáveis" remotamente.

O GWT consegue enviar diretamente objetos comuns do Java, como todos os tipos primitivos, Strings e Dates (junto com suas representações como arrays), mas os tipos definidos pelo usuário precisam de um tratamento especial. As classes definidas pelo usuário que necessitem ser enviadas por invocações remotas devem implementar a interface de marcação com.google.gwt.user.client.rpc.IsSerializable e ter como propriedades apenas outros objetos que também implementem essa interface ou os objetos comuns do Java citados anteriormente. Como o banco utilizado foi o Bigtable, que usa como unidade de comunicação com o banco denominadas

como “entity” [1], foram utilizadas DTO’s (Data Transfer Objects) para transferir dados através de requisições ao servidor.

Na Figura 4.2, temos uma visão percentual de todas entidades do sistema e o percentual em tipos primitivos (string, int, boolean).

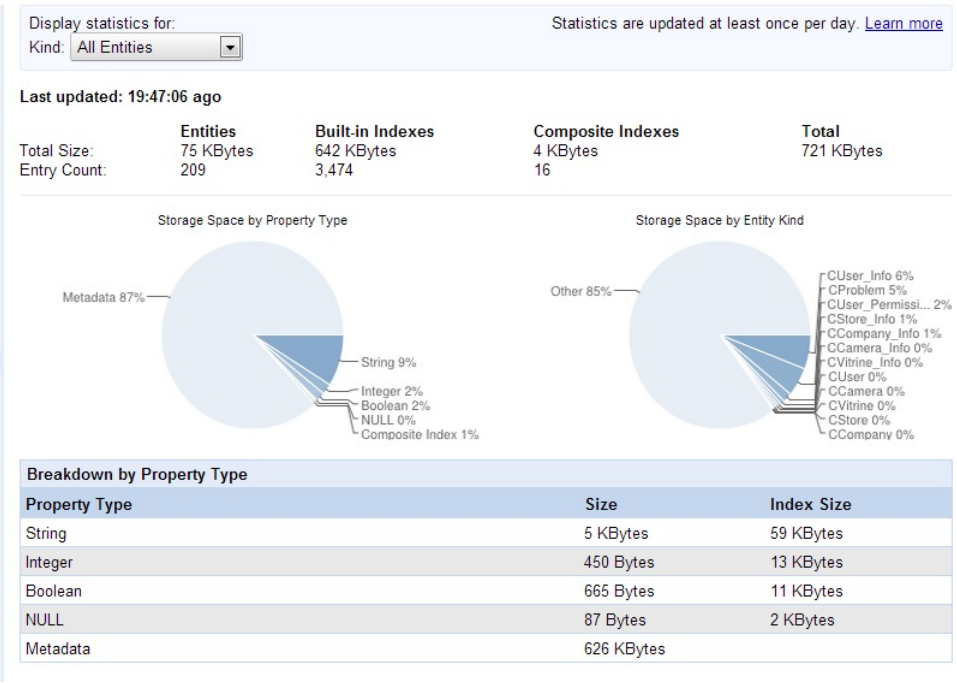

Figura 4.2: Entitades e espaço ocupado na aplicação

4.3 Comunicação Cliente-Servidor

Com a interface base do serviço e o objeto que ele retorna definidos corretamente, nós precisamos definir uma “interface assíncrona” para a execução do serviço no cliente. A definição da interface assíncrona é necessária porque a invocação ao serviço utilizando AJAX acontece de forma assíncrona, para o cliente, a aplicação não para de executar, a chamada ao serviço acontece em background, então é necessário definir a interface assíncrona que vai ser a interface real do serviço no cliente. A interface de serviço assíncrona deve estar definida no mesmo pacote do serviço e a url do sevlet deve ser especificada dentro do arquivo web.xml, na pasta do projeto.

A interface assíncrona do serviço tem as declarações de métodos praticamente

[1]Como será detalhado na seção 4.4

Cliente	Servidor	Interface de Serviço Assíncrona
MyServlet	MyServletImpl	MyServletAsync

Tabela 4.1: Comunicação cliente-servidor através de servlets

iguais às da interface do serviço definida anteriormente, mas os métodos de serviço agora devem retornar "void" e o último parâmetro deve ser um objeto do tipo com.google.gwt.user.client.rpc.AsyncCallback, que é o objeto que vai funcionar como um "listener" do evento de execução do serviço.

AsyncCallback é uma interface que define dois métodos, "onSucess()" que é chamado quando o método executa corretamente e tem como parâmetro o resultado da invocação do serviço (por isso o método da interface assíncrona retorna "void") e "onFailure()" que é chamado quando a chamada do serviço não acontece de forma correta e recebe como parâmetro um objeto do tipo Throwable com informações sobre o erro acontecido.

Com as interfaces do serviço definidas no lado do cliente, vamos implementá-lo no servidor criando um objeto que estenda com.google.gwt.user.server.rpc.RemoteServiceServlet e implemente a interface real do serviço. Esse objeto deve ficar dentro dos pacotes "server" no mesmo nível do pacote "client" e "public" da aplicação, para que o GWT saiba que esse objeto não deve ser transformado em JavaScript, pois ele pode vir a fazer uso de várias APIs do Java que não estão disponíveis em JavaScript.

Já no servidor podemos usar e abusar das API's Java para agrupar, comparar e formatar os dados consultados, gerando um DTO serializável para enviar à interface.

A aplicação executa normalmente, acessando dados no servidor e sem fazer refresh na página inteira, atualizando apenas os campos que precisam ser atualizados, diminuindo o consumo de banda e de processamento no servidor, pois apenas os dados que são necessários são requisitados.

4.4 Comunicação Servidor-Banco de Dados

Como comentado no capítulo 3, o servidor utilizado foi o GAE situado na "nuvem" do Google. Isso permitiu à ICE Interactive, que ainda é uma "Start Up",

economizar com manutenção, instalação e configuração de um servidor dedicado. Daí concluiu-se que o projeto deveria armazenar e transmitir o mínimo de dados, economizando banda e evitando repetição de informações no banco.

Existem várias frameworks de comunicação com o Bigtable, dentre eles o Objectify e Twig ; ambos lembram a implementação Java tradicional utilizando JDO (Objetos de dados Java) ou JPA (Java Persistence API). No período de especificação do sistema esses foram descartados, escolhendo-se usar a Low-level API oficial do GAE, implementando uma estrutura em folhas. Nessa estrutura, as informações que caracterizam um objeto ficam numa "Classe" na ponta da árvore evitando a transferência de dados básico do tipo nome e descrição a cada requisição, a longo prazo foi um modelo bem criticado já que nas telas de administração essas informações devem estar visíveis em todos os níveis da árvore.

