UNIVERSIDADE FEEVALE

LUCAS DEMUTI DIAS

AVISA IMÓVEL:
Aplicação WEB com NoSQL

Novo Hamburgo
2013

LUCAS DEMUTI DIAS

AVISA IMÓVEL:
Aplicação WEB com NoSQL

Trabalho de Conclusão de Curso
apresentado como requisito parcial
à obtenção do grau de Bacharel em
Sistemas de Informação pela
Universidade Feevale

Orientador: Juliano Varella de Carvalho

Novo Hamburgo
2013

AGRADECIMENTOS

Especialmente quero agradecer a minha namorada Gabriele, que sempre esteve presente me apoiando, entendendo a distância e a ausência em vários momentos durante este período e a muita paciência depositada.

Quero agradecer a minha mãe, que tenho certeza que somente consigo concluir mais esta etapa pela educação, força e conselhos que ela me deu.

Também quero agradecer a minha irmã, primos, demais familiares e amigos por toda ajuda que me deram para que eu conseguisse chegar aqui.

Ao meu orientador quero deixar meu muito obrigado por todas as dicas, ajudas e paciência no decorrer deste um ano.

Por fim, quero agradecer a Emanuela, Cláudio e ao Rodrigo por toda confiança depositada em mim, com toda certeza sem vocês não seria possível finalizar este trabalho.

RESUMO

O crescimento acelerado da construção civil e do mercado imobiliário faz com que as imobiliárias utilizem meios de publicidade para alcançar o consumidor. Em uma era digital, uma das principais formas é através da internet. Com isso, o consumidor procura em inúmeros portais imóveis do seu interesse e a cada busca, é necessário informar diversas vezes as mesmas características do imóvel pretendido. Deste modo, este trabalho tem como objetivo conectar, através de uma ferramenta digital, as imobiliárias e o consumidor, de forma inteligente e ativa. A ligação entre as duas pontas do processo é realizada aplicando soluções tecnológicas, que permitem o acesso rápido a informação. Além da disponibilidade, a escalabilidade do sistema é essencial, para que à medida que o número de usuários aumente, não ocorra queda do desempenho. Portanto, foi desenvolvida uma aplicação Web, utilizando um banco de dados relacional, quando a funcionalidade do sistema exigiu consistência e um banco de dados NoSQL (*Not Only* SQL – Não apenas SQL) quando exigiu escalabilidade e alta disponibilidade.

Palavras-chave: Imobiliária. NoSQL. Software Imobiliário. Mercado Imobiliário.

ABSTRACT

The civil construction and the housing market are growth. Therefore, the real estate agency needs to use advertising media to reach customers. In a digital age, internet is one of the main means advertising. With that, the consumers search for a property that interest you in several websites. But for each search is necessary to inform many times the same characteristics of the intended property. Thus, the present study aims to connect to real estate agency and customers by a smart and active digital tool. The link between the two ends of the process is accomplished by applying technological solutions that allow quick access to information. In addition to availability, scalability of the system is essential. So, with the increases of the number of users, the performance degradation will not occur. Therefore, it was developed a web application using a relational database and a NoSQL (Not Only SQL) database. Whether the functionality of the system requires consistency, the system uses the relational database. Whether it requires scalability and high availability use the NoSQL database.

Keywords: real estate agency. NoSQL. Real estate software. Housing market.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	Interface de Programação de Aplicativos
AWS	Amazon Web Services
CAP	Consistência, Disponibilidade e Tolerância ao Particionamento
CIL	Linguagem Intermediária Comum
CLR	Common Language Runtime
JSON	JavaScript Object Notation
MVCC	Multiversion Concurrency Control
NoSQL	Não Apenas SQL
SGBD	Sistema Gerenciador de Banco de Dados
SGBDR	Sistema Gerenciador de Banco de Dados Relacional
SQL	Linguagem de Consulta Estruturada

SUMÁRIO

INTRODUÇÃO

No Brasil, a construção civil está crescendo em ritmo acelerado, o elo deste setor com os consumidores são as imobiliárias. Atualmente, os consumidores estão com o crédito facilitado, fazendo com que o mercado imobiliário atravesse um momento histórico no país (MARUCCI, 2012).

Em Porto Alegre, no ano de 2002 foram feitas 2078 vendas de imóveis, com uma velocidade de venda em média de 6,5% (média do percentual das vendas no mês dividido pela oferta no início do período). Já no ano de 2012, o número de vendas subiu para 4751 imóveis, com uma velocidade média de venda de 7,58%, conforme dados elaborados pelo Banco de Dados da CBIC (Câmara Brasileira da Indústria da Construção) com fonte no SINDUSCON-RS (Sindicato da Indústria da Construção Civil no Estado do Rio Grande do Sul).

Frente a esse aquecimento do mercado imobiliário, as imobiliárias necessitam meios de publicidade para alcançar o consumidor. Uma dessas formas é através da presença no mundo digital, podendo neste ambiente publicar todos os imóveis disponíveis a venda ou à locação.

Do outro lado, conforme afirma Marucci (2012), o consumidor que está com o crédito facilitado procura adquirir a sua primeira moradia, investir em imóveis, alugar uma loja ou outros variados tipos de negociação possíveis no mercado. Para isto, este consumidor inicia uma busca, entre as imobiliárias existentes em sua cidade, atrás de um imóvel de acordo com as características que ele deseja.

Em um mundo cada vez mais virtual, o principal meio de procurar imóveis é através de pesquisa no site de cada imobiliária ou a utilização de portais que concentram anúncios de várias imobiliárias. Porém, a cada dia surgem novas oportunidades de negócios que podem se adequar melhor ao que o consumidor procura. Deste modo, é necessário a cada busca informar diversas vezes as características do imóvel pretendido, nos vários sites de imobiliárias ou de portais.

Diante deste cenário é necessário um sistema, que seja capaz de avisar ao consumidor quando um imóvel com as características desejadas ficam disponíveis no mercado. É neste contexto que o trabalho em questão se concentra, criando a ferramenta adequada para que o consumidor seja avisado sobre os imóveis de seu interesse. Sendo assim,

o consumidor se cadastra em um portal informando uma única vez o que procura e é avisado sempre quando surgirem imóveis adequados ao seu perfil.

Os imóveis são cadastrados de forma automática através de um programa de computador. Este *software* é um robô, que obtém as informações do site de cada imobiliária e faz o cadastro no portal automaticamente, ou seja, é desenvolvida uma integração do site com o sistema. Além disso, é possível através de um formulário no portal que qualquer vendedor cadastre um imóvel.

Esta ferramenta é vantajosa tanto para o consumidor, quanto para as imobiliárias. Os consumidores recebem, sem perder tempo, imóveis de acordo com o que procuram, não recebendo informações de imóveis diferentes do desejado e também não correndo o risco de o corretor de imóveis não lhes informar sobre a nova oportunidade de negócio. Por sua vez, as imobiliárias possuirão mais um meio de publicidade, no qual chegam até o consumidor de forma inteligente e oportuna, não sendo necessário investir em material humano para anunciar as oportunidades.

A ferramenta é capaz de conectar automaticamente várias imobiliárias com o consumidor. Com o alto número de imóveis disponíveis no mercado, se faz necessária a utilização de meios tecnológicos que disponibilizem a informação rapidamente e que permitam a escalabilidade à medida que o número de usuários aumente, para que não ocorra queda do desempenho.

Atualmente existem duas principais tecnologias de Banco de Dados capazes de armazenar dados em grande escala: o Modelo Relacional de Dados e o NoSQL (*Not Only* SQL – Não apenas SQL). O Modelo Relacional de Dados criado em 1970 é o mais utilizado em todo o mundo. Tem sua estrutura baseada em tabelas que se relacionam através de chaves estrangeiras tornando o sistema extremamente consistente, porém existem limitações quando é exigido deste modelo disponibilidade e escalabilidade (BRITO, 2010). Para solucionar a escalabilidade e disponibilidade dos sistemas iniciaram-se vários estudos para projetar novos modelos de bancos, capazes de atender a estes requisitos. Então, foram criados vários sistemas de bancos em que a consistência é menor, porém com alta disponibilidade e de fácil escalabilidade, sendo conhecido por NoSQL (DIANA & GEROSA, 2010).

Os bancos NoSQL não têm como objetivo substituir o Modelo Relacional de Dados, mas sim atender casos em que os requisitos exijam disponibilidade e escalabilidade em sistemas com grande carga de dados. (ALMEIDA & BRITO, 2012). Isto significa que pode

ser desenvolvida uma aplicação utilizando os dois modelos de bancos de dados, de acordo com a necessidade de cada funcionalidade do sistema.

Portanto, esta ferramenta será desenvolvida utilizando a linguagem de programação *.NET* que oferece suporte para Web. A ferramenta armazena os dados em SGBDR (Sistemas de Gerenciamento de Banco de Dados Relacional) e um SGBD NoSQL. Desta forma, quando uma determinada funcionalidade exigir consistência será utilizado o Modelo Relacional de Dados e quando necessário disponibilidade e escalabilidade será usado o SGBD NoSQL.

Neste trabalho o capítulo 1 aborda as principais mídias existentes e o ambiente atual de divulgação dos imóveis das imobiliárias. No capítulo 2 são definidas as tecnologias utilizadas e as funcionalidades do sistema desenvolvido. No capítulo 3 são analisados alguns bancos de dados NoSQL e define-se qual desses bancos este trabalho usou. Em seguida, no capítulo 4, são definidos os requisitos funcionais e não-funcionais, os casos de uso, o desenvolvimento da ferramenta e o resultado da aplicação. Ao término, o capítulo 5 descreve os testes executados, avalia a utilização do banco de dados NoSQL e apresenta a avaliação dos usuários sobre o sistema, além de problemas e dificuldades encontradas no decorrer deste trabalho.

1 AMBIENTE ATUAL

No Brasil, a demanda pelos serviços do setor da construção civil está cada vez aumentando. Com isto, as imobiliárias que fazem o elo com o consumidor, acompanham este crescimento acelerado. Tal demanda tem aumentado devido ao consumidor estar com o acesso ao crédito facilitado e por causa disso o mercado imobiliário atravessa um momento histórico no país (MARUCCI, 2012).

De acordo com os dados elaborados pelo Banco de Dados da CBIC (Câmara Brasileira da Indústria da Construção) com fonte no SINDUSCON-RS (Sindicato da Indústria da Construção Civil no Estado do Rio Grande do Sul), na cidade de Porto Alegre, no ano de 2002 foram feitas 2078 vendas de imóveis, com uma velocidade de venda em média de 6,5% (média do percentual das vendas no mês dividido pela oferta no início do período). Já no ano de 2012, o número de vendas subiu para 4751 imóveis, com uma velocidade média de venda de 7,58%. Sendo assim, de 2002 para 2012 ocorreu um crescimento aproximado de 128% no número de imóveis vendidos, comprovando o aquecimento do mercado imobiliário.

1.1 Publicidade

As mídias publicitárias estão se expandindo e focando cada vez mais para atingir seu público alvo. A escolha da mídia é feita comparando os perfis do público alvo que se deseja atingir. Com isso, as mídias alternativas como televisão, jornais, internet e rádio vêm sendo exploradas (DORDOR, 2007).

Define-se a mídia alternativa como aquela “que não é tradicional” (ATTON, 2001). Em muitos casos, não há um conceito geral de mídia alternativa, mas uma descrição de características ou de critérios de identificação restritivos. Segundo Perez e Barbosa (2008), a mídia alternativa atualmente se encontra organizada em quatro grupos e apresenta resultados comprovados aos que anunciam nelas:

- Mídia Impressa: jornais, revistas e *folders*;
- Mídia Eletrônica: televisão, rádio e cinema;
- Mídia Exterior: *Outdoor*, painéis eletrônicos, placas de rua, cartazes, *banners* e pontos de ônibus;
- Mídia Digital de Relacionamento: Internet, celulares e jogos digitais.

Na figura 1.1 é apresentada a divisão dos investimentos em mídias no Brasil em 2012. Os dados são do Projeto Inter-meios, que tem como objetivo levantar o volume de investimento publicitário em mídia no Brasil. É possível identificar que há uma enorme importância na mídia da televisão aberta, que concentra 64,70%, seguido por investimentos de 11,24% em jornal. Na quarta posição encontra-se a internet, que obteve um percentual de 5,03% (INTER-MEIOS, 2012).

Houve grande aumento dos investimentos em internet em comparação a 2005, onde o percentual de investimento era de 1,66%, e o de jornal era de 16,30%. Sendo assim, é possível identificar o crescimento da internet e a queda dos investimentos em jornal, claramente apresentando o quanto a publicidade na internet aumenta.

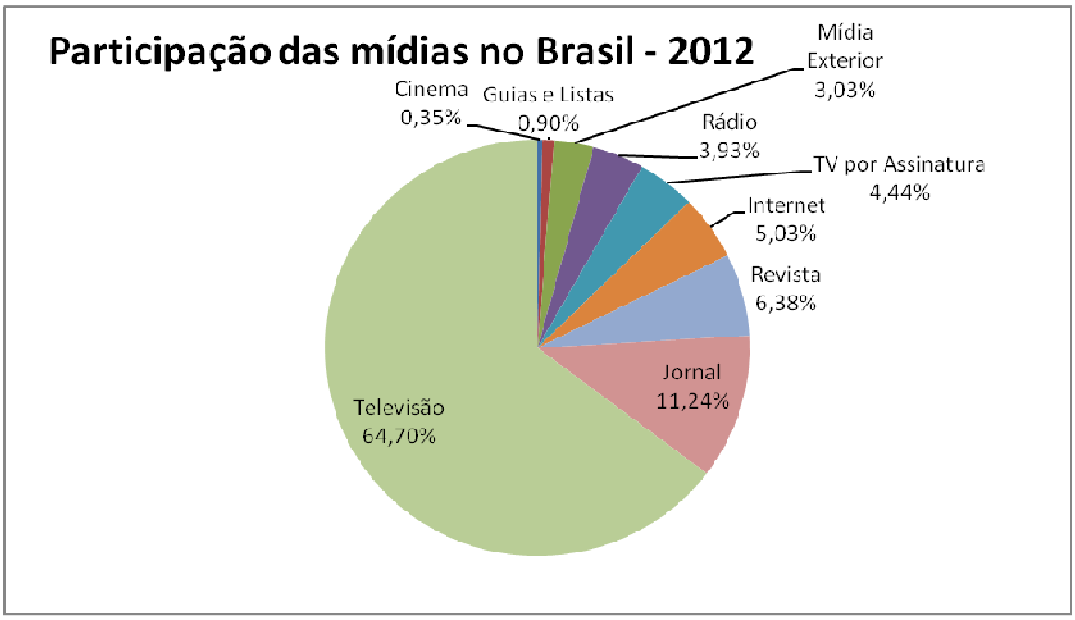


Figura 1.1 – Participação das mídias no Brasil em 2012
Fonte: INTERMEIOS (2012)

1.2 Mídias

De acordo com Perez e Barbosa (2008) não cabe falar sobre todos os meios publicitários disponíveis, o que seria praticamente impossível. Todavia, a seguir serão apresentadas as características mais significativas das principais mídias, no tocante aos aspectos publicitários.

1.2.1 Televisão aberta

A televisão aberta é um meio de comunicação forte e um importante meio de publicidade. Dificilmente encontra-se um lar brasileiro sem uma televisão, ela se faz presente

na vida e sem ela não há comunicação. Existe uma divisão entre a televisão aberta e fechada, sendo que a televisão aberta é utilizada para anunciar grandes marcas, sendo segmentada geograficamente. O “horário nobre”, que se estende das 18 horas às 23 horas, concentra maior audiência, ou seja, um número maior de pessoas assistindo, sendo isso muito valorizado para a publicidade (PEREZ & BARBOSA, 2008).

1.2.2 Jornal

De acordo com Perez e Barbosa (2008), o jornal é um meio muito tradicional de publicidade, muito antigo e atende um publico mais qualificado. O jornal favorece a crítica e o debate, possuindo uma alta credibilidade, pois caracteriza compromisso formal com o leitor. Embora atinja um público menor, que a televisão aberta, ainda possui muitos leitores. O faturamento com vendas de jornal está estável, devido aos anúncios e esforços comerciais de oferecer anúncios, com preços diferenciados, tornando o veículo mais atraente. Não existe um jornal a nível nacional, pois a maioria dos jornais diários do Brasil tem sua zona de influência circunscrita à cidade de origem. Com isso, na maioria dos casos, limita-se a veicular propaganda local, sendo o setor varejista, o grande sustento publicitário do jornal (TAMANAHA, 2006).

1.2.3 Revista

O público consumidor de revistas abrange as classes A, B e C, que juntas apresentam 83% de todos os leitores, segundo Perez e Barbosa (2008). Apesar disso, o consumo é muito baixo em no Brasil. Os principais diferenciais das revistas em relação aos jornais têm diminuindo, pois ambas possuem um apelo a propaganda, e a exposição de marcas conhecidas. Publicitariamente, se diz que as revistas em geral falam mais intimamente a seu leitor, que em tese, dedica mais tempo e interesse a sua leitura. A durabilidade da revista também é maior que o jornal, ela pode durar durante muito tempo e ficar exposta também durante semanas, diferente do jornal que é exposto diariamente. O número de leitores também é maior que no caso dos jornais, isso significa que um anúncio em uma revista também atinge mais que um leitor (PEREZ & BARBOSA, 2008).

1.2.4 Rádio

Perez e Barbosa (2008) afirmam que com o passar dos tempos, o rádio teve uma queda na participação de mídias. Com o surgimento da televisão, o desaparecimento do radio

foi relevante, pois há um grau de atenção menor do que a televisão. A vantagem dos anúncios em rádios ocorria pelo excessivo uso em massa, e um baixo custo de veiculação e produção. As rádios AM trazem notícias, e as FM trazem entretenimento, porém ambas trazem anúncios. A rádio e a Internet têm muito a convergir, pois muitas emissoras disponibilizam a sua programação on-line por meio de seus sites. Por outro lado, com o avanço da tecnologia, os softwares de programação sonora permitem a criação de emissoras personalizadas, fazendo com que o rádio perca a audiência (PEREZ & BARBOSA, 2008).

1.2.5 Outdoor

O outdoor é um "cartaz" que é exposto ao ar livre ou à margem de vias publicas, em locais de grande visibilidade. Os outdoors são comercializados com o intuito de realizar propagandas publicitárias fazendo com que a marca sejas lembrada. A palavra outdoor é de origem inglesa e, em inglês, tem sentido totalmente diverso do seu significado em português. *Billboard* é a palavra inglesa para qualquer propaganda (painel, letreiro luminoso) (PEREZ & BARBOSA, 2008).

Embora seja muito utilizado, o outdoor gera críticas de muitos analistas por contribuir com uma parcela da chamada poluição visual, que aflige principalmente cidades grandes (PEREZ & BARBOSA, 2008).

1.2.6 Internet

A internet é a mídia que mais cresceu em 2012, mundialmente, conforme dados publicados por Nielsen (2012). Este crescimento se deve ao fato de a internet ser a evolução da publicidade, complementando as outras mídias existentes (KENDZERSKI, 2007). Além disso, Davini (2013), presidente do IAB Brasil afirmou: "O Digital oferece aos anunciantes alta efetividade aliada a ferramentas de mensuração muito poderosas". Por isso a publicidade na internet tende a crescer fazendo com que os investimentos migrem para a esta mídia, ao invés de outras. Deste modo, a internet é objetiva ao induzir o consumidor a encontrar e visualizar a marca, além de prover ferramentas capazes de avaliar a repercussão de uma campanha publicitária.

1.3 Publicidade das imobiliárias

Frente ao aquecimento do mercado imobiliário, as imobiliárias necessitam utilizar diversos meios de publicidade para alcançar o consumidor. Várias mídias são utilizadas

atualmente como revistas, jornais e *outdoor*. Outro meio que cresce cada vez mais é o mundo digital, em que a imobiliária pode ter seu próprio site, anunciar em sites de classificados *online* e ter perfis em redes sociais.

O investimento das imobiliárias em publicidade visa alcançar o consumidor, que está com o crédito facilitado, com o crescimento de sua renda e usufruindo de programas de incentivos governamentais (MARUCCI, 2012). Estes consumidores procuram adquirir a sua casa própria, investir em imóveis, alugar uma loja ou outros variados tipos de negociação existentes no mercado.

O consumidor, conectado ao mundo virtual, utiliza da facilidade da internet para procurar o imóvel pretendido, de acordo com as características desejadas. Esta busca é realizada, na maioria das vezes, por meio do site de cada imobiliária ou em portais que concentram anúncios de imóveis. Porém, a cada dia surgem novas oportunidades de negócios que podem se adequar melhor ao que o consumidor procura. Deste modo, é necessário, a cada busca, informar as características do imóvel pretendido, nos vários sites de imobiliárias ou de portais.

1.4 Imobiliárias

Através de uma reunião com o Prof. Me. Hugo Springer Junior, coordenador dos Cursos Superiores de Tecnologia em Construção de Edifícios e Engenharia Civil da Universidade Feevale, foram obtidos contatos de algumas imobiliárias da região. Também foram realizadas pesquisas no Google para encontrar imobiliárias que atuam na cidade de Novo Hamburgo.

Após estas buscas, selecionaram-se oito imobiliárias e foi realizada uma análise dos sites destas empresas. Na análise foi levantada a arquitetura do site, a quantidade de oportunidades de negócio (quantidades de imóveis disponíveis no site no dia 8 de maio de 2013) e a quantidade de tipos de negociação oportunizados (locação e/ou venda), conforme exibido na tabela 1.1.

Para determinar qual era o modelo de arquitetura do site estudaram-se as funcionalidades e o código HTML das principais páginas do site (lista de imóveis e detalhe do imóvel). Verificou-se a disposição das informações e o padrão de endereço eletrônico utilizado para acessar cada página. Basicamente, sites desenvolvidos pelo mesmo fornecedor seguem um padrão do código HTML e também o mesmo modo de obter a listagem dos

imóveis disponíveis para posterior acesso a página de detalhamento do imóvel. Esta avaliação é necessária, pois a diferença entre os sites determinará a quantidade de programas a serem desenvolvidos na aplicação, que obterá os imóveis dos sites das imobiliárias, então sites que tinham o código HTML muito parecido na página de listagem e detalhe de imóvel foram categorizados como tendo o mesmo tipo de arquitetura.

Tabela 1.1 – Análise das imobiliárias

	Tipo da Arquitetura do site	Oportunidades de negócios	Quantidade de tipos de negociação
Imobiliária 1	1	700	2
Imobiliária 2	3	470	2
Imobiliária 3	4	120	1
Imobiliária 4	2	750	2
Imobiliária 5	1	670	2
Imobiliária 6	1	1580	2
Imobiliária 7	1	340	1
Imobiliária 8	5	280	2

Fonte: Próprio Autor. 2013.

Diante dos valores apresentados na Tabela 1.1, selecionou-se a Imobiliárias 1, a Imobiliária 4 e a Imobiliária 6, para que sejam obtidos os imóveis de seus sites e inseridos no sistema a ser desenvolvido neste trabalho. Para esta seleção considerou-se as imobiliárias com maior número de oportunidade de negócios e que oferecessem tanto venda quanto locação como tipos de negócio, com o objetivo de que a aplicação a ser desenvolvida neste trabalho tenha o maior número de imóveis. Além disso, para cumprir o cronograma, definiu-se que desenvolverá somente dois programas para obter os imóveis, portanto dois dos três sites selecionados devem ser da mesma arquitetura e o outro de uma diferente.

Depois de selecionadas, entrou-se em contato com a Imobiliária 1 por telefone, correio eletrônico e visitas. Foram apresentados os objetivos deste trabalho e a imobiliária foi convidada a participar. Esta aceitou a participação e autorizou o trabalho a obter os imóveis do seu site, conforme pode ser visto no Apêndice A.

Com a Imobiliária 4 não foi conseguido o contato para que fosse apresentado o trabalho para os responsáveis pela decisão de participar ou não. Já a Imobiliária 6 foi contatada por telefone e marcada uma reunião com o responsável, nesta foi apresentado os objetivos do trabalho e os problemas existentes hoje na publicidade utilizada pelo mercado. Em contrapartida a imobiliária se mostrou desinteressada no trabalho alegando que seus

corretores faziam o que este trabalho se propõe a fazer e, portanto não seria importante a ferramenta para a empresa.

Diante disso, foi obtido o contato do responsável pela Imobiliária 7, no qual foi contatado por telefone e em uma reunião foi apresentado os objetivos, seguindo o padrão anterior feito nas outras imobiliárias, esta aceitou participar do trabalho, concluindo como um grande diferencial e caracterizando o projeto como perfeito para o consumidor, porém por outro lado igualando as imobiliárias que investem mais em mídias com as que investem menos. Além disso, foi autorizado por correio eletrônico este trabalho a obter os imóveis do seu site e publicar no Avisa Imóvel, conforme pode ser visto no Apêndice A.

2 SISTEMA

Para atingir os objetivos deste trabalho foi necessário desenvolver três módulos que compõe um sistema capaz de conectar os consumidores com o mercado imobiliário. Para isso foi necessário definir as tecnologias que seriam utilizadas para a construção deste trabalho.

2.1 Tecnologias

Essas aplicações foram desenvolvidas utilizando uma linguagem de programação, um banco de dados relacional e um banco de dados NoSQL. Abaixo são apresentadas a linguagem e o banco relacional escolhido e no capítulo 3 é pesquisado e definido o sistema NoSQL. A utilização de dois tipos de bancos de dados visa atingir o objetivo de prover escalabilidade e disponibilidade para as aplicações que necessitarem destes princípios utilizando NoSQL. Entretanto, quando a aplicação necessitar de alta consistência e relacionamentos será utilizado o modelo relacional.

2.1.1 Linguagem de Programação

Para as aplicações a serem desenvolvidas é preciso que a linguagem a ser utilizada seja atual, orientada a objetos e que dê suporte para o desenvolvimento voltado à *Web*. Com estas características existem duas que atualmente são utilizadas em grande escala: *Java* e *.NET*. A Oracle (2013) descreve o *Java* como:

> "[...] a base de praticamente todos os tipos de aplicativos em rede, e é o padrão global para desenvolvimento e fornecimento de aplicativos para celular, jogos, conteúdo on-line e software corporativo. Com mais de 9 milhões de desenvolvedores em todo o mundo, o Java permite desenvolver e implantar aplicativos e serviços incríveis de maneira eficiente. Com ferramentas abrangentes, um ecossistema sólido e um desempenho eficiente, o Java oferece a portabilidade de aplicativos mesmo entre os ambientes computacionais mais diferentes."

Enquanto isso a Microsoft (2013) descreve o *.NET* como sendo:

> "[...] o modelo de programação completo e consistente da Microsoft para a criação de aplicativos que oferece uma experiência visualmente surpreendente aos usuários, comunicação segura e sem interferências e a capacidade de modelar uma variedade de processos de negócios."

O *.NET* é um *framework* que, de acordo com a Microsoft (2013), oferece suporte a criação e execução da próxima geração de aplicativos e serviços da *Web*. Um *framework* é formado por várias classes que os desenvolvedores podem usar, estender ou adaptar para uma solução específica (FAYAD & SCHMIDT, 1997).

A Microsoft (2013, tradução nossa) define os seguintes objetivos do *.NET Framework*:

> "Fornecer um ambiente de programação consistente orientado a objetos, que o código compilado seja armazenado e executado localmente ou seja executado localmente mas distribuído pela Internet ou executado remotamente;
>
> Fornecer um ambiente de execução que minimize conflitos de versionamento e de publicação;
>
> Fornecer um ambiente de execução que promova a execução segura de código, incluindo os códigos criados por desconhecidos ou código de terceiros com baixo nível de confiança;
>
> Fornecer um ambiente de execução que elimine os problemas de desempenho dos ambientes interpretados ou com *scripts*;
>
> Tornar a experiência do desenvolvedor consistente, através dos diversos tipos de aplicativos, como aplicativos baseados no Windows e aplicativos baseados na Web;
>
> Executar toda comunicação usando padrões da indústria, garantindo assim que códigos baseados no .NET Framework possam se integrar a qualquer outro código."

Ainda, o *.NET Framework* é um ambiente de execução composto por um mecanismo de gerenciamento do código em execução (*CLR*), gerenciando a memória e provendo a interoperabilidade de linguagens de programação, devido aos compiladores gerarem o *CIL* (Linguagem Intermediária Comum). Pelos conhecimentos do autor deste trabalho e as vantagens oferecidas, a linguagem utilizada no trabalho foi o C# com o *.NET Framework.*

2.1.2 Banco de Dados Relacional

O modelo relacional de banco de dados comercialmente surgiu em meados de 1980, sendo até hoje utilizado na maioria dos sistemas desenvolvidos, conhecidos como Sistemas Gerenciadores de Dados Relacionais (SGBDRs). Os mais utilizados são: DB2 da IBM, o Oracle da Oracle, o SQLServer da Microsoft e os de código aberto MySQL (comprado pela Oracle) e o PostgreSQL (ELMASRI & NAVATHE, 2010).