Na Figura 4.3 onde as "Classes-Info", que possuem os detalhes da "Classe" pai. Na conclusão deste projeto será apresentado um planejament para melhorar esta estrutura.

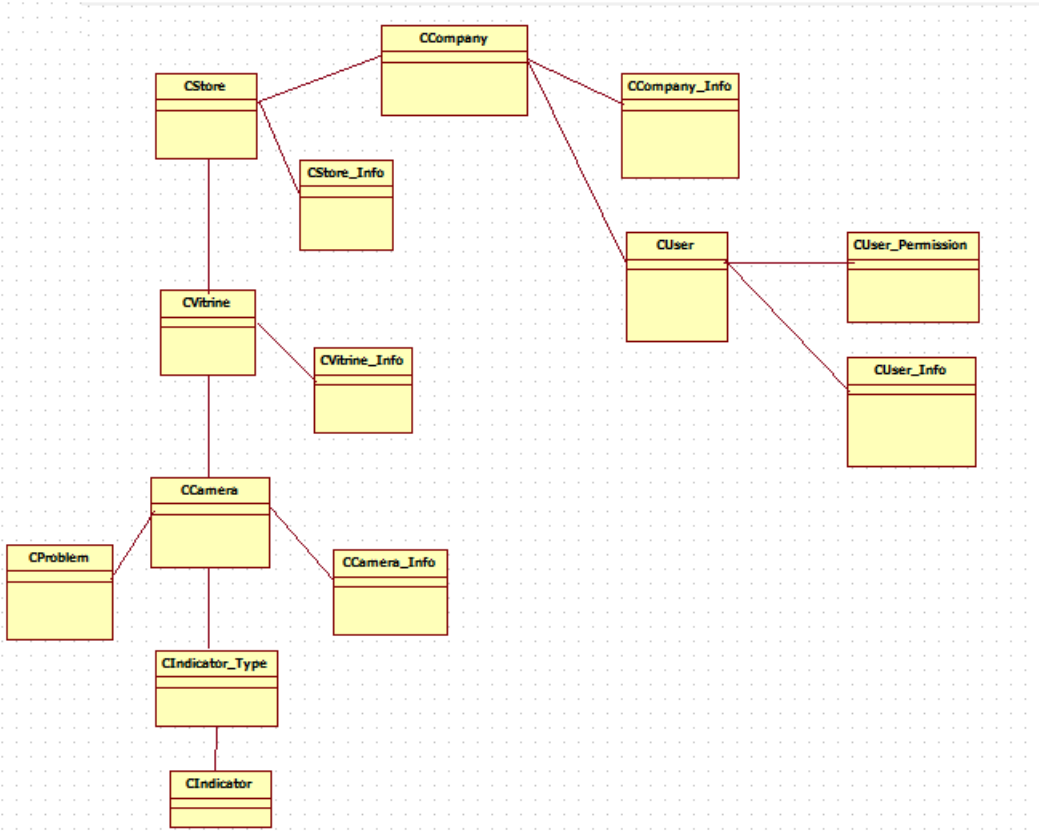

Figura 4.3: Modelo Entidade-Classes

A grande dificuldade de utilizar a Low-level API é que não existem classes de persistências no banco. O Bigtable é uma estrutura não relacional (NoSQL) onde não precisamos nos preocupar com a criação de tabelas. O GAE executa e armazena dados sobre uma "entidade" que possui propriedades variáveis, essas propriedades seriam as colunas no modelo SQL.

Com a possibilidade de o modelo de dados escolhido ser modificado ao longo do projeto (o que tornou-se uma solução viável) e para garantir que as "entidades" repesentando uma determinada classe de informação teria campos fixos, modelou-se o sistema na forma de uma Classes Java padrão. Ou seja, criou-se uma classe Java que possuía como propriedade privada uma "entidade", e seu construtor padrão fixo e seus Get's e Set's limitando acessos às propriedades da "entidade".

Dessa forma, a "entidade" seria visível como uma "classe" para o desenvolvedor, ou um bloco que poderia ser mutável em uma implementação futura. Apesar de todo esse cuidado na melhor forma da criação das "classes de dados", as consultas não eram tão simples de generalizar, mas o código foi isolado de forma a facilitar a sua localização e alteração.

No código a seguir temos um exemplo de como foi implementada essa "classe-entidade":

```
1  public class CCamera_Info  {
2      private Entity entity;
       public CCamera_Info(Key keyParent, String name,
          String descricao){
4          entity = new Entity("CCamera_Info",keyParent);
           entity.setProperty("name", name);
6          entity.setProperty("descricao", descricao);
       }
8  }

10 DatastoreService ds = DatastoreServiceFactory.
     getDatastoreService();
   CCamera_Info cam_info = new CCamera_Info(KeyParent,
     name,descricao);
```

```
12 ds.put(cam_info.getEntity());
```

Consultar “entidades” do banco de dados pode ser análogamente muito simples:

```
1
2 DatastoreService ds = DatastoreServiceFactory.
     getDatastoreService();
  Query camerajQuery = new Query("CCamera_Info");
4 PreparedQuery pq = ds.prepare(camerajQuery);
  List<Entity> results = pq.asList(FetchOptions.Builder.
     withDefaults());
6 if (!results.isEmpty())
  {
8  for (Iterator iterator = results.iterator(); iterator
      .hasNext();) {
    Entity entity = (Entity) iterator.next();
10  CCamera_Info cam_info = new CCamera_Info(entity.
       getKey(),(String)entity.getProperty("name"),(
       String)entity.getProperty("descricao"));
   }
12 }
```

No exemplo anterior, por simplicidade não foram colocados filtros às queries.

4.5 Dashboard - Highcharts

O dashboard foi construído a partir do componente “Chart” da highcharts [47], gerando um componente proprio “ModuloChart” já configurado, através de um JavaScript para mostrar resultados de acordo com a periodicidade escolhida pelo usuário (hora, dia, mês e ano), módulo de visualização (gráfico de barras ou linha) e a quantidade de séries visiveis que dependem da forma de agrupamento dos dados (gênero, faixa etária e total).

Tooltips, labels, quantidade de eixos disponíveis e visualização estatísticas através de pizza ou percentuais também estão modularizadas nesse componente, per-

mitindo a fácil duplicação do mesmo para fazer análises comparativas entre períodos e a geração de uma tela de auditoria, para conferência dos dados comparando o gráfico e tabela de dados em um grid.