O SGBDRs comerciais utilizam uma linguagem de consulta padrão chamada SQL (*Structured Query Language*), porém cada sistema oferece suas vantagens e desvantagens (ELMASRI & NAVATHE, 2010). Neste trabalho será optado pelo MySQL, que tem um bom custo-benefício, é um banco de dados altamente difundido entre as hospedagens de servidores *Web* no Brasil, além de ser oferecido nos planos de hospedagem de forma gratuita. (LOCAWEB, 2013).

Além disso, a Oracle (2013) apresenta as seguintes facilidades do MySQL:

> "Fácil de usar — Vá do download à instalação completa em menos de 15 minutos
> Menor TCO — Implante o MySQL para aplicações essenciais com economias significativas nos custos em relação ao Microsoft SQL Server
> Escalabilidade e desempenho — Atenda às necessidades de escalabilidade e desempenho dos web sites com mais tráfego e das aplicações mais exigentes. [...]"

2.2 O Sistema

O sistema desenvolvido é composto por três módulos nomeados: *Web*, *Seeker* e *Job*. Em síntese, o sistema funciona da seguinte forma: os usuários cadastram o padrão de imóvel que procuram, criando um perfil; as imobiliárias são integradas de forma automática com o sistema ou cadastram os imóveis manualmente; as informações dos imóveis e do perfil dos usuários são cruzadas e são geradas sugestões de imóveis, que são enviadas por e-mail para o usuário. A seguir são detalhados cada módulo desenvolvido.

2.2.1 Módulo Web

O módulo Web faz a interação dos usuários com os dados armazenados, como imóveis e sugestão de imóveis. Neste módulo o funcionamento do sistema é explicado em algumas telas e o restante do conteúdo, que contém as informações dos imóveis e sugestões, só é possível acessar mediante cadastro.

Conforme a figura 2.1, este módulo disponibiliza o cadastro de usuários, cadastro das preferências por imóveis, cadastro de imobiliárias, cadastro de imóveis, visualização de sugestões e visualização de imóveis. Além disso, todo o módulo foi desenvolvido com suporte a multi-idiomas, sendo toda a modelagem dos dados e a aplicação criada para sustentar isso, utilizando os padrões de globalização oferecidos pela linguagem, porém neste momento somente o português estará disponível.

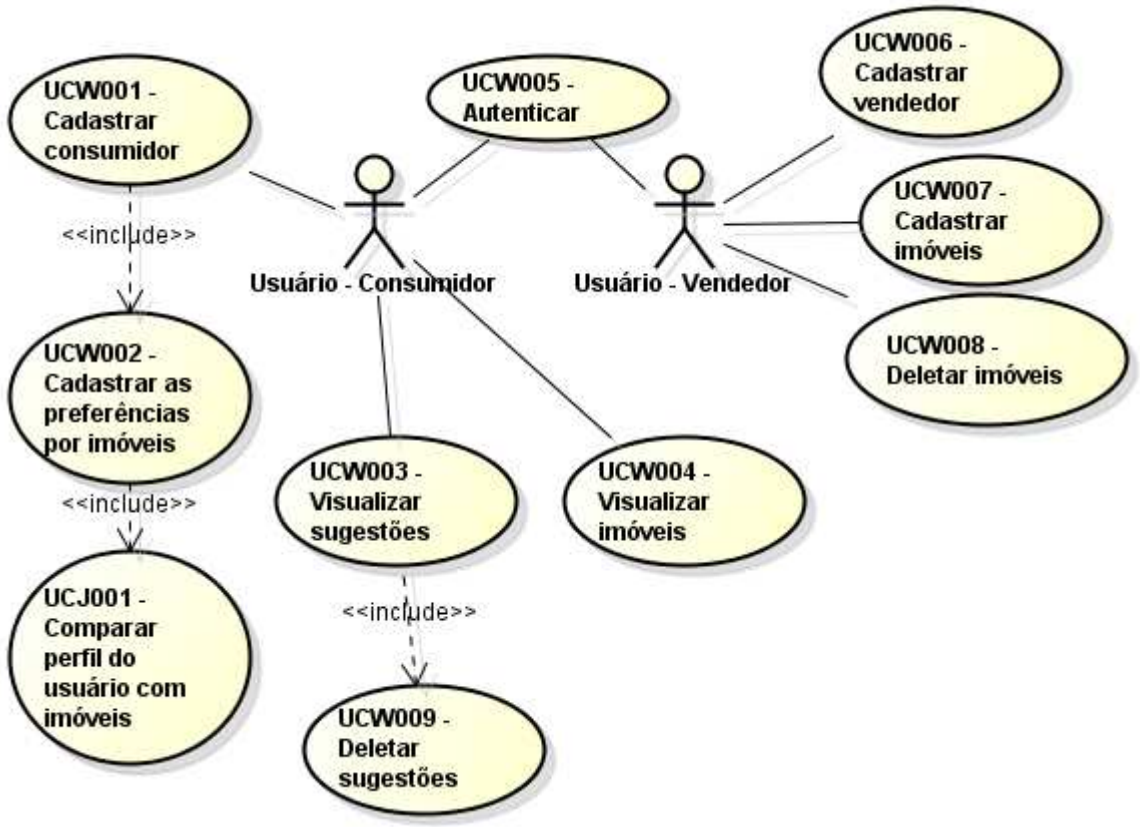


Figura 2.1 – Diagrama de caso de uso do módulo Web
Fonte: Próprio Autor (2013)

O cadastro de usuários exige o preenchimento do nome, email, telefone e senha. Em seguida, é exigido do usuário que ele faça o cadastro das preferências por imóveis, que são os padrões de imóveis que ele procura. Neste será preciso informar para cada preferência o tipo de negociação desejada (aluguel ou compra), o tipo do imóvel (apartamento, casa, terreno, entre outros), a faixa de valor, o estado, a cidade e os bairros da cidade escolhida, além de informações adicionais opcionais como número de dormitórios, garagens e banheiros procurados.

A quantidade de preferências cadastradas pelo usuário é ilimitada, permitindo ao usuário fazer inúmeras combinações entre as características de imóveis que ele procura. Por fim, o usuário deve indicar a frequência com que deseja receber os e-mails (diariamente, semanalmente ou mensalmente). Ao cadastrar as preferências, o caso de uso do módulo *Job* é incluído para que ele gere as sugestões para o usuário. Esses dados necessitam de forte relacionamento entre as tabelas e serão armazenados no MySQL.

O cadastro de imobiliária pode ser feito tanto por pessoa física quanto por pessoa jurídica, disponibilizando desta forma que qualquer usuário possa cadastrar um imóvel no sistema e que este seja sugerido aos usuários consumidores. O cadastro de imóvel exigirá o

tipo de negociação (para aluguel ou para compra), o tipo do imóvel (apartamento, casa, terreno, entre outros), a faixa de valor, o estado, a cidade e o bairro. Ainda será possível adicionar fotos, link do imóvel em outro site, descrição do imóvel e informações adicionais como número de dormitórios, garagens e banheiros. Esses dados necessitam de forte relacionamento entre as tabelas e serão armazenados no MySQL.

A visualização de sugestões estará disponível para os usuários que procuram imóveis. As sugestões serão geradas pelo módulo *Job* e ficarão armazenadas no banco NoSQL, pois a quantidade de dados inseridos nesta etapa crescerá à medida que o uso do sistema aumentar e portanto a escalabilidade é essencial. É possível ordenar as sugestões buscando as mais recentes ou também as mais recomendadas. As sugestões ficarão ativas enquanto o imóvel estiver disponível.

A visualização de imóveis disponibilizará o acesso detalhado do imóvel. Cada vez que um usuário acessar a página de detalhe de um imóvel, essa informação será armazenada no banco NoSQL, para futuros relatórios de acesso aos imóveis sugeridos. Além disso, caso exista um endereço eletrônico externo, por exemplo, o imóvel no site da imobiliária, e o usuário clique neste link, essa informação também será armazenada no NoSQL. Ainda, todas as informações extras, como *log* de avisos ou erros, serão armazenadas no NoSQL.

O módulo Web também é composto por um sistema gerenciador, que tem acesso a todos os *logs* do sistema e qualquer funcionalidade extra que é necessária ação do administrador da ferramenta. Uma funcionalidade já implantada é a inserção, atualização e exclusão de itens que ajustam a que bairro pertence o imóvel obtido na integração, utilizado em casos que este bairro não exista no sistema ou em que esteja com a grafia incorreta.

2.2.2 Módulo *Seeker*

O módulo *Seeker* integra o sistema com os sites das imobiliárias escolhidas, conforme figura 2.2.

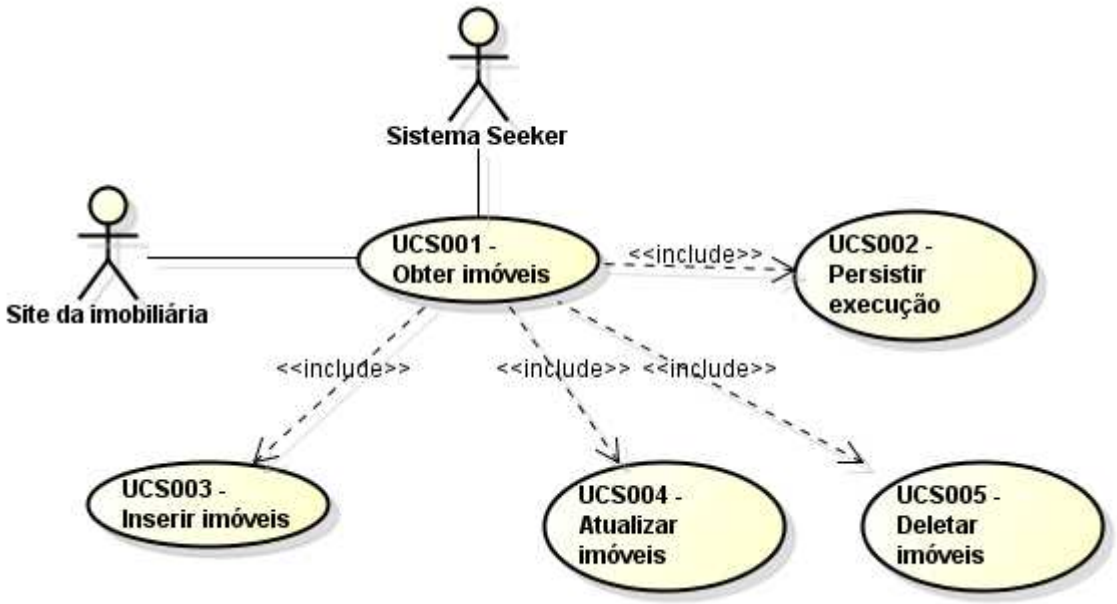


Figura 2.2 – Diagrama de caso de uso do módulo *Seeker*
Fonte: Próprio Autor (2013)

Este módulo acessará o site de cada imobiliária e obterá todos os imóveis cadastrados. O acesso será realizado baixando o código HTML das páginas e de acordo com a arquitetura dos sites das imobiliárias, será identificado como obter a informação de cada imóvel. Após este procedimento o imóvel será salvo no banco de dados relacional.

Este módulo rodará diariamente e o *script* para obter, inserir, atualizar e deletar os imóveis será desenvolvido para cada site, de acordo com a arquitetura já analisada anteriormente. O registro de *log* ficará armazenado no banco de dados NoSQL.

2.2.3 Módulo *Job*

O módulo *Job* analisa e cruza os dados armazenados pelos dois módulos anteriores, sendo executado a cada 10 minutos. Este tempo foi definido para que caso seja cadastrado um novo imóvel manualmente, o sistema gere em menos de 20 minutos, a sugestão do imóvel para os usuários. Na figura 2.3 é apresentado o diagrama dos casos de uso deste módulo.

O módulo *Job* cria, atualiza, deleta e classifica as sugestões de imóveis para os usuários, armazenando o resultado no banco NoSQL. As sugestões são criadas de acordo com o cruzamento do perfil do usuário com os imóveis ativos no sistema, e essas informações são obtidas no banco de dados relacional.

Após gerar as sugestões, o *Job* faz o envio de um e-mail com sugestões para os usuários, conforme a frequência cadastrada pelo usuário (diariamente, semanalmente ou

mensalmente). Depois essas sugestões podem ser visualizadas no módulo Web, como fora descrito anteriormente.

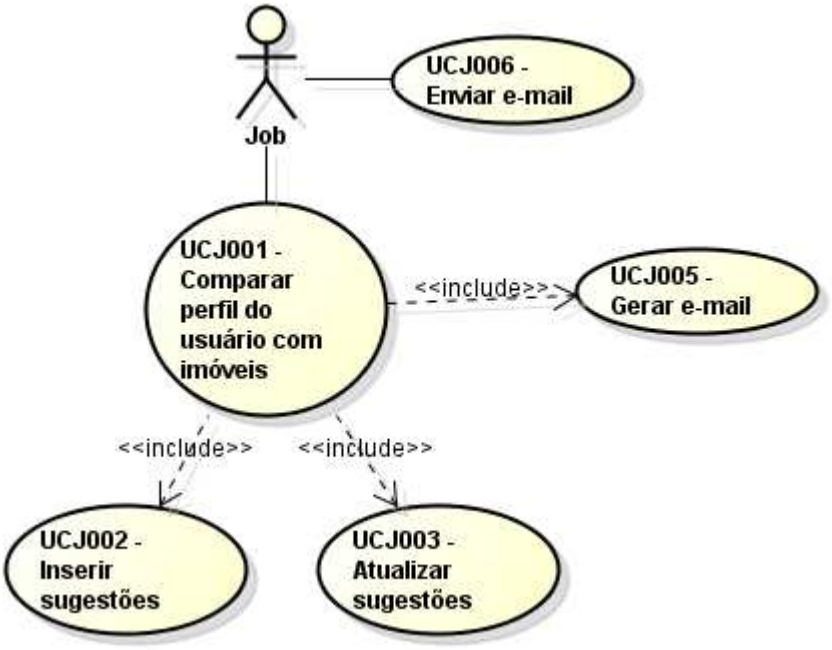


Figura 2.3 – Diagrama do módulo *Job*
Fonte: Próprio Autor (2013)

2.2.4 Integração

Os módulos são integrados pelos dados armazenados nos bancos de dados. Para facilitar a visualização da comunicação e o acesso de cada módulo a cada banco de dados criou-se a figura 2.4.