A consulta dos dados é feita uma única vez, desde que o período não seja alterado. As aglutinações de dados são feitas na própria interface do cliente através de complexos algoritmos de que os ordena/agrupa por hora, dia, mês ou ano as séries de dados já divididas por gênero, faixa etária ou total. Essa complexidade esta associada a limitação da API Java no cliente, como já mencionado, algoritmos de auto grau de dificuldade sem poder utilizar a vantagem da API.

A atual situação de aplicação piloto no cliente permite monitorar o bom funcionamento desses algoritmos para massa de dados.

4.6 Transmissão de Dados para a Nuvem

Como descrito no capítulo 3, o sandbox só permite a comunicação de dados através de uma url e não seria possível criar uma outra aplicação que comunica-se com o banco associado à aplicação. As imagens capturadas pela câmera, passam por um sistema proprietário de processamento de imagens, que extrai as informações de gênero, faixa-etária, tempo de visualização e outras não utilizadas no nosso sistema e às salva num formato "csv". Num período programado, um script é executado pelo linux, criando uma string com essas informações e enviando para uma url na qual um HttpServlet está mapeado.

No servlet, essa string é validada e suas informações são salvas na "entidade" CIndicator que tem como "entidade pai" um CIndicator-Type, aonde estão localizadas as informações de gênero e faixa-etária.

4.7 Monitoramento

Como será detalhado no capítulo 5, existem duas formas de monitoramento, apresentadas a seguir:

O primeiro monitoramento é executado localmente na máquina do cliente, na loja, verificando se as imagens de vídeo estão sendo capturadas corretamente. A câmera poderia ser removida, ter sua posição modificada de forma que as imagens

ficassem obstruídas, por falta de energia ou problema de hardware ela estar desligada. Essa aplicação é um script programado no linux para rodar periodicamente, que verifica a resposta da conexão USB da porta onde a câmera esta conectada, cria e envia uma string com essa informação e a hora para uma url na qual um HttpServlet esta mapeado e aguardando dados.

A string recebida pela aplicação é tratada no servlet e sua informação é salva na "entidade" denominada CAlive.

O segundo monitoramento é executado na nuvem através de um cron Job. O App Engine Cron Service permite que você configure tarefas regulares que operam em horários definidos ou intervalos regulares. Estas tarefas são comumente conhecidos como cron jobs. No nosso caso, mapeamos url desse cron job para um HttpServlet cuja função é verificar a string do sinal de CAlive, gerar uma "entidade" denominada Problem, se necessário e enviar um email aos responsáveis pela manutenção. Essa "entidade" Problem, pode ser visualizada através de uma interface gráfica de monitoramento pelo usuário, Figura 5.3.

```
<?xml version="1.0" encoding="UTF-8"?>
<cronentries>
  <cron>
    <url>/Monitoramento</url>
    <description>Monitoramento Alive</description>
    <schedule>every 1 hours from 10:10 to 23:10</
       schedule>
  </cron>
</cronentries>
```

Na figura 4.4, temos uma análise das requisões por segundo executadas pelos dois monitoramentos, comparados a uma adição de uma câmera (salvar uma entidade). Em laranja apenas o sinal de CAlive enviado de hora-em-hora, em verde o sinal de CAlive e um cron Job no mesmo período que verifica o o conteúdo do CAlive gerado e possivelmente envia um email se houver uma situação anormal e em amarelo a adição de uma câmera. Como pode ser visto na imagem, a execução desses processos programados exige muito pouco da aplicação.

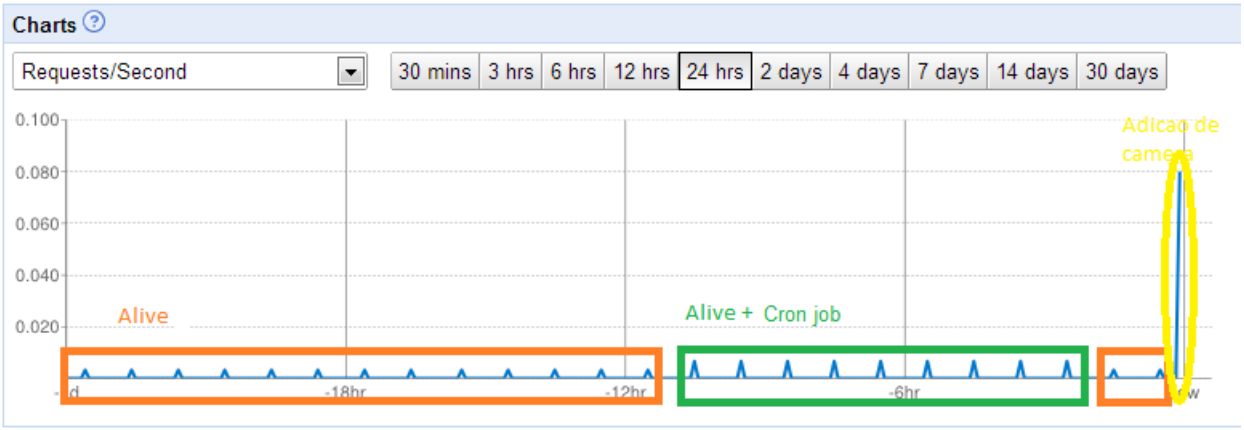

Figura 4.4: Requisições do monitoramento

Capítulo 5

Funcionamento do Sistema

Neste capítulo serão apresentados os aspectos práticos do projeto, ou seja, como ele funciona e como os usuários interagem com ele.

Além das funcionalidades propriamente ditas, também serão descritas as necessidades físicas (para as instalações) e os perfis dos diferentes usuários, com suas diferentes necessidades. Será discutido também ao final do capítulo, através de exemplos fictícios, como os dados exibidos pela ferramenta podem ser utilizados por agentes econômicos na gestão dos seus negócios.

Figura 5.1: Tela inicial da aplicação final

5.1 Instalação do Sistema

A primeira etapa para colocar o sistema em funcionamento é a instalação física dos equipamentos necessários, a saber:

- Câmera - o sistema funciona com câmeras comuns USB (webcams); dependendo da posição e/ou altura há a necessidade de uma câmera com resoluções mais altas. Neste projeto estão sendo utilizadas as webcams Logitech QuickCam Pro 9000;

- PC - o sistema de processamento de video quase não consome memória ou processador. Minicomputadores do tipo NUC (Next Unit of Computer) da Intel foram utilizados por serem pequenos e portáteis (cerca de 10cm x 10cm). Suas dimensões e a ausência de cooler tornam as instalações (físicas) extremamente simples, além de serem robustos a ambientes hostis (como o forro do teto das lojas).

- Infraestrutura de rede (para acesso à internet) - para que os dados processados possam ser enviados para a nuvem e que o sistema de monitoramento de câmera funcione corretamente, modens do tipo 3G são utilizados conectados no NUC.