Deste modo, as três aplicações estão integradas de acordo com a necessidade das funcionalidades. Dividir o sistema em módulos oferece a vantagem de melhoria do desempenho, pois cada módulo pode ser instalado em servidores distintos, ou seja, um sistema distribuído, sendo necessário somente todos acessarem os mesmos bancos de dados.

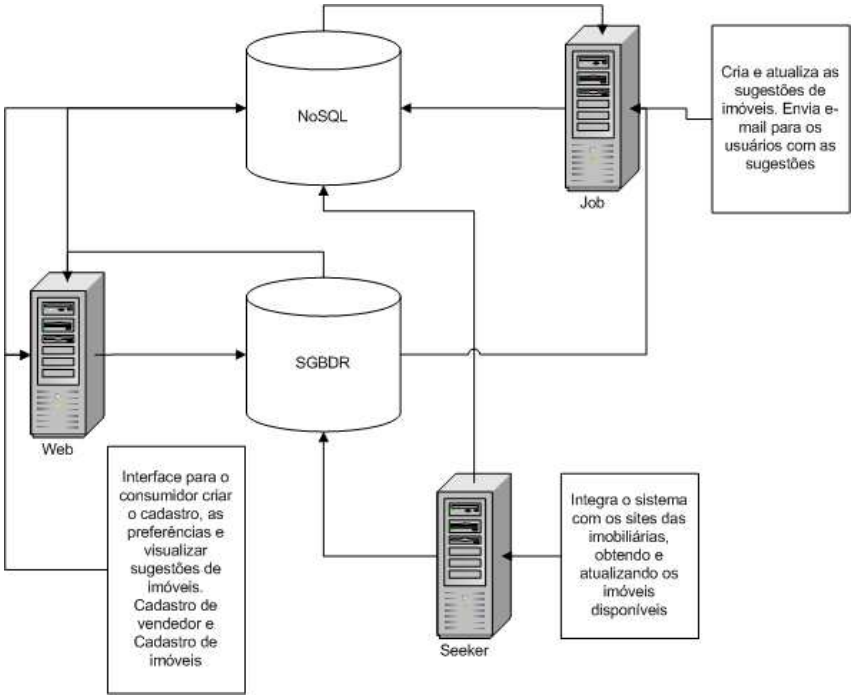


Figura 2.4 – Integrações dos módulos do sistema
Fonte: Próprio Autor (2013)

2.2.5 Hospedagem

Atualmente, o sistema está hospedado em quatro servidores distintos, divididos da seguinte forma: um servidor com o módulo Web, um servidor com o módulo Seeker e o módulo Job, um servidor com o banco MySQL e um servidor com o banco NoSQL. A figura 2.5 ilustra a divisão.

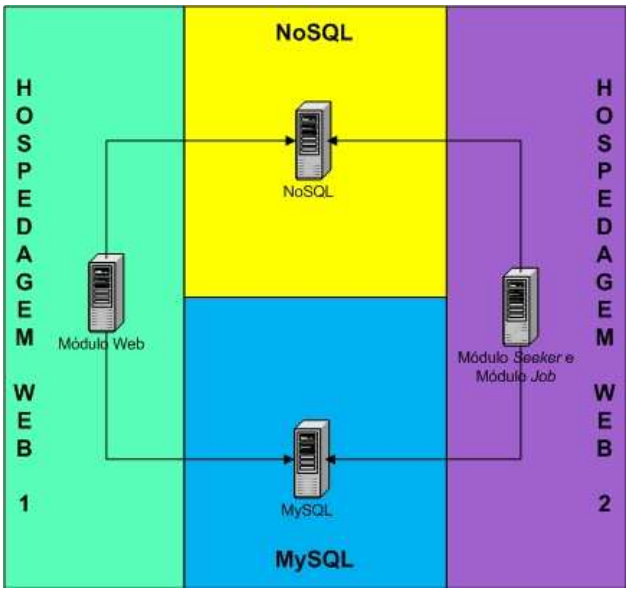


Figura 2.5 – Divisão do sistema por servidor
Fonte: Próprio Autor (2013)

Os módulos Web, *Job* e *Seeker* foram divididos em dois servidores Web distintos, foi necessária esta distribuição devido à utilização de hospedagens Web compartilhadas. A hospedagem compartilhada é um servidor com vários sites hospedados nele, por conta disso a utilização dos recursos do servidor são limitadas para cada site e as aplicações podem ser finalizadas cada vez que ocorre o estouro de memória ou o alto uso do processador.

Diante deste cenário, foi preciso tomar alguns cuidados no desenvolvimento dos módulos. No módulo Web foi preciso armazenar os dados das sessões dos usuários no banco de dados MySQL ao invés de salvar estes dados na memória do servidor web. O tempo da sessão é de 20 minutos sem interações com o site, ao término deste tempo a sessão é deletada do banco. Já no módulo *Seeker*, foi preciso manualmente persistir cada ação executada, armazenando a informação no MySQL, com isso caso a aplicação seja finalizada é possível reiniciar a integração do local onde estava.

Por fim, quando uma aplicação é finalizada em uma hospedagem compartilhada ela só é iniciada quando o site for acessado, por conta dos módulos *Job* e *Seeker* estarem em um servidor não publicado ao usuário é preciso que a aplicação seja iniciada logo após ser finalizada, pois esses módulos executam tarefas repetitivas vezes em um espaço de tempo. Para solucionar este problema, foi contratado um *CronJob*, que executa o acesso a um site desejado em um determinado período. Sendo assim, a cada 5 minutos essa tarefa acessa a aplicação, fazendo com que ela seja iniciada.

No próximo capítulo serão analisados os bancos de dados NoSQL e define-se qual desses bancos que este trabalho utiliza.

3 NOSQL (NOT ONLY SQL)

Com o crescimento do uso de aplicações Web, os bancos de dados relacionais não atendiam requisitos de manipulação de grande quantidade de dados, com disponibilidade e escalabilidade. Desta forma, começaram a serem desenvolvidas outras aplicações de banco de dados, que atendiam aos novos requisitos trabalhando com grande volume de dados não estruturados ou semi-estruturados. Estas novas soluções não relacionais criadas foram chamadas de NoSQL (*Not Only SQL* – Não Apenas SQL) (LÓSCIO & OLIVEIRA & PONTES, 2011).

3.1 Escalabilidade

Um sistema com escalabilidade é uma solução que, à medida que aumenta a utilização do *software* não perde desempenho. Para tornar um sistema escalável é possível utilizar a escalabilidade vertical ou a horizontal.

Ferreira (2010) explica que a escalabidade vertical é o aumento da capacidade de processamento e de memória dos computadores. Já a escalabilidade horizontal é o aumento do número de computadores disponíveis para processar e armazenar informações. Diante disso, pode-se concluir que a escalabilidade vertical tende chegar a um ponto em que não seja mais possível crescer, sendo assim a melhor técnica, a escalabilidade horizontal.

Em banco de dados relacionais a escalabilidade horizontal pode se tornar inviável. Isto porque vários processadores executando uma informação aumentam a concorrência, influenciando no tempo de acesso à tabela relacional (FERREIRA, 2010).

Por outro lado, conforme afirma Ferreira (2010), NoSQL oferece como característica principal o uso da escalabilidade horizontal. Por não oferecer bloqueios no acesso concorrente aos dados, o que reduz a consistência, faz com que o tempo de retorno seja mais rápido do que em SGBD relacionais. Além de, à medida que o sistema aumente, é possível sempre aumentar a capacidade de processamento do sistema.

De acordo com o teorema CAP (Consistência, Disponibilidade e Tolerância ao Particionamento), desenvolvido por Eric Brewer em 2000, um banco de dados distribuído só pode ter duas das três características do CAP. Harrison (2010, tradução nossa) apresenta essas características da seguinte forma:

> “Consistência: Todos os nós da rede tem exatamente o mesmo dado em qualquer momento;

> Disponibilidade: A falha em um nó da rede não faz com que o banco de dados fique inoperante;
> Tolerância ao Particionamento: Os nós da rede continuam funcionando quando a conexão com outro grupo de nós é perdida.”

Na figura 3.1 é apresentada a forma de relação do teorema do CAP e onde as aplicações são executadas. Verifica-se que as aplicações NoSQL oferecem Disponibilidade e Tolerância ao Particionamento.

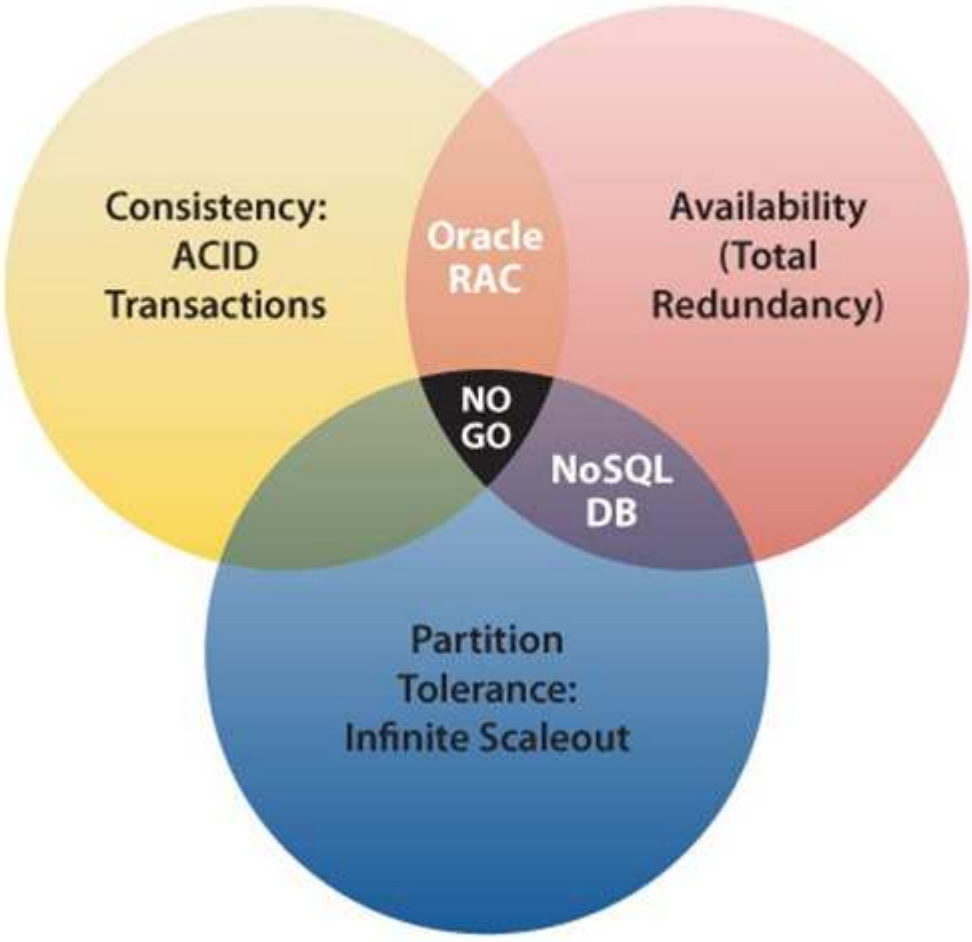


Figura 3.1 – Diagrama do Teorema CAP
Fonte: Consistency models in Non relational Databases (HARRISON, 2010)

Segundo Vogels (2008), a consistência dos dados não é o fator primordial em bancos de dados NoSQL, ao contrário dos bancos relacionais. Entretanto, a solução oferece alta disponibilidade e escalabilidade. Isto não significa que não existe consistência em bancos não relacionais, o que ocorre é que os bancos de dados NoSQL trabalham para tolerar inconsistências temporárias, fornecendo alta disponibilidade.

3.2 Técnicas

Para atender as características dos bancos de dados NoSQL, apresentadas na seção anterior, existem diversas técnicas utilizadas. A seguir serão descritas as principais técnicas utilizadas e seu funcionamento.

3.2.1 Consistent hashing

Conforme citado anteriormente, à medida que a base de dados NoSQL aumenta é necessário incluir novos computadores para processar as informações sem perder desempenho. Para Shalom (2009) são necessários algoritmos que permitam o crescimento dos *data cluster* sem ser necessário desativar ou forçar uma repartição dos dados, para isso são utilizados algoritmos que implementam o *consistent hashing*. Portanto, o *consistent hashing* é utilizado para migrar e recuperar dados entre os nós da rede, alocando e desalocando os dados distribuídos.

De acordo com Shalom (2009, tradução nossa) podem existir vários algoritmos de *consistent hashing* que operam de modos diferentes:

> "[...] Um algoritmo notifica o vizinho sobre uma partição que se conectou ou caiu. Somente os nós vizinhos são impactados pela mudança e não todos os nós da rede. Durante o período de redistribuição dos dados entre os computadores já existentes e os recém adicionados existe um protocolo para tratar essa transação.
> Outro algoritmo bastante significativo e mais simples utiliza partições lógicas. Com partições lógicas, o número de partições é fixo, mas a distribuição de partições entre as máquinas são dinâmicas. Por exemplo, se você inicia com duas máquinas e 1000 partições lógicas, você tem 500 partições lógicas por máquina. Quando você adicionar uma terceira máquina, você terá 333 partições por máquina [...]."

3.2.2 Vector clocks

Vector clocks são algoritmos criados para aumentar a consistência de dados em bancos NoSQL. Cada nó do sistema distribuído mantém um contador de mudanças (*change number*). O vetor armazena os dados do contador do nó atual, assim como o *change number* dos outros nós. Então, quando uma atualização de um campo é realizada o *change number* é transmitido para os outros nós do sistema os nós ao receber o dado compara com o vetor atual e verifica se é necessário atualizar o valor ou se o dado tem um registro mais recente (HARRISON, 2010).

Sadalage e Fowler (2013, tradução nossa), que utilizam o termo *Vector Stamp* ao se referenciar à *Vector Clocks*, sintetizam o significado desta técnica afirmando que: "[...] é um

conjunto de contadores, um para cada nó. [...] ao atualizar um registro, o contador é atualizado no nó e sincronizado os vetores com os outros nós [...].".