A Figura 5.2 ilustra o minicomputador e a câmera utilizados;

Figura 5.2: Logitech QuickCam Pro 9000 e Next Unit of Computer respectivamente

A câmera é a responsável pela captura das imagens das faces das pessoas. Essas imagens são processadas para a extração das informações de gênero, faixa etária, tempo de permanência, etc. Por sua vez, estas informações são compiladas e enviadas para o banco de dados na nuvem, através da internet.

O processamento das imagens e geração dos dados é feita por um aplicativo desenvolvido por um parceiro da ICE, e a cada vez que uma pessoa é detectada, salva em um arquivo, no formato csv, as informações relativas a essa pessoa.

São elas: ID, instante em que foi detectada, tempo de permanência no campo de visão da câmera, Gênero , Faixa etária , Número de olhares.

Essas informações são agrupadas de forma a facilitar o seu envio para o Banco de Dados. Isto é feito por um aplicativo desenvolvido pela ICE, que é configurado para enviá-las periodicamente (esta periodicidade varia de acordo com as necessidades do cliente).

Ambos os aplicativos são multiplataforma, rodando tanto em Windows como em Linux. O sistema operacional selecionado, por questões financeiras, foi o segundo, que além da economia gerada também permite uma fácil configuração e agendamentos tarefas usando scripts de linha de comando.

Apesar de serem poucos os itens necessários, a instalação é um ponto extremamente crítico, já que é necessário posicionar as câmeras de modo que elas sejam capazes de capturar as faces das pessoas, mas ao mesmo tempo não sejam facilmente detectáveis nem tenham impactos negativos na decoração. No caso específico de vitrines, quando o objetivo é medir o resultado do impacto causado por elas, as câmeras devem estar posicionadas o mais discretamente possível, para que não tenham influência nesse resultado. A estratégia adotada atualmente pela empresa é a utilização de cúpulas semiesféricas, amplamente utilizadas em câmeras de segurança, e ás quais as pessoas já estão acostumadas (ou seja, não chamam atenção).

5.2 Envio de Dados

Como já foi apresentado no Capítulo 4, para o armazenamento dos dados são utilizadas entidades chamadas CIndicators, que basicamente contém o tipo de informação (contagem ou tempo), o valor (quantas pessoas, no caso da contagem) e a janela temporal a que se referem, tudo isso relativo a uma combinação de gênero e faixa etária.

Isso foi estruturado desta forma para que não houvesse a necessidade de enviar todos os dados de cada face detectada individualmente, sobrecarregando o

processamento na nuvem.

Com isso, o que é realmente enviado é uma string cuja estrutura básica é no seguinte formato:

72&camid=11&observation=ALIVE&TimeStamp=2013-06-05 16:40:32&ErrMessage=ERRO

Essa string é devidamente encriptada e enviada através de um request de um HttpServlet pelo método post, para evitar que os dados possam ser acessados externamente. Na nuvem, esses dados são decriptados, e as informações dessa string são extraídas e inseridas no banco através dos CIndicators.

O agrupamento dos dados para a criação dessas string é realizado por um aplicativo que roda localmente na máquina do cliente, e é executado periodicamente dependendo de quanto em quanto tempo o cliente deseja ter os dados disponíveis para o seu acesso.

5.3 Monitoramento do Sistema

Para garantir que o sistema esta funcionando corretamente dois aplicativos são utilizados para garantir o funcionamento das câmeras.

5.3.1 Monitoramento Local

No computador instalado no cliente há um aplicativo responsável pelo monitoramento. Este aplicativo é de extrema importância, pois é através dele que são identificados problemas que possam impedir o correto envio dos dados para a nuvem. Este aplicativo também é executado periodicamente, enviando para o sistema na nuvem um sinal de Alive, contento duas informações: o instante em que foi enviado e uma mensagem que indica se está tudo OK ou se foi encontrado um erro (e que erro foi esse). Atualmente os erros mapeados nesse sistema são relacionados ao correto funcionamento do aplicativo de processamento das imagens e da câmera.

5.3.2 Monitoramento na Nuvem

Na nuvem há um outro sistema responsável por varrer o banco á procura desses sinais de Alive para cada câmera. Caso sejam encontrados, sua mensagem é analisada: se for "OK" então não há erro; do contrário, é gerado um outro sinal, chamado Problem, que é enviado para os responsáveis pelo monitoramento do sistema, para que sejam notificados do problema que foi detectado. Além disso, esse sistema também verifica se o Alive está atualizado (seu instante de envio está dentro do intervalo entre duas verificações). Se não estiver, é gerado um Problem indicando que o Alive está desatualizado (o que pode indicar que o computador no cliente não está conectado á internet, ou que ele travou, ou ainda que está desligado.

Figura 5.3: Visualização do monitoramento automático na nuvem

Além de notificar os responsáveis pelo monitoramento e manutenção dos sistema, este sistema exibe, em uma página exclusiva para o monitoramento, as câmeras que apresentam problemas, para que a busca pela solução possa ser acompanhada.

Na Figura 5.3 é mostrado um exemplo do sistema de monitoramento na nuvem;

Posteriormente deverão ser acrescentadas novas informações a esse sistema de monitoramento, incluindo dados de funcionamento do próprio PC, como consumo de memória, temperatura, etc., para a realização de manutenção preventiva dos equipamentos ou mesmo sua substituição, antes que algum problema efetivamente

possa ocorrer.

5.4 Definição dos usuários e permissões

Figura 5.4: Usuários

O sistema foi planejado para a existência de três perfis de usuários, que têm necessidades e interesses distintos, e por essa razão têm permissões de acesso diferentes aos tipos de informação. Devido á instrutura interna do sistema, classe de permissões esta associada á classe de usuários, outras configurações de permissão podem ser designadas aos usuários em questão de acordo com a necessidade do cliente.

A seguir segue a definição das classes padrão de usuários :

1. Super Usuário - Usuário responsável por criar as contas dos demais usuários e das demais entidades presentes no sistema. Este usuário é interno da empresa ICE, já que a introdução de novos usuários no sistema está diretamente ligada ás vendas do sistema. Além de criar as contas para os demais usuários, o SuperUsuário também cria as contas dos clientes (PJ); a cada nova venda e instalação do sistema, cria-se uma nova estrutura no Banco de Dados que contém a Companhia (o cliente em si), a Loja (local onde o sistema foi instalado), a Vitrine (que neste caso não se restringe apenas a uma vitrine, mas a qualquer ponto de interesse que o cliente deseje analisar) e as câmeras (já que um ponto de interesse pode ser analisado através de mais de uma câmera).

2. Gerente - Este nada mais é do que o real interessado por parte do cliente na utilização do sistema para a análise das métricas. Sua principal atividade é

acessar o Dashboard de cada uma de suas câmeras para analisar as métricas disponíveis e em cima delas tomar decisões sobre a estratégia adotada. No caso de uma vitrine de loja essa estratégia pode ser a decoração, iluminação ou qualquer fato que desperte a atenção das pessoas que circulam á frente da loja, ou mesmo uma mudança na sua estratégia ou equipe de vendas.

3. Monitor - Sua função é basicamente uma só: garantir que o sistema está sempre funcionando, buscando solucionar todos os problemas de hardware ou software que possam atrapalhar o sincronismo do sistema. Ele também pode ser um usuário interno da ICE, que tem acesso exclusivamente á tela de monitoramento, onde são listadas as câmeras que apresentam problemas, onde também ele pode indicar que a solução já foi alcançada. É este usuário que é notificado pelo sistema de monitoramento apresentado na seção anterior.

Existe também mais uma função do "Super Usuário": associar as entidades "Gerente" e "Loja". Um determinado Gerente pode ser responsável por analisar todas as lojas de uma empresa, ou pode dividir essa tarefa com outros. Ou seja, há uma grande flexibilidade com relação a como os clientes irão trabalhar com o acesso ás informações geradas pelo Horus Analytics.

Em resumo, o usuário de principal interesse aos clientes da ICE Interactive é o Gerente, pois ele é o responsável pela análise das métricas geradas pelas câmeras instaladas e será responsável pela tomada de desisão sobre a estratégia.

5.5 Definição dos dados e das métricas

Como foi apresentado na seção anterior, é o Gerente quem tem acesso ao dashboard e ás informações nele apresentadas (Figura 5.5). Vamos então entender um pouco mais sobre como elas podem ser utilizadas no dia-a-dia do cliente.

Atualmente o sistema disponibiliza as seguintes indicadores aos clientes:

- Número de pessoas que passaram em frente ao ponto de interesse, independente de terem olhado ou não. Essa medida é conhecida no mercado como OTS (Opportunity to see). Ou seja, são as pessoas que têm a oportunidade de visualizar o que quer que os clientes queiram lhes mostrar;

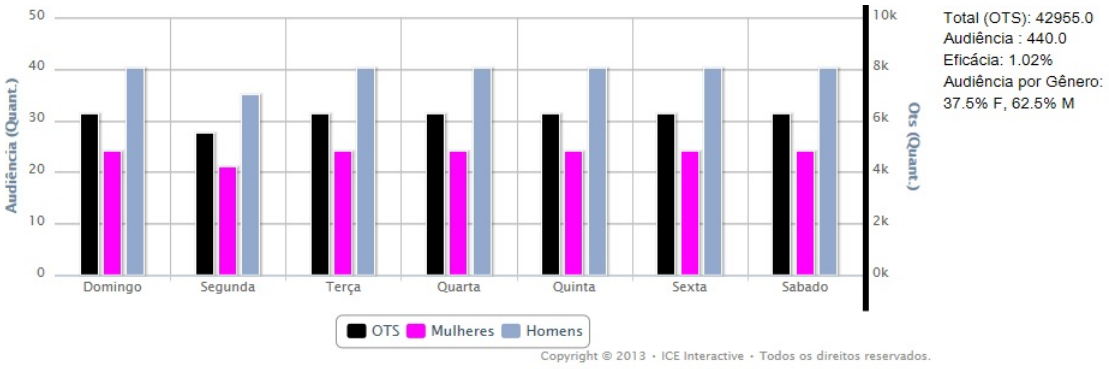

Figura 5.5: Audiência classificada por gênero e OTS

- Número de pessoas que efetivamente olham para a vitrine (ou qualquer outra mídia). Além do valor numérico, também é feita a classificação dessas pessoas de acordo com o gênero e a faixa etária. No caso do OTS não há essa classificação porque ela depende de a face poder ser identificada pela câmera para que essas informações possam ser obtidas, o que nem sempre ocorre quando as pessoas passam e não olham para o ponto de interesse;
- Engajamento, que é o tempo médio que as pessoas olham;
- Número de pessoas que param. é uma medida importante, pois as pessoas que param em frente a uma vitrine demonstram um comportamento diferenciado em relação ás que apenas passam olhando por ela;
- Engajamento das pessoas que pararam: tempo médio em que essas pessoas ficaram paradas olhando para o local de interesse;
- Número de pessoas que entraram na loja, chamado OTB (Opportunity to buy). Essas pessoas também são classificadas por gênero e faixa etária.

Essas medidas, por si só, já podem fornecer uma grande variedade de informações aos clientes. A mais importante delas é a visualização direta das taxas de conversão do seu ponto de interesse. As taxas de conversão são as razões entre as pessoas que passam de um estágio para outro. Usando como exemplo uma vitrine de uma loja, podemos Definir os estágios como “Passou em frente á loja”, “Olhou para a vitrine”, “Parou em frente á vitrine” e “Entrou na loja”.

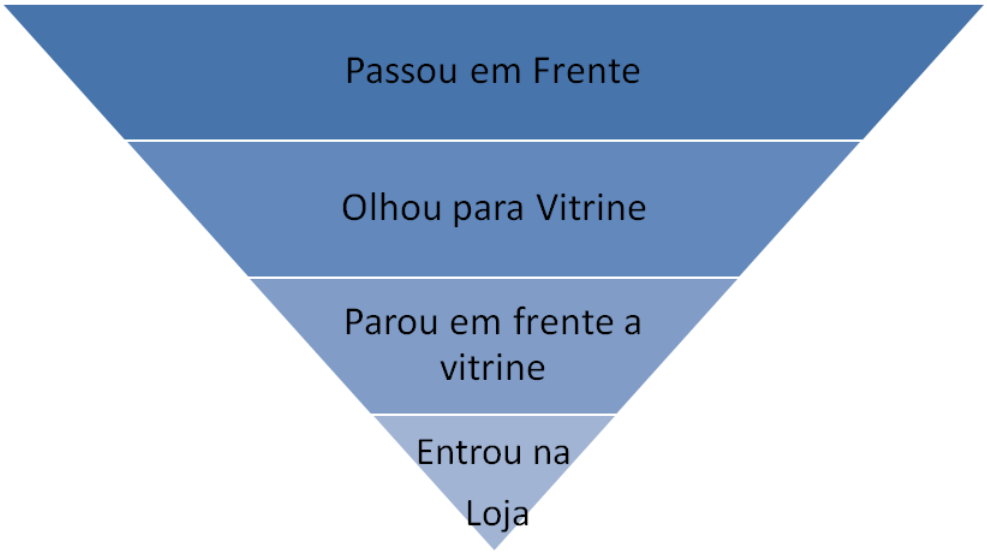

Figura 5.6: Funil de análises métricas

Através da análise desse funil são criadas métricas, que representam a taxa de conversão de cada um dos estágios. A Atratividade, por exemplo, é a razão entre o número de pessoas que passam pela vitrine e o número de pessoas que olham; já a efetividade é a razão entre as que olham e as que entram na loja. Através da análise das diferentes combinações entre as medidas e as taxas de conversão, um cliente pode tomar decisões estratégicas para aumentar essas taxas de conversão e, consequentemente, suas vendas.