3.2.3 Map/reduce

O *Map/reduce* é uma técnica para dividir grandes problemas em pequenos problemas, distribuindo estes para dispositivos que resolverão e retornarão a solução. É um padrão para organizar o processamento em diversas máquinas (SADALAGE & FOWLER, 2013).

Para exemplificar esta funcionalidade, Sadalage e Fowler (2013) apresentam a situação de existir um sistema distribuído, onde os dados ficam armazenados em diversas máquinas. Os dados armazenados são de ordens de venda e cada item da ordem é armazenado junto com ela, pois é comum no acesso de uma ordem querem ver ela por completa. No entanto, quando for necessário tirar um relatório dos produtos com o rendimento total de cada um nas ordens de venda de um determinado período, esses dados não estarão agrupados de forma estruturada. Então, é necessário utilizar o *Map/reduce*. A primeira etapa executada é o *Map*, em que é enviado o identificador da ordem de venda e são retornados os valores referentes aos itens da ordem. Ao final do *Map* é executada a função *Reduce*, em que todas as saídas são agrupadas, de acordo com o identificador, em um único registro, combinando os outros valores.

3.2.4 MVCC (*Multiversion Concurrency Control*)

Por padrão os bancos de dados relacionais ao executar uma transação de escrita utilizam a funcionalidade de bloquear tarefas enquanto a escrita é feita, com o objetivo de ter consistência. O MVCC permite a concorrência dos dados de forma otimizada, sendo assim nenhuma tarefa ou consulta é bloqueada, pois cada transação tem uma cópia dos objetos do banco (GRAVES, 2010).

Quando se utiliza o MVCC pode ocorrer de um dado no mesmo instante de tempo ser atualizado, então ocorre um conflito e uma tarefa será realizada novamente. No entanto, conforme afirma Graves (2010), do ponto de vista de desempenho, é melhor ocasionalmente executar uma tarefa novamente do que sempre bloquear tarefas enquanto executa outra.

3.3 Modelos

Os bancos de dados NoSQL são diferenciados por quatro principais tipos e a utilização de cada tipo varia de acordo a aplicação desejada. Atualmente, de acordo com o site nosql-database (2013) existem 150 diferentes bancos de dados NoSQL.

3.3.1 Armazenamento orientado a Chave-Valor

O armazenamento orientado a *Key-Value* (Chave-Valor) é uma tabela de *hash* e todos os acessos à tabela são realizados através de uma chave primária (ALMEIDA & BRITO, 2012), conforme ilustra a figura 3.2. Neste modelo podemos fazer uma analogia com o banco de dados relacional, no qual existe uma tabela com duas colunas "IDENTIFICADOR" e "DESCRICAO", onde a coluna "identificador" armazena a chave e a coluna "descricao" armazena o dado.

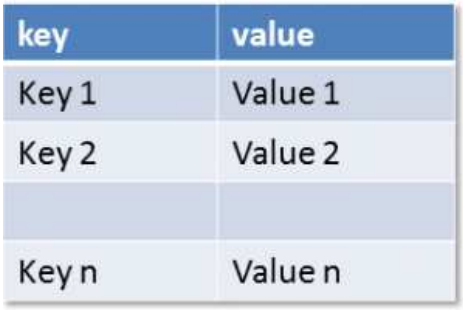


Figura 3.2 – Exemplo de armazenamento orientado a Chave-Valor
Fonte: NoSql-like approach for Gideros offline save game files (KHANH, 2012)

Lóscio, Oliveira e Pontes (2011, p. 6) explicam que este modelo é de fácil implementação e contribui para a alta disponibilidade, e ainda complementam detalhando sobre as operações para manipulação de dados:

> "Este modelo, por ser de fácil implementação, permite que os dados sejam rapidamente acessados pela chave, principalmente em sistemas que possuem alta escalabilidade, contribuindo também para aumentar a disponibilidade de acesso aos dados. As operações disponíveis para manipulação de dados são bem simples, como o get( ) e o set( ), que permitem retornar e capturar valores, respectivamente. A desvantagem deste modelo é que ele não permite a recuperação de objetos por meio de consultas mais complexas."

De acordo com o site nosql-database (2013) existem diversos sistemas deste tipo, como por exemplo:

- *DynamoDB*: Banco de dados NoSQL altamente escalável. Esse serviço é disponibilizado pela Amazon e os dados são salvos em rápidas unidades de estado sólido, oferece *Map/Reduce* e *Backup*;

- *Riak*: Acesso via *JSON* utilizando o protocolo *REST*, utiliza nas consultas *Map/Reduce*. Oferece consistência eventual através de *Vector Clocks* e alta escalabilidade.

3.3.2 Armazenamento orientado a Colunas

Este modelo é orientado a colunas ou atributos, variando do padrão anterior e dos modelos relacionais que são orientados a registros. Sendo assim, a variação se dá pelo fato dos dados serem indexados por uma tripla (linha, coluna e *timestamp*). A linha e a coluna são identificadas por uma chave e o *timestamp* permite diferenciar as versões de um mesmo dado (LÓSCIO et al., 2011, p. 6), a figura 3.3 ilustra esse modelo.

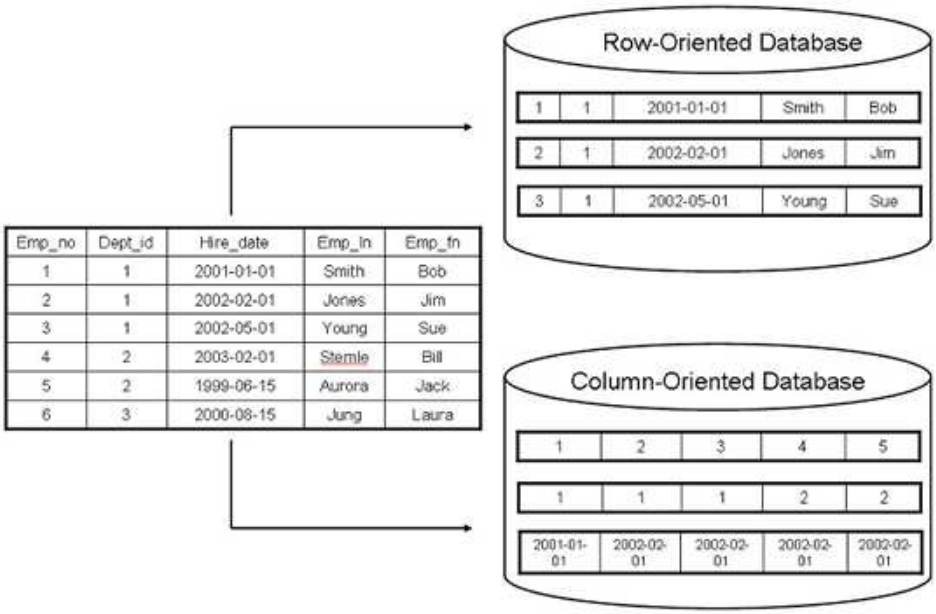


Figura 3.3 – Exemplo de armazenamento orientado a Colunas
Fonte: Column Oriented Database Technologies (ANDERSON, 2012)

Por armazenar os dados dessa forma, este modelo oferece vantagem somente para a leitura das informações onde se deseja obter algumas colunas de muitos registros, ou seja, deve ser utilizado quando é preciso fazer o processamento analítico online. Além disso, a escrita de um novo registro é mais custosa que um banco de dados relacional (DIANA & GEROSA, 2010).

O site nosql-database (2013) lista os bancos de dados NoSQL deste modelo, como por exemplo:

- *Hadoop*: Altamente recomendado para armazenar grandes quantidades de dados e processamento de informações, desenvolvido em Java, utiliza a técnica de *Map/Reduce*;
- *BigTable*: Desenvolvido pela empresa Google, foi o pioneiro deste modelo, oferece escalabilidade e é tolerante a falhas;
- *Cassandra*: Sistema desenvolvido pelo Facebook, baseado no *DynamoDB* da Amazon e o modelo de dados no *BigTable*, oferecendo assim escalabilidade e ótimo desempenho na leitura de dados.

3.3.3 Armazenamento orientado a Documento

Este modelo é orientado a documento, onde um documento contém um identificador e vários campos, que podem ser listas, documentos aninhados e textos. Sendo assim, não existe estrutura fixa nos dados, como há nos bancos relacionais. Este modelo é uma evolução do orientado a Chave-Valor, pois os documentos são armazenados em um conjunto e em cada documento tem um conjunto de chaves e o valor de cada chave, que é uma tabela *hash* (LÓSCIO et al., 2011, p. 7). A figura 3.4 ilustra esse modelo, onde os artigos são armazenados em um documento e dentro dele contem inúmeros comentários.

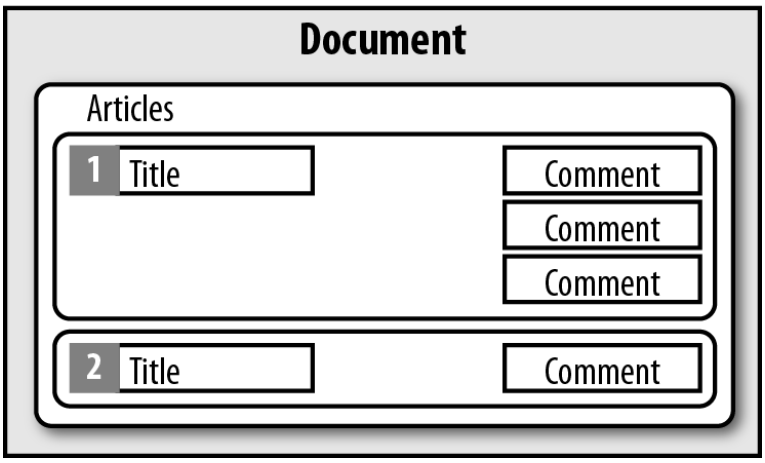


Figura 3.4 – Exemplo de armazenamento orientado a Documento
Fonte: Document Oriented Database (PEIRIS, 2013)

Neste modelo os dados são duplicados, ficando as informações de relacionamento salvas em cada documento. Sendo assim, a busca de informações é mais rápida do que o modelo relacional, pois não é necessário fazer junções entre tabelas (DIANA & GEROSA, 2010).

Como exemplo deste modelo, de acordo com o site nosql-database (2013), existem:

- *MongoDB*: Desenvolvido em C++, não tem esquemas, oferece alta disponibilidade e é baseado em documentos no formato *JSON*;
- *CouchDB*: Oferece acesso via protocolo *REST* retornando as informações no formato de *JSON*, ao buscar estas informações implementa a técnica de *Map/Reduce*.

3.3.4 Armazenamento orientado a Grafos

O modelo orientado a grafos é composto por três componentes básicos, que são explicados por Lóscio, Oliveira e Pontes (2011, p. 8), como complemento a figura 3.5 ilustra o comportamento deste tipo:

> "[...] os nós (são os vértices do grafo), os relacionamentos (são as arestas) e as propriedades (ou atributos) dos nós e relacionamentos. Neste caso, o banco de dados pode ser visto como um multigrafo rotulado e direcionado, onde cada par de nós pode ser conectado por mais de uma aresta. Uma consulta relevante para esta aplicação seria: "Quais cidades foram visitadas anteriormente (seja residindo ou viajando) por pessoas que viajaram para o Rio de Janeiro?". No modelo relacional esta consulta poderia ser muito complexa devido a necessidade de múltiplas junções, o que poderia acarretar uma diminuição no desempenho da aplicação. Porém, por meio dos relacionamentos inerentes aos grafos, estas consultas tornam-se mais simples e diretas."

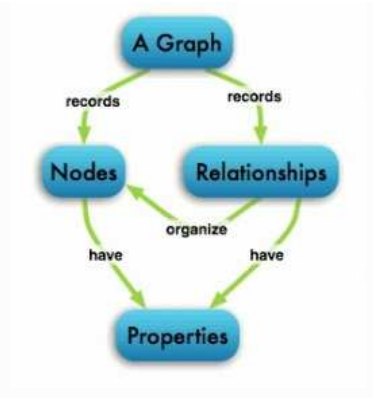


Figura 3.5 – Exemplo de armazenamento orientado a Grafos
Fonte: NoSQL, graph databases and Big Data (BUFITHIS, 2011)

De acordo com o site nosql-database (2013), existem diversos modelos, por exemplo:

- *Neo4J*: Desenvolvido em Java, utiliza um modelo de grafos nativo;
- *Infinite Graph*: Tem o *core* do sistema desenvolvido em C++ e o restante em Java, oferece consultas por navegação de grafos.

3.4 Definição do banco *NoSQL - DynamoDB*

Conforme visto na seção anterior, um modelo de banco de dados NoSQL, que oferece disponibilidade e escalabilidade é o *DynamoDB.* Este serviço de banco de dados oferecido pela Amazon, direcionando a gestão do banco, armazenamento e o provisionamento do hardware para a AWS (*Amazon Web Services*). A Amazon (2013) apresenta o produto da seguinte forma:

> "O *Amazon DynamoDB* é um serviço de banco de dados NoSQL de gestão completa que fornece um desempenho rápido e previsível com facilidade de escalabilidade. O *Amazon DynamoDB* permite aos clientes redirecionar as cargas administrativas de operação, assim como escalar bancos de dados distribuídos para a AWS. Dessa forma, os clientes ficam isentos de preocupações com provisionamento, instalação e configuração de hardware, replicação, patches de software ou escalabilidade de cluster..."

Diante dessas características oferecidas pela Amazon de alta disponibilidade, escalabilidade, facilidade na administração e um ótimo custo-benefício, este trabalho utiliza o *DynamoDB* como a ferramenta NoSQL para uso nas aplicações.

Todas as outras ferramentas estudadas traziam como principal desvantagem não oferecer facilidades em sua administração, pois não é um serviço prestado por uma empresa, o que é disponibilizado é somente o produto. Então, custos como aquisição de *hardware*, instalação e configurações deveriam ser levados em consideração.

Além disso, dentre os modelos estudados, o chave-valor é o que mais aderente a necessidade deste trabalho. Não sendo interessante utilizar, por exemplo, o armazenamento orientado a grafos, pois as informações a serem persistidas ficam dentro do padrão que é oferecido pelo *DynamoDB* (tabelas com uma chave e valores associados a esta chave).

3.4.1 Principais funcionalidades

O *DynamoDB* oferece diversas funcionalidades que resolvem os principais problemas nos bancos de dados e diminui o tempo utilizado para administrar o hardware. No decorrer deste capítulo serão apresentadas as principais funcionalidades de acordo com a Amazon (2013).