5.6 Utilização do Analytics

Nesta seção o foco está apenas na utilização do sistema por parte do "Gerente", pois como já foi concluído na Seção 5.4, é ele o interessado na navegação pelo dashboard.

Acessando o site do Horus Analytics, a primeira tela que aparece para o usuário é a de login (Figura 5.1). Após efetuar o login, o Gerente é encaminhado para a sua tela inicial, onde estão listadas as lojas às quais ele está vinculado e onde ele pode escolher qual dentre delas irá visualizar (Figura 5.7).

Clicando sobre a vitrine desejada, o Gerente é encaminhado para o seu dashboard. Já são previamente carregados os dados referentes a data de análise, período default.

Figura 5.7: Visualização de vitrines nas lojas

O item (1) realçado na Figura 5.8 mostra o menu com as opções de métricas que o Gerente pode visualizar (Audiência, Engajamento, etc.). O item (2) mostra algumas informações e métricas adicionais relativas aos dados exibidos no gráfico (como por exemplo a eficiência da vitrine).

Em (3) são destacados os botões que permitem ao Gerente escolher de que forma deseja agrupar os dados para visualização (por hora, dia, dia da semana ou mês). (4) mostra como pode ser alterada a janela temporal de análise. O item (5) mostra como o Gerente pode alternar entre os dashboards de diferentes câmeras. E

Figura 5.8: Funcionalidades da Tela de Dashboard

por fim o item (6) mostra três botões: o primeiro para alterar o modo do gráfico para linhas, o segundo para barras, e o terceiro para habilitar o modo de comparação (exibido na Figura 2.2). Navegando por todas essas opções, o Gerente pode obter

diversos tipos de combinações entre as métricas, para analisar o desempenho de suas vitrines e utilizar essas informações como auxílio na tomada de decisões extra.

5.7 Análise dos dados

Nesta seção vamos mostrar, através de exemplos fictícios, de que forma as informações apresentadas podem ser aplicadas ao dia-a-dia do Gerente, tornando-se uma ferramenta de auxílio à decisão.

Os exemplos são de certa forma simplistas, mas servem muito bem para ilustrar a aplicação prática da ferramenta.

5.7.1 Exemplo 1

O Gerente A deseja analisar o perfil da audiência de sua vitrine, para saber se está alcançando o público-alvo desejado. Para isso, contrata o sistema Horus, e instala uma câmera em sua vitrine. Sua loja é focada na venda de roupas para mulheres.

Após a análise contínua por uma semana, o Gerente A percebeu que ele a audiência de sua vitrine é em sua maioria formada por homens jovens, diferente do que ele imaginava (que seria de mulheres jovens e adultas). Conversando com seus vendedores, ele descobre que seus clientes principais são homens que compram roupas para presentear suas namoradas ou esposas.

Com isso, o Gerente A resolve modificar a sua estratégia de vendas, orientando os seus vendedores a lidar melhor com homens jovens ao invés de apenas mulheres. Cruzando os dados da audiência de sua vitrine com o faturamento do caixa nas semanas seguintes, ele percebe que a razão entre as pessoas que entram na sua loja e as que efetuam uma compra aumentou.

Ou seja, partindo de uma análise do perfil das pessoas que têm a sua atenção chamada pela vitrine, o Gerente foi capaz de aumentar o faturamento de sua loja sem precisar de muitas modificações na sua estrutura.

5.7.2 Exemplo 2

O Gerente B possui duas lojas em shoppings distintos. Há uma grande diferença no faturamento de ambas, que pode ser causado por diversos fatores, desde a própria localização da loja até a eficiência de sua equipe de vendas.

Para analisar o desempenho de ambas, ele instala o Sistema Horus em cada uma delas, interessado em analisar o seu funil de conversão e verificar a origem dessa diferença de faturamento.

Após um mês de análise, ele verificou que o funil de entrada das duas lojas era bastante semelhante, ou seja, o número de pessoas que passava por cada um dos estágios (passar, olhar, parar, entrar) era praticamente o mesmo. Com isso, ele pôde descartar fatores como localização da loja no shopping, diferença na quantidade de frequentadores, etc. Sabendo também que o perfil socioeconômico dos frequentadores de ambos os shoppings não difere, ele concluiu que o problema é interno à loja. Sua ação foi, então, renovar a equipe de vendas, o que nos meses seguintes fez com que o faturamento dessa loja se equiparasse ao da outra.

5.7.3 Conclusões

Como já foi dito, esses exemplos são extremamente simples; eles deixam diversas variáveis de fora, mas por outro lado ilustram muito bem de que forma as informações proporcionadas pelo Horus podem ser combinadas com outras informações que o cliente possua (ou até mesmo outras que ele seja indicado a buscar), para refinar as opções que ele possui para aumentar o seu faturamento.

É importante observar que o Horus, sozinho, não funciona como uma ferramenta de estratégia, mas que é mais uma das diversas fontes de informação que um comerciante deve possuir para tornar-se mais eficiente e mais lucrativo. Do mesmo modo, pouco se pode concluir usando como base apenas a informação de faturamento de uma loja, sem levar outros fatores em consideração.

A ideia do Horus é poder agregar diversas outras fontes de informação para tornar-se uma ferramenta de auxílio à decisão cada vez mais completa. Outras métricas podem ser criadas a partir das medidas realizadas atualmente, também pode ser feita uma atumatização no cruzamento com dados externos (como faturamento, CRM, controle de estoque, gerenciamento de equipe, etc.) de forma que o

cliente não precise ele mesmo buscar outras fontes de informação e cruzá-las manualmente.

Capítulo 6

Resultados

A fim de avaliar o desempenho da aplicação algumas etapas de testes foram estabelecidas:

1. Velocidade de resposta para uma quantidade massiva de dados
2. Sincronismo entre os sistemas de monitoramento
3. Compatibilidade com navegadores e dispositivos (computadores, tablets e celulares)
4. Clareza de usabilidade (interface homem-máquina)
5. Resposta do cliente

Para validação das etapas citadas foram gerados dados gerados artificialmente através de macros no excel, dados capturados no Restaurante Central da UFRJ, dados capturados na Incubadora de Empresas da COPPE e dados coletados da aplicação piloto que está sendo utilizada em um shoping do Rio de Janeiro.

Testes com o aplicativo de processamento de imagens também foram feitos, para verificar o seu desempenho e o do PC com o aumento do número de câmeras conectadas. Atualmente o limite de câmeras suportadas pelo software de processamento é de 8, limitado pelo próprio software. Porém estes não influenciam no sistema desenvolvido na monografia.

6.1 Velocidade de resposta para uma quantidade massiva de dados

Esse teste foi de grande importância para definir as ferramentas de desenvolvimento do sistema. Apesar da confiança na empresa Google e sua tradição em sistema de quantidade massiva de dados, foi importante analisar o desenpenho da aplicação desenvolvida. Foram gerados cerca de 6 meses de dados para 2 câmeras enviando os dados de hora-em-hora 12 horas por dia : 6 x 30 x 12 x 8 x 2 = 34560 CIndicators.

Então foi avaliado o tempo de resposta web dos filtros por períodos distintos: 5 dias, 30 dias, 3 meses. A medida de tempo não foi possível de ser coletada mais foi considerada aceitavel pela equipe da ICE Interactive que acompanhou o processo.

Essa velocidade também está relacionada à velocidade da conexão web.

6.2 Sincronismo entre os sistemas de monitoramento

Foi programado ao sistema de monitoramento local para que enviasse de hora-em-hora resultados referentes ao funcionamento da câmera de forma simulada. Algumas vezes mandando resultados de funcionamento normal e em outras diversos erros.

O sistema de monitoramento na nuvem, também foi programado de hora-em-hora, e analisou esses resultados e quando não eram positivos, enviou emails para os responáveis e criou um "CProblem" visualizado na tela de monitoramento (Figura 5.3), sinalizando a origem do sinal (Loja/Vitrine/Câmera), a data/hora e a mensagem enviada pelo monitoramento local.

Esse sistema também apresentou sincronismo e confiança na etapa final de testes e na aplicação piloto.

6.3 Compatibilidade com navegadores e dispositivos

A empresa Google garante compatibilidade com diversos navegadores inclusive o Internet Explorer a partir da versão 6.

- A aplicação foi testada no Internet Explorer versões 8 e 9, Safari versão 5, Chrome 28 e Firefox versão 21.
- Também foi testada nos seguintes processadores: i5 e i3 da Intel , Amd Dual Core e A6X do iPad 4.
- A aplicação foi testada em diferentes plataformas: notebooks, desktops, tablets (Xoom 2 e iPac 4) e celulares (Galaxy Aple, Lg Optimus e IPhone 3)

Em todos os testes o sistema funcionou normalmente independente dos processadores, navegadores e plataformas que foram testadas.

6.4 Clareza de usabilidade

Para o desenvolvedor e a equipe de projetos o sistema sempre é o mais claro possível. Essas opiniões não podem ser consideradas devido ao envolvimento no constante processo de evolução do sistema.

Pessoas de outras equipes de desenvolvimento e empresas parceiras foram usadas como base para determinar a clareza e a usabilidade do sistema. É claro, que toda aplicação precisa um treinamento básico e explicação das formas que ela pode ser empregada.

Apesar da compatibilidade do GWT é importante ressaltar que limitações físicas da tela do dispositivo podem dificultar a visualização do sistema, porém seu redimensionamento é bem eficiênte.

6.5 Resposta do cliente

Existe uma aplicação sendo executada em um shopping do Rio de Janeiro, na forma de piloto, para podermos avaliar o comportamento do Sistema em um

ambiente não controlado.

A aplicação foi apresentada em um congresso da Associação Brasileira de Shopping Centers (ABRASCE), no inicio de agosto desse ano, causando um impacto muito positivo de seus participantes, que puderam avaliar seu desempenho em tempo real.

Capítulo 7

Conclusões

Durante a execução do projeto tornou-se clara a viabilidade de um sistema de visualização de eventos comportamentais relacionados a estímulos externos, de forma dinâmica e de fácil visualização.

A arquitetura escolhida privilegia a escalabilidade; não é necessário tirar a aplicação do ar para disponibilizar atualizações. O aspecto modular da estrutura desenvolvida também privilegia a adição de novas funcionalidades ao sistema.

O sistema também apresenta grandes expectativas para se tornar uma ferramenta de apoio estratégico ao mercado de lojas e varejo.

Atualmente o sistema encontra-se em uma versão piloto instalada em um cliente e várias propostas e parcerias estão surgindo à empresa ICE Interactive, demostrando uma boa aceitação no mercado.

Com isso podemos concluir que os critérios de usabilidade, escalibilidade e modularidade foram preenchidos com sucesso.

Capítulo 8

Projetos Futuros

Muito se aprendeu no desenvolvimento desse trabalho; ao longo do processo algumas melhorias puderam ser implementadas e algumas possíveis falhas de projeto puderam ser mapeadas para que a cada versão do sistema correções sejam feitas.

O sistema "Horus Analytics" se apresenta na versão 4.0 e as versões 5.0 e 6.0 já estão em fase de planejamento. Nessa última já esta programada a mudança do acesso aos dados na cloud para uma API de acesso mais rápida, confiável e que permita ao usuário administrador ter um melhor controle sobre as informações das classes.

Outra mudança ainda não programada é permitir, em uma interface de auditoria, o acesso aos videos de onde os dados são originados para viabilizar um controle de qualidade dos gráficos fornecidos.

Devido à estrutura do sistema, que permite associar ferramentas ao perfil do usuário, muitas funcionalidades sugeridas pelos clientes estão sendo avaliadas para implementações futuras como upgrade da ferramenta. Dentre essas podemos destacar novos modelos de gráficos (pizza, pirâmide demográfica ...), relatórios gerados de forma automática periodicamente e podendo ser enviados por emails, configurações de gráficos favoritos e etc.

Referências Bibliográficas

[1] "GOOGLE APP ENGINE", Página oficial de desenvolvedores, https://developers.google.com/appengine/?hl=pt-br, 2012, (Acesso de novembro de 2012 à agosto 2013).

[2] "GOOGLE WEB TOOLKIT", Página oficial de desenvolvedores, https://developers.google.com/web-toolkit/?hl=pt-br, 2012, (Acesso de novembro de 2012 à agosto 2013).

[3] KUAN , Joe. "Learning Highcharts". London, 2013.

[4] SMEETS, Bram; BONESS, Uri; BANKRAS, Roald. "Programando Google Web Toolkit, do iniciante ao profissional". Rio de Janeiro, Alta Books, 2009.