A escalabilidade é o grande forte deste sistema, que foi desenvolvido pensando nisso. O rendimento de cada tabela do banco é provisionado, então ao criar a tabela na AWS é especificada a capacidade de leitura e escrita requerida e o *DynamoDB* aloca recursos dedicados, de forma abstraída, para o desempenho desejado. Se o uso da aplicação aumenta

ou diminui, a capacidade de leitura e escrita no *DynamoDB* pode ser alterada facilmente utilizando o *AWS Management Console* ou a API de acesso ao *DynamoDB.* Além disso, não existem limites para a quantidade de dados armazenados e o serviço automaticamente aumenta mais memória à medida que for preciso. Para oferecer toda essa arquitetura, o banco de dados oferece escalabilidade horizontal, sendo totalmente distribuído, a figura 3.6 ilustra isso e apresenta que conforme aumenta a quantidade de usuários utilizando o sistema, os dados são distribuídos entre várias bases de dados.

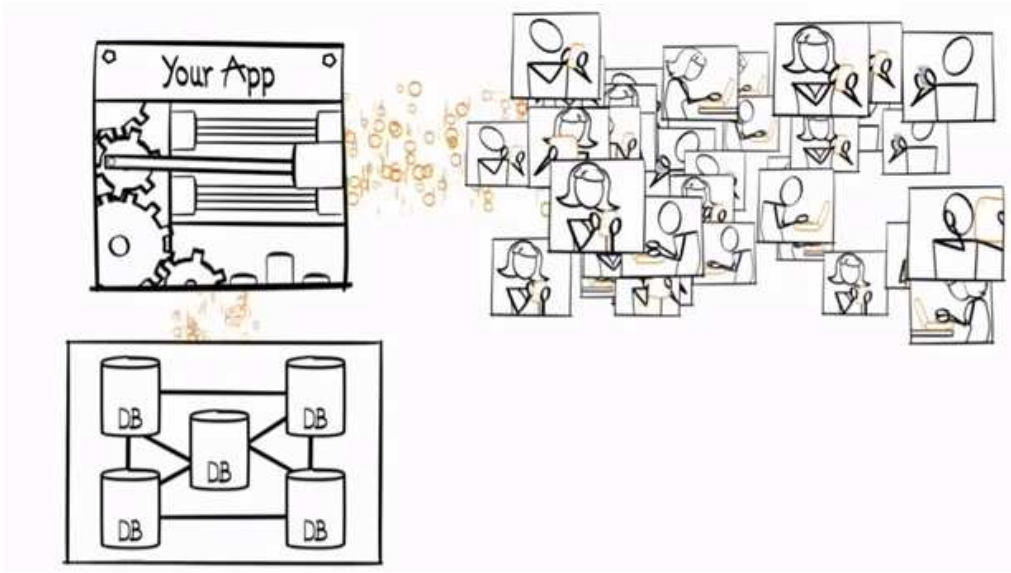


Figura 3.6 – Escalabilidade no *DynamoDB*
Fonte: Amazon DynamoDB Overview, a fully managed NoSQL database service (AWS, 2013)

O serviço é rápido e tem um desempenho previsível. A latência dos serviços nos servidores da Amazon é normalmente de poucos milissegundos, pois os dados são armazenados em discos de estado sólido. Com tudo isso, a administração dos recursos é facilitada. Só é preciso criar a tabela e deixar o serviço cuidar do restante, não é necessário se preocupar com o provisionamento de hardware ou software, instalações e configurações, atualizações de software, distribuições do banco de dados nos servidores ou particionamento dos dados.

O *DynamoDB* foi construído para ser tolerante a falhas, o sistema automaticamente sincroniza e replica os dados por regiões geograficamente distribuídas, oferecendo alta disponibilidade, conforme ilustra a figura 3.7. O banco ainda não tem um esquema fixo, desta forma cada item da tabela pode ter diferentes atributos de diferentes tipos (texto, binário, numérico), tornando o sistema altamente flexível.

Figura 3.7 – Distribuição dos dados por região no *DynamoDB*
Fonte: Amazon DynamoDB Overview, a fully managed NoSQL database service (AWS, 2013)

Cada item de uma tabela armazenada no *DynamoDB* contém uma chave primária, que permite acessar rapidamente e de modo eficiente o item. Além disso, o sistema oferece forte consistência, ao contrário da maioria dos bancos não relacionais, conforme afirma a AWS (2013, tradução nossa):

> "Diferente da maioria dos bancos não relacionais, o DynamoDB faz com que o desenvolvimento seja mais fácil, permitindo ter forte consistência na leitura dos dados para garantir que você sempre está lendo os valores mais recentes. [...] O serviço ainda suporta nativamente contadores atômicos, permitindo você atomicamente incrementar ou decrementar um atributo numérico com uma simples chamada na API."

Com tudo isso, o *DynamoDB* foi arquitetado para oferecer um grande custo-benefício independente do tamanho da aplicação que o utiliza, segurança por utilizar métodos criptografados para autenticar usuário e não permitir inserção de dados não autorizados.

3.4.2 Armazenamento dos dados

Os dados armazenados pelo *DynamoDB* são três réplicas distribuídas em espaços diferentes, permitindo maior disponibilidade e durabilidade dos dados. Ao fazer a leitura dos dados, a aplicação que consome pode escolher entre dois modos: Leitura eventualmente consistente e a Leitura fortemente consistente.

De acordo com a AWS (2013), a leitura eventualmente consistente otimiza a taxa de transferência de leitura, entretanto neste modo a leitura pode não refletir os resultados de uma gravação recentemente concluída. Normalmente, o tempo para concluir a gravação nas três réplicas é de um segundo, então repetindo a leitura depois de um curto espaço de tempo retornará os dados atualizados. Já a leitura fortemente consistente oferece uma leitura consistente retornando os dados que receberam uma resposta bem-sucedida antes da leitura ser executada.

Os dados são armazenados em discos de estado sólido e os motivos disto, de acordo com a AWS (2013), são:

> "O Amazon DynamoDB é executado exclusivamente em discos de estado sólido (SSDs). Os SSDs ajudam a atingir objetivos em projetos de momentos de resposta previsível de baixa latência para armazenar e acessar dados em qualquer dimensão. O alto desempenho de E/S dos SSDs também nos permite atender a solicitações de grandes cargas de trabalho com boa relação custo-benefício [...]."

3.4.3 Modelo de banco de dados

De acordo com a AWS (2013), o *DynamoDB* é um modelo de banco de dados do tipo Chave-Valor e por isso suporta as operações de *GET* (obter) e *SET* (definir) usando a chave primária definida na criação da tabela. A chave primária é a o único campo obrigatório para os itens de uma tabela e também é um valor único, que não se repete. Uma base de dados é composta por um conjunto de tabelas, que tem uma coleção de itens e cada item tem uma coleção de atributos.

Ao criar a tabela é possível definir que a chave primária será somente um atributo simples ou um atributo simples e uma faixa de valor (*hash-range key*). Por exemplo, ao definir a chave primária como *hash-range,* um atributo poderia ser o identificador (*hash*) e o outro uma variação de horário (*range*). Ao utilizar este tipo de chave primária é possível utilizar as consultas da API do AWS, que retornará os valores para um determinado identificador de uma determinada faixa de tempo.

Este banco de dados é NoSQL e portanto não tem uma estrutura, sendo somente obrigatório o preenchimento da chave primária. Os outros atributos podem variar de acordo com o item adicionado, podendo armazenar somente uma informação ou várias no mesmo atributo. Por exemplo, em um cadastro de uma pessoa é possível adicionar quando necessário um atributo como CPF para pessoas físicas e quando for uma pessoa jurídica adicionar o

atributo CNPJ, além disso seguindo este exemplo, uma pessoa pode ter mais do que um telefone, portanto é possível armazenar todos os telefones em um único atributo.

3.4.4 Operações

Ao desenvolver com o *DynamoDB* existem diversas operações que são utilizadas para o controle de tabelas e itens e para executar consultas. O sistema oferece operações para criação, atualização e exclusão de tabelas. Depois de criar a tabela deve ser utilizada a operação *UpdateTable* para executar alterações do provisionamento de utilização da tabela. Além disso, é possível obter as informações da tabela utilizando *DescribeTable* e obter as lista de tabelas na conta na AWS pela operação *ListTables*.

As operações de itens servem para adicionar, atualizar e deletar os itens da tabela. Para atualizar um atributo, adicionar novo atributo ou deletar um é preciso utilizar a operação *UpdateItem*. É possível também criar atualizações condicionais para que somente seja executada a atualização de um atributo, se este estiver de acordo com a condição.

Para obter itens das tabelas é necessário utilizar a operação *Query* ou *Scan*. Na primeira é necessário utilizar a chave primária na consulta e um filtro opcional da faixa de um valor. Só é possível utilizar essa operação se a chave primária é do tipo *hash-range*. Esta também é a operação mais eficiente para retornar os itens de uma tabela. Já a operação *Scan* deve ser utilizada somente quando necessário, pois ela examina cada item na tabela e o resultado é limitado em 1 MB depois de filtrar os dados (AWS, 2013).

3.4.5 Provisionamento da taxa de transferência

Ao criar ou atualizar uma tabela no *DynamoDB* é possível especificar o provisionamento da taxa de transferência na leitura e na escrita das requisições ao banco de dados, que irá automaticamente reservar os recursos necessários para garantir a disponibilidade e o alto desempenho do sistema, é por esta taxa e pelo tamanho do banco de dados que a AWS executa a cobrança do serviço.

Uma unidade na capacidade de leitura (*read capacity*) representa uma leitura consistente por segundo (ou duas leituras eventualmente consistente por segundo) para itens com tamanho de até 4 KB, quando ocorrer do tamanho ser maior, mais do que uma unidade da capacidade será utilizada. Uma unidade na capacidade de escrita (*write capacity*) representa uma escrita de 1 KB por segundo.

3.4.6 Acessando o *DynamoDB* com *.NET*

O *DynamoDB* é um *WebService* que ao ser consumido retorna os dados no formato *JSON* (*JavaScript Object Notation*), oferecendo para as aplicações a utilização de API para acessar os dados. A AWS disponibiliza em seu site[1] um kit de desenvolvimento para *.NET*.

Na figura 3.8 é apresentado um exemplo de criação de uma tabela (*ProductCatalog*), que tem somente a chave primária (*Id*) com capacidade de leitura de 10 unidades e de escrita de 5 unidades.

```
client = new AmazonDynamoDBClient(credentials);
string tableName = "ProductCatalog";

var request = new CreateTableRequest
{
  TableName = tableName,
  AttributeDefinitions = new List<AttributeDefinition>()
  {
    new AttributeDefinition
    {
      AttributeName = "Id",
      AttributeType = "N"
    }
  },
  KeySchema = new List<KeySchemaElement>()
  {
    new KeySchemaElement
    {
      AttributeName = "Id",
      KeyType = "HASH"
    }
  },
  ProvisionedThroughput = new ProvisionedThroughput
  {
    ReadCapacityUnits = 10,
    WriteCapacityUnits = 5
  }
};

var response = client.CreateTable(request);
```

Figura 3.8 – Criação de tabela no *DynamoDB*
Fonte: Working with Tables Using the AWS SDK for .NET Low-Level API (AWS, 2013)

Na figura 3.9 é apresentado um exemplo de inclusão de um item do tipo livro na tabela de catálogo de produtos. Aqui é possível comprovar que a tabela é sem esquema, podendo ser adicionado novos atributos a qualquer momento.

[1] http://aws.amazon.com/sdkfornet/

```
client = new AmazonDynamoDBClient(credentials);
string tableName = "ProductCatalog";

var request = new PutItemRequest
{
   TableName = tableName,
   Item = new Dictionary<string, AttributeValue>()
      {
          { "Id", new AttributeValue { N = "201" }},
          { "Title", new AttributeValue { S = "Book 201 Title" }},
          { "ISBN", new AttributeValue { S = "11-11-11-11" }},
          { "Price", new AttributeValue { S = "20.00" }},
          {
            "Authors",
            new AttributeValue
            { SS = new List<string>{"Author1", "Author2"}   }
          }
      }
};
client.PutItem(request);
```

Figura 3.9 – Inclusão de itens no *DynamoDB*
Fonte: Working with Items Using the AWS SDK for .NET Low-Level API (AWS, 2013)

Na figura 3.10 é apresentado um exemplo de atualização de um item do tipo livro na tabela de catálogo de produtos obtido utilizando a chave primária, onde são adicionados dois novos autores, o valor do item é reduzido em 1 unidade, é adicionado um novo atributo e por fim deletado um atributo existente.

```
client = new AmazonDynamoDBClient(credentials);
string tableName = "ProductCatalog";

var request = new UpdateItemRequest
{
   TableName = tableName,
   Key = new Dictionary<string,AttributeValue>() { { "Id", new AttributeValue { N = "202" } } },
   AttributeUpdates = new Dictionary<string, AttributeValueUpdate>()
   {
     // Add two new authors to the list.
     { "Authors",
       new AttributeValueUpdate
          {
              Action="ADD",
              Value = new AttributeValue{SS = { "Author YY", "Author ZZ" }}
           }
      },
      // Reduce the price. To add or subtract a value,
      // use ADD with a positive or negative number.
     { "Price",
       new AttributeValueUpdate
          {
              Action="ADD",
              Value = new AttributeValue{N = "-1"}
           }
      },
      // Add a new attribute.
     { "NewAttribute",
       new AttributeValueUpdate  { Value = new AttributeValue{S = "someValue" } } },
      // Delete the existing ISBN attribute.
     { "ISBN", new AttributeValueUpdate  { Action="DELETE" } }
   }
};
var response = client.UpdateItem(request);
```

Figura 3.10 – Atualização de itens no *DynamoDB*
Fonte: Working with Items Using the AWS SDK for .NET Low-Level API (AWS, 2013)

Na figura 3.11 é utilizado a operação de *Query* do *DynamoDB* para obter todos os itens com o assunto especificado.

```
var request = new QueryRequest
{
  TableName = "Reply",
  KeyConditions = new Dictionary<string,Condition>()
  {
    {
      "Id",
      new Condition()
      {
        ComparisonOperator = "EQ",
        AttributeValueList = new List<AttributeValue>()
        {
          new AttributeValue { S = "Amazon DynamoDB#DynamoDB Thread 1" }
        }
      }
    }
  }
};

var response = client.Query(request);
var result = response.QueryResult;

foreach (Dictionary<string, AttributeValue> item in response.QueryResult.Items)
{
  // Process the result.
  PrintItem(item);
}
```

Figura 3.11– Consulta de itens no *DynamoDB - Query*
Fonte: Querying Tables Using the AWS SDK for .NET Low-Level API (AWS, 2013)

Na figura 3.12 é apresentado o uso da operação de *Scan* do *DynamoDB* para obter todos os itens com o valor menor que zero.

```
var forumScanRequest = new ScanRequest
 {
   TableName = "ProductCatalog",
   // Optional parameters.
   ScanFilter = new Dictionary<string, Condition>()
           {
               { "Price", new Condition
                   {
                      ComparisonOperator = "LT",
                       AttributeValueList = new List<AttributeValue>()
                       {
                           new AttributeValue { N = 0 }
                       }
                   }
               }
           },
   AttributesToGet = new List<string> { "Id" }
 };
```

Figura 3.12 – Consulta de itens no *DynamoDB- Scan*
Fonte: Scanning Tables Using the AWS SDK for .NET Low-Level API (AWS, 2013)

Sendo assim, o *DynamoDB* oferece escalabilidade, alta disponibilidade e segurança no armazenamento dos dados, além disso oferece um ótimo custo-benefício e portanto é o escolhido para ser utilizado neste trabalho. Ainda, com essa API e com os exemplos acima é possível desenvolver as funcionalidades do sistema que utilizam o banco NoSQL com a linguagem *.NET*.

4 DESENVOLVIMENTO

Após definições das tecnologias e imobiliárias envolvidas, iniciou-se o desenvolvimento do produto. O texto a seguir detalha a especificação dos requisitos funcionais e não-funcionais, os casos de uso, o diagrama do modelo relacional e a escrita do código, além de mostrar o resultado final.

4.1 Especificação dos Requisitos

No capítulo 2 foram descritas as funcionalidades do sistema, de modo a caracterizar como a ferramenta deveria ser construída. Neste momento serão especificados os requisitos funcionais e os não funcionais de cada módulo. De acordo com Ávila e Spínola (2007) os requisitos funcionais são características diretamente ligadas ao programa de computador, enquanto os requisitos não funcionais definem o que o programa deve atender ou que qualidade deve ter.

Em seguida, são apresentados os casos de uso dos diagramas presentes no capítulo 2, de maneira a ficarem detalhadas as funcionalidades do sistema e como os requisitos funcionais serão atingidos. Por uma questão de organização, nesta seção serão exibidos somente os principais casos de uso de cada módulo.

4.1.1 Módulo Web

As tabelas 4.1 e 4.2 apresentam os requisitos funcionais e não-funcionais, respectivamente, do módulo Web.

Tabela 4.1 – Requisitos funcionais do Módulo Web

Identificador	Descrição
RFW1	O sistema deve ter um mecanismo de autenticar os usuários
RFW2	O sistema deve permitir o cadastro e a edição de usuários
RFW3	O sistema deve permitir o cadastro, a edição e a exclusão das preferências de imóveis dos usuários consumidores
RFW4	O sistema deve permitir a visualização das sugestões geradas pelo Módulo *Job* para o usuário autenticado
RFW5	O sistema deve permitir ordenar as sugestões pela mais recente e o mais recomentado
RFW6	O sistema deve permitir somente para usuários autenticados visualizar as informações dos imóveis
RFW7	O sistema deve permitir o cadastro de imóveis
RFW8	O sistema deve permitir a exclusão do imóvel cadastrado
RFW9	O sistema deve permitir a exclusão das sugestões

Fonte: Próprio Autor. 2013.

Tabela 4.2 – Requisitos não funcionais do Módulo Web

Identificador	Descrição
RNFW1	O módulo deve ser compatível com os principais navegadores do mercado (Chrome / Firefox / IE8 / IE9 / IE10)
RNFW2	O sistema deve manter o usuário autenticado durante os próximos 20 minutos da última requisição
RNFW3	O sistema deve ser hospedado em uma hospedagem Windows com *.NET Framework 4* instalado
RNFW4	Utilização de Banco de Dados Relacional: MySQL
RNFW5	Utilização de Banco de Dados NoSQL: DynamoDB

Fonte: Próprio Autor. 2013.

No quadro 4.1 é descrito o caso de uso do cadastro do consumidor. Este apresenta o fluxo principal e alternativos das etapas entre o usuário do tipo consumidor e o sistema, onde basicamente o usuário preenche seus dados e aceita os termos de uso.

Quadro 4.1 – Caso de uso Cadastrar consumidor

UCW001 – Cadastrar consumidor	
Descrição	Na página de cadastro de usuário consumidor tem que ser informado o nome, telefone, e-mail, senha, confirmação de senha, frequência com que deseja que os e-mails de aviso sejam enviados (diariamente, semanalmente ou mensalmente), preenchimento do UCW002 e aceitação dos termos de uso do site.
Requisitos	RFW2 RNFW1 RNFW3 RNFW4
Ator	Usuário consumidor, Módulo *Job*
Fluxo Principal	- Usuário acessa a página para um novo cadastro - Usuário preenche todos os campos corretamente e clica em finalizar - Sistema cadastra o novo usuário - Sistema *Job* é disparado automaticamente para comparar o perfil do usuário com os imóveis e gerar sugestões (UCJ001) - Sistema autentica o novo usuário e o redireciona para o UCW003
Fluxo Alternativo 1	- Usuário autenticado clica para alterar o seu cadastro - Usuário preenche todos os campos corretamente e clica em finalizar

	- Sistema atualiza o cadastro do usuário - Sistema redireciona o usuário para o UCW003
Fluxo Alternativo 2	- Usuário não aceita os termos de uso - Sistema não permite que o usuário clique no botão finalizar
Fluxo Alternativo 3	- Usuário não preenche qualquer um dos campos e clica em finalizar - Sistema exibe ao lado do campo a mensagem “Campo Obrigatório” na cor vermelha
Fluxo Alternativo 4	- Usuário preenche com valores diferentes o campo senha e confirmação de senha - Sistema exibe ao lado do campo a mensagem “Senhas não conferem” na cor vermelha
Fluxo Alternativo 5	- Usuário preenche um e-mail que já está cadastrado no sistema e clica em finalizar - Sistema exibe ao lado do campo a mensagem “E-mail já cadastrado” na cor vermelha
Fluxo Alternativo 6	- Usuário não preenche o UCW002 - Sistema exibe ao lado a mensagem “Adicione pelo menos um item” na cor vermelha
Fluxo Alternativo 7	- Usuário preenche o telefone com letras - Sistema não permite a inclusão de letras
Pós-condição	Os dados do usuário estão armazenadas no bancos de dados relacional

Fonte: Próprio Autor. 2013.

O quadro 4.2 apresenta o caso de uso do cadastro das preferências por imóveis, no qual o usuário preenche as características dos imóveis que ele procura, como a localização, faixa de valor, tipo de localização e tipo do imóvel.

Quadro 4.2 – Caso de uso Cadastrar as preferências por imóveis

UCW002 - Cadastrar as preferências por imóveis	
Descrição	Este caso de uso está inserido dentro do UCW001. O usuário deve preencher os campos a seguir obrigatoriamente: tipo de negociação (alugar ou comprar), tipo do imóvel (Apartamento, Área de Terra, Casa, Casa Comercial, Casa Em Condomínio, Chácara/Sítio, Cobertura, Garagem/Box, Loja, Outro, Prédio, Sala Comercial, Terreno ou Terreno Em Condomínio), faixa de valor, estado, cidade e bairros da cidade selecionada. Opcionalmente, o usuário pode preencher as informações adicionais de número de quartos, garagens e banheiros do imóvel a pesquisar. Para cada informação adicional deve ser colocado um valor e selecionar se deseja que o imóvel a ser encontrado tenha um valor superior, superior ou igual, igual, inferior ou igual, inferior ao valor informado. Ao terminar o preenchimento e clicado o botão adicionar, o item é exibido numa listagem e o usuário pode incluir quantos itens desejar.
Requisitos	RFW3 RNFW1 RNFW3 RNFW4
Ator	Usuário consumidor
Fluxo Principal	- Novo usuário preenche todos os campos corretamente e clica em adicionar - Sistema armazena a informação - Sistema aguarda o usuário clicar em Finalizar no UCW001 - Sistema gera as primeiras sugestões e envia um e-mail com elas
Fluxo Alternativo 1	- Usuário autenticado clica para alterar o seu cadastro - Usuário preenche todos os campos corretamente e clica em finalizar - Sistema cadastra a nova preferência armazenando no banco de dados
Fluxo Alternativo 2	- Usuário autenticado clica para alterar o seu cadastro

	- Usuário seleciona um item na listagem de preferência para editar - Usuário altera os campos corretamente e clica em finalizar - Sistema atualiza a nova preferência armazenando no banco de dados
Fluxo Alternativo 3	- Usuário não preenche qualquer um dos campos obrigatório e clica em finalizar - Sistema exibe ao lado do campo a mensagem “Campo Obrigatório” na cor vermelha
Fluxo Alternativo 4	- Usuário preenche a faixa de valor de forma incorreta (os dois valores iguais a zero ou o valor final menor que o valor inicial) - Sistema exibe ao lado do campo a mensagem “Verifique os valores” na cor vermelha
Fluxo Alternativo 5	- Usuário preenche a faixa de valor com letras - Sistema não permite a inclusão de letras
Fluxo Alternativo 6	- Usuário seleciona na listagem de preferências um item e clica em excluir - Sistema questiona se o usuário tem certeza que deseja excluir o item - Se confirmado, sistema exclui o item, caso contrário o item permanece inalterado
Fluxo Alternativo 7	- Usuário seleciona na listagem uma cidade que o sistema não tem imóveis cadastrados - Sistema exibe ao lado do campo a mensagem “Atenção: Não temos imóveis neste momento para esta cidade” na cor vermelha
Pós-condição	As preferências do usuário estão armazenadas no MySQL

Fonte: Próprio Autor. 2013.

Os demais casos de uso de visualizar sugestões, visualizar imóveis, autenticar, cadastrar vendedor, cadastrar imóveis, deletar imóveis e deletar sugestões são apresentados no

apêndice B. Conclui-se assim a especificação de requisitos do módulo Web, documentando-o para o desenvolvimento deste módulo.

4.1.2 Módulo *Seeker*

Nas tabelas 4.3 e 4.4 são apresentadas as tabelas com os requisitos funcionais e não-funcionais do módulo *Seeker*.

Tabela 4.3 – Requisitos funcionais do Módulo *Seeker*

Identificador	Descrição
RFS1	O sistema deve obter os imóveis das imobiliárias integradas
RFS2	O sistema deve inserir, atualizar e excluir os imóveis
RFS3	O sistema deve persistir cada etapa executada
RFS4	O sistema ao ser reiniciado deve recomeçar de onde parou
RFS5	O sistema deve registrar *log* de cada etapa efetuada

Fonte: Próprio Autor. 2013.

Tabela 4.4 – Requisitos não funcionais do Módulo *Seeker*

Identificador	Descrição
RNFS1	O sistema deve ser executado diariamente
RNFS2	O sistema deve ser hospedado em uma hospedagem Windows com *.NET Framework 4* instalado
RNFS3	Utilização de Banco de Dados Relacional: MySQL
RNFS4	Utilização de Banco de Dados NoSQL: DynamoDB

Fonte: Próprio Autor. 2013.

O quadro 4.3 apresenta o caso de uso de obter imóveis do módulo *Seeker*, onde descreve como é feita a integração dos sites das imobiliárias parceiras com o sistema.

Quadro 4.3 – Caso de uso Obter Imóveis

UCS001 – Obter Imóveis	
Descrição	Para obter os imóveis, o módulo *Seeker* faz o *download* do código HTML do site da imobiliária e analisa as informações. O sistema é executado diariamente, inserindo atualizando e excluindo os imóveis do Avisa Imóvel.
Requisitos	RFS1 RNFS1 RNFS2 RNFS4
Ator	Site da imobiliária, Sistema *Seeker*
Pré-condição	Endereço do site da imobiliária de onde serão obtidos os imóveis
Fluxo Principal	- Sistema verifica se é preciso iniciar a obtenção dos imóveis ou recomeçar de onde parou (UCS002)

	- Sistema inicia a obtenção dos imóveis - O sistema faz o *download* do código HTML das páginas onde estão os imóveis cadastrados no site da imobiliária - Cada código é analisado e obtido o endereço para acessar o detalhe de cada imóvel - Ao obter o endereço o caso de uso UCS002 é iniciado - Ao término das páginas de listagem, o módulo inicia o *download* do código HTML de cada página de imóvel persistida pelo UCS002. - Cada código é analisado e são obtidas as seguintes informações do imóvel: Tipo de negociação; Tipo do imóvel; localização; valor; descrição; código do imóvel; endereço na imobiliária; fotos e informações adicionais - Sistema verifica se é preciso incluir ou atualizar o imóvel (casos de uso UCS003 e UCS004, respectivamente). - Sistema chama o UCS002 armazenando que o imóvel foi salvo - Ao término da lista de endereços do detalhe de cada imóvel, o sistema exclui os imóveis que não estavam mais no site (UCS005)
Fluxo Alternativo 1	- Sistema verifica se é preciso iniciar a obtenção dos imóveis ou recomeçar de onde parou (UCS002) - Sistema deve recomeçar - Sistema inicia o *download* do código HTML de cada página de imóvel persistida pelo UCS002, que ainda não foi salvo. - Cada código é analisado e são obtidas as seguintes informações do imóvel: Tipo de negociação; Tipo do imóvel; localização; valor; descrição; código do imóvel; endereço na imobiliária; fotos e informações adicionais - Sistema verifica se será preciso incluir ou atualizar o imóvel (casos de uso UCS003 e UCS004, respectivamente). - Sistema chama o UCS002 armazenando que o imóvel foi salvo

	- Ao término da lista de endereços do detalhe de cada imóvel, o sistema exclui os imóveis que não estavam mais no site (UCS005)
Fluxo Alternativo 2	- Sistema não encontra o bairro do imóvel - Sistema verifica em tabela de correção de bairro se existe o equivalente para o bairro do imóvel - Caso o sistema encontre, o bairro é atualizado pelo equivalente - Caso o sistema não encontre é registrado o *log* no banco de dados NoSQL com a mensagem: "Não foi encontrado o bairro para o imóvel da 'Imobiliária X' – 'Endereço do imóvel'"
Fluxo Alternativo 3	- Sistema não encontra a cidade do imóvel - Sistema insere *log* no banco de dados NoSQL com a mensagem: "Não foi encontrado o município para o imóvel da 'Imobiliária X'"
Fluxo Alternativo 4	- Imóvel está sem valor - Sistema coloca o valor zero
Fluxo Alternativo 5	- Tipo do imóvel não existe - Sistema coloca o tipo do imóvel igual a outros
Pós-condição	Os imóveis da imobiliária foram todos atualizados

Fonte: Próprio Autor. 2013.