[5] MICROSOFT AZURE. http://www.microsoft.com/azure/. (Acesso julho de 2013)

[6] CIURANA, E. Developing with Google App Engine. Apress, Berkely, CA, USA, 2009.

[7] LIU, S.; LIANG, Y.; BROOKS, M. Eucalyptus: a web service enabled e-infrastructure. In CASCON '07: Proceedings of the 2007 conference of the center for advanced studies on Collaborative research, pages 1-11, New York, NY, USA, 2007.

[8] ROBINSON, D. Amazon Web Services Made Simple: Learn how Amazon EC2, S3, SimpleDB and SQSWeb Services enables you to reach business goals faster. Emereo Pty Ltd, London, UK, 2008.

[9] LIMA, Fernando Cesar. Tópicos Avançados em Tecnologia Java - AJAX. Cornélio Procópio: UTFPR, 2010. (apostila).

[10] BILLING and Budgeting Resources. Disponível em: http://code.google.com/appengine/ docs/billing.html. (Acesso em: 13 agosto 2013).

[11] iGOOGLE. Disponível em: http://www.google.com/ig.(Acesso em: 13 agosto 2013).

[12] PORTAL GWT. Disponível em: http://portalgwt.com/. (Acesso em: 13 agosto 2013).

[13] WIDGET GALLERY. Disponível em: http://code.google.com/webtoolkit/doc/1.6/RefWidget Gallery.html. (Acesso em: 10 agosto 2013).

[14] AMAZON. Disponível em: http://aws.amazon.com/.

[15] APACHE CouchDB. Disponível em: http://couchdb.apache.org.

[16] BARROS, F. Cloud Computing: Prepare-se para a nova onda em tecnologia, Computerworld, Abril de 2008.

[17] CREEGER, M. Cloud Computing: An Overview, Communications of the ACM, Junho de 2009, pg. 3-4.

[18] GNU. Disponível em: http://www.gnu.org/licenses/licenses.html. (Acesso em 2013).

[19] SANDERSON, D. Programming Google App Engine, Editora O'Reilly Media, 2009.

[20] SCHOFIELD, J. Google angles for business users with "platform as a service", The Guardian, Abril de 2008.

[21] TAURION, C. Computação em Nuvem: Transformando o Mundo da Tecnologia da Informação, Editora Brasport, 2008.

[22] WIKIPEDIA. Disponível em: http://en.wikipedia.org/wiki/Infrastructure-as-a-service.

[23] NAITO, Daniel da Silva; ROSA, Eder Vinícius, Trabalho de graduação, Faculdade de Tecnologia de São José dos Campos, 2010.

[24] W3SCHOOLS. Ajax: Introduction. Disponível em: http://www.w3schools.com/ajax/ajax-intro.asp. (Acesso em agosto 2013).

[25] NoSQL East. Disponível em: http://nosqleast.com. (Acesso em agosto 2013).

[26] OPENCLOUD. The Open Could Manifesto. Disponível em: http://www.opencloudmanifesto.org. (Acesso em agosto 2013).

[27] SOUSA, Flávio R. C. ; MOREIRA, Leonardo O. ; MACÊDO, José Antônio F. de e MACHADO, Javam C. Gerenciamento de Dados em Nuvem: Conceitos, Sistemas e Desafios, Capítulo 4. Universidade Federal do Ceará (UFC), 2011.

[28] JUNIOR, Maurício L. de A. Google Web Toolkit: AJAX rápido, fácil e puro Java com o Google Web Toolkit (apostila).

[29] CHEE, Brian J. S. e JUNIOR, Curtis F. Cloud Computing : Tecnologies and Strategies of the Obiquitous Data Center. CRC Press, 2010.

[30] LIMEIRA, J. L. S. Utilização de AJAX no desenvolvimento de sistemas web, páginas de 19-20, 2006.

[31] TEIXERA, Luiz F. G. Uma introdução ao Google App Engine e GWT, 2012.

[32] IRANI, Romin K. Google App Engine Java Experiments. Apostila, 2010.

[33] KUZMENKO, Ivan. Google App Engine and Datastore. Apresentação 2010.

[34] DEITEL & DEITEL. Java como Programar. Tradução da sexta edição. Editora Pearson, 2005.

[35] HEJLSBERG, Anders; WILTAMUTH, Scott e GOLDE, Peter. The Java Programming Language. Addison-Wesley, 2003.

[36] ROWE, Glenn. an Introduction to Data Structures and algorithms with Java. Prentice hall Europe, 1998.

[37] STAIR, Ralph M. Princípios de sistemas de Informação : Uma abordagem gerencial. LTC editora, 1998.

[38] IBOPE inteligência. Disponível em: http://www.ibope.com. (Acesso em agosto 2013).

[39] MILLER, Victor D. Desenvolvimento de aplicações sob o paradigma da computação em nuvem com ferramentas Google. Trabalho de conclusão de curso. Universidade Federal de Santa Catarina, 2010 .

[40] ANDERSON, Chris. A cauda longa: do mercado de massa para o mercado de nicho. Rio de Janeiro: Elsevier, 2006.

[41] CHANG, Fay et al. Bigtable: A Distributed Storage System for Structured Data. OSDI, 2006.

[42] KEENE, Christopher. What Is Platform as a Service (PaaS)? - 18 de Março de 2009. Disponível em: http://www.keeneview.com/2009/03/what-is-platform-as-service-paas.html (Acesso em agosto de 2013).

[43] FOSTER, Ian; ZHAO Yong; RAICU, Ioan e LU, Shiyong. Cloud Computing and Grid Computing 360-Degree Compared. Department of Computer Science, University of Chicago.

[44] GOOGLE ARCHITECTURE. High Scalability. 22 de Novembro de 2008. Disponível em: http://highscalability.com/google-architecture (Acesso em agosto de 2013).

[45] URL FETCH JAVA API OVERVIEW. Disponível em: https://developers.google.com/appengine/docs/java/urlfetch/

[46] WIKIPEDIA. Cloud computing. Disponível em: http://en.wikipedia.org/wiki/Cloud-computing (acesso em agosto de 2013).

[47] HIGHCHARTS. Disponível em: http://www.moxiegroup.com/moxieapps/gwt-highcharts/ (acesso em agosto de 2013).

[48] WIKIPÉDIA Google Web Toolkit . Disponível em: http://pt.wikipedia.org/wiki/Google-Web-Toolkit. (acesso em agosto de 2013).

[49] FXP Labs e-commerce eficiente . Disponível em: http://www.fxplabs.com.br/blog/nosql-conceito-de-banco-de-dados-nao-relacional/. (acesso em agosto de 2013).