No quadro 4.4 são apresentados os fluxos para persistir cada execução feita pelo módulo *Seeker* ao obter os imóveis dos sites das imobiliárias parceiras.

Quadro 4.4 – Caso de uso Persistir execução

UCS002 – Persistir execução	
Descrição	A cada endereço de detalhe de imóvel obtido é salva a informação juntamente a qual imobiliária pertence. Ao ser iniciado o *download* das informações de imóveis é inserida a informação de que aquele imóvel foi obtido
Requisitos	RFS3 RNFS2 RNFS3
Ator	Sistema *Seeker*

Fluxo Principal	- O UCS001 envia o endereço de detalhe do imóvel obtido - Sistema armazena o endereço e a qual imobiliária pertence
Fluxo Alternativo 1	- O UCS001 envia a informação de que o imóvel foi obtido - Sistema armazena que o imóvel já foi obtido
Fluxo Alternativo 2	- O UCS001 quer saber se o sistema parou e é necessário recomeçar de onde parou - Se nenhum imóvel foi obtido ou se já passou mais de 5 horas que o último imóvel foi obtido, precisa obter os endereços novamente, endereços são excluídos - Caso contrário, retorna a lista de endereços para recomeçar de onde parou
Pós-condição	Execução persistida

Fonte: Próprio Autor. 2013.

Os outros casos de uso de inserir, atualizar e deletar imóveis da integração do sistema com os sites das imobiliárias são apresentados no apêndice C. Sendo assim, a especificação dos requisitos do módulo *Seeker* foi concluída e documentada para que seja utilizada na elaboração do sistema.

4.1.3 Módulo *Job*

As tabelas 4.5 e 4.6 apresentam os requisitos funcionais e não-funcionais, respectivamente, do módulo *Job*.

Tabela 4.5 – Requisitos funcionais do Módulo *Job*

Identificador	Descrição
RFJ1	O sistema deve comparar o perfil do usuário com os imóveis
RFJ2	O sistema deve inserir e atualizar sugestões de imóveis para os usuários consumidores
RFJ3	O sistema deve classificar as sugestões geradas em uma escala de 1 a 5
RFJ4	O sistema deve gerar os e-mails a serem enviados com as sugestões para os usuários consumidores
RFJ5	O sistema deve enviar os e-mails, de acordo com um número parametrizado de envios por hora

Fonte: Próprio Autor. 2013.

Tabela 4.6 – Requisitos não funcionais do Módulo *Job*

Identificador	Descrição
RNFJ1	O sistema só pode criar, atualizar e excluir sugestões quando o módulo *Seeker* não está sendo executado
RNFJ2	O sistema deve ser hospedado em uma hospedagem Windows com *.NET Framework 4* instalado
RNFJ3	Utilização de Banco de Dados Relacional: MySQL
RNFJ4	Utilização de Banco de Dados NoSQL: DynamoDB
RNFJ5	O módulo deve ser executado a cada 10 minutos

Fonte: Próprio Autor. 2013.

O quadro 4.5 descreve o caso de uso da comparação do perfil dos usuários consumidores com os imóveis cadastrados no sistema. Ao comparar também são classificadas as sugestões geradas em uma escala de 1 a 5 e armazenadas aquelas sugestões igual ou maior que 3.

Quadro 4.5 – Caso de uso Comparar perfil do usuário com imóveis

UCJ001 – Comparar perfil do usuário com imóveis	
Descrição	O sistema recebe o perfil do usuário e compara com todos os imóveis ativos do sistema, classificando a sugestão a ser gerada. A classificação do imóvel é baseada em uma escala de 1 a 5, sendo que somente sugestões maiores ou iguais a 3 são inseridas. As escalas seguem as seguintes regras: - Escala 5: O usuário deve ter selecionado em seu perfil o mesmo tipo de negociação, o mesmo tipo de imóvel, o mesmo bairro, cidade e estado, a faixa de valor dentro do valor do imóvel e as mesmas informações adicionais - Escala 4: O usuário deve ter selecionado o mesmo tipo de negociação, o mesmo tipo de imóvel, o mesmo bairro, cidade e estado, a faixa de valor dentro do valor do imóvel com tolerância de 10% para mais e para menos - Escala 3: O usuário deve ter selecionado o mesmo tipo de negociação, o mesmo tipo de imóvel, o mesmo bairro, cidade e estado, a faixa de valor dentro do valor do imóvel com tolerância de 30% para mais e para menos

Requisitos	RFJ1 RFJ3 RFJ4 RNFJ1 RNFJ2 RNFJ3 RNFJ4
Ator	Sistema *Job*
Fluxo Principal	- Sistema *Job* é iniciado - Sistema verifica se o módulo *Seeker* não está sendo executado - Sistema carrega todos os usuários ativos - Para cada usuário, o sistema executa consulta ao banco de dados relacional para obter: * Os imóveis que são classificados como 5 na escala. * Os imóveis que são classificados como 4 na escala. * Os imóveis que são classificados como 3 na escala. - Para cada imóvel obtido é criada a sugestão classificada - Verifica se o usuário já tem sugestão para aquele imóvel e se a classificação continua igual, caso seja diferente atualiza a sugestão (USJ003). Caso não exista, insere uma nova sugestão (USJ002). - Sistema verifica se é necessário enviar e-mail com as novas sugestões geradas, de acordo com a periodicidade escolhida pelo usuário. Se sim, gera um novo e-mail (USCJ005).
Fluxo Alternativo 1	- Sistema recebe o perfil de um usuário para comparar - Sistema executa a consulta ao banco de dados relacional para obter: * Os imóveis que são classificados como 5 na escala. * Os imóveis que são classificados como 4 na escala. * Os imóveis que são classificados como 3 na escala. - Para cada imóvel obtido é criada a sugestão classificada - Verifica se o usuário já tem sugestão para aquele imóvel e se a classificação continua igual, caso seja diferente atualiza a sugestão (USJ003). Caso não exista, insere uma nova sugestão (USJ002). - Sistema verifica se é necessário enviar e-mail com as novas sugestões geradas, de acordo com a periodicidade escolhida pelo

	usuário. Se sim, gera um novo e-mail (USCJ005).
Fluxo Alternativo 2	- Sistema não encontra nenhuma nova sugestão para o usuário - Sistema verifica se está no dia de enviar e-mail com as últimas sugestões geradas - Sistema verifica se as sugestões geradas são mais novas que a data do último e-mail enviado e seleciona somente as que são mais novas. - Sistema gera um e-mail com as sugestões novas (USCJ005).
Fluxo Alternativo 3	- Ocorre algum erro no momento de comparar o usuário - Sistema registra *log* no banco de dados NoSQL com a mensagem: “Erro na comparação do usuário X. Mensagem de erro”.
Pós-condição	As sugestões e o e-mail são gerados

Fonte: Próprio Autor. 2013.

O quadro 4.6 apresenta o caso de uso gerar e-mail, em que o sistema recebe a lista de sugestões e utilizando um *template* padrão inclui as sugestões geradas, formando um e-mail para avisar ao usuário consumidor sobre novos imóveis.

Quadro 4.6 – Caso de uso Gerar e-mail

UCJ005 – Gerar e-mail	
Descrição	O sistema recebe a lista de sugestões que devem ser enviadas e é gerado o HTML do e-mail a ser enviado posteriormente
Requisitos	RFJ4 RNFJ2 RNFJ3
Ator	Sistema *Job*
Pré-condição	Sugestões a serem enviadas
Fluxo Principal	- Sistema obtém o *template* padrão de e-mail, com identificação do Avisa Imóvel e *link* para o módulo Web alertando que todas as sugestões geradas podem ser visualizadas nele - Sistema carrega o imóvel de cada sugestão, considerando as 10 mais recentes

	- Sistema gera dinamicamente o HTML para cada sugestão, com a foto principal do imóvel, tipo de negócio, localização, classificação, valor e *link* para o detalhe do imóvel no módulo Web. - Sistema registra no banco de dados relacional, que um novo e-mail deve ser enviado, o HTML do e-mail e o endereço de e-mail.
Fluxo Alternativo 1	- Ocorre algum erro no momento de gerar o e-mail - Sistema registra *log* no banco de dados NoSQL com a mensagem: “Erro ao salvar e-mail do usuário X. Mensagem de erro”.
Pós-condição	E-mail gerado e pronto para o envio

Fonte: Próprio Autor. 2013.

Os demais casos de uso de inserir e atualizar sugestões e enviar e-mail são apresentados no apêndice D. Desta forma, a especificação dos requisitos do módulo *Job* foi descrito para servir de documentação para o desenvolvimento.

4.2 Elaboração

Após a especificação dos requisitos, o desenvolvimento foi dividido em três etapas sendo elas: a modelagem dos dados no banco de dados relacional, a modelagem das classes do banco de dados NoSQL e a programação da ferramenta.

4.2.1 Modelagem dos dados relacionais

Para modelar os dados no SGBD MySQL foi utilizada a ferramenta MySQLWorkbench, que é uma ferramenta de código aberto e de uso gratuito, além de ser integrada com o MySQL.

Nesta ferramenta foram modeladas no total 29 tabelas. Por uma questão de espaço, na figura 4.1 é apresentado o Esquema Relacional da tabela imóvel e alguns de seus relacionamentos. A tabela imóvel é relacionada a tabela de fotos dos imóveis, o tipo de imóvel, o tipo de negócio, o usuário que cadastrou, a localização e com a tabela de informações adicionais do imóvel.

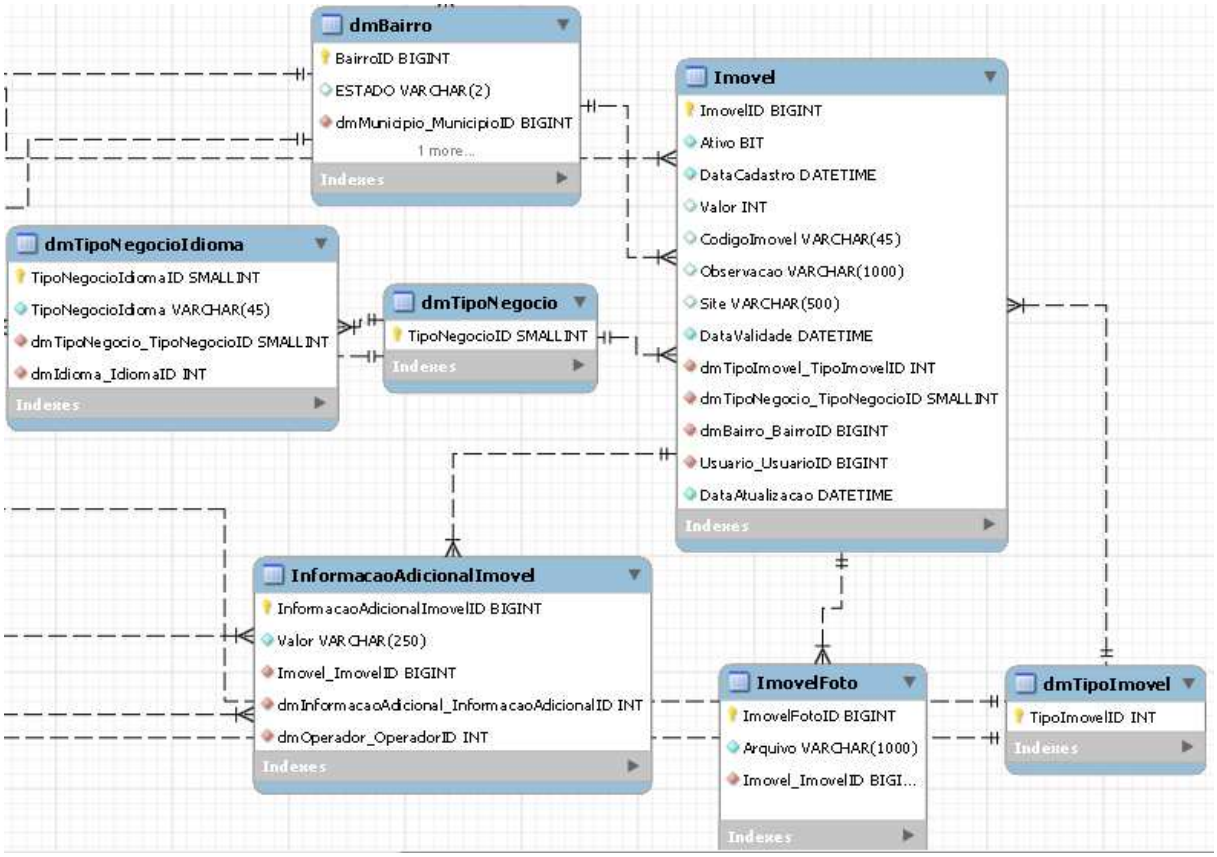


Figura 4.1 – Tabela imóvel e relacionamentos

Fonte: Próprio Autor (2013)

4.2.2 Modelagem dos dados NoSQL

Para modelar os dados no banco de dados NoSQL (DynamoDB) foi utilizado o Diagrama de Classes, por não existir um padrão de representação das tabelas. A figura 4.2 ilustra as duas tabelas e seus atributos, que foram criadas no banco de dados NoSQL.

Figura 4.2 – Diagrama de Classes - DynamoDB

Fonte: Próprio Autor (2013)

A tabela de sugestão armazena as sugestões geradas pelo módulo *Job* e o módulo Web faz a leitura dessas informações para exibir ao usuário consumidor a lista das sugestões geradas. A tabela *Log* salva todos os possíveis erros ocorridos, alertas ou estados de alguns processos.

4.2.3 Programação

Conforme definido na seção 2.1.1, a linguagem de programação utilizada para o desenvolvimento deste sistema foi o C#. O ambiente de desenvolvimento utilizado foi o Visual Studio 2010 e toda ferramenta foi desenvolvida utilizando o *.NET Framework 4.0*. Para acesso ao banco de dados MySQL foi utilizada a ferramenta LINQ, que abstrai o acesso aos dados.

A solução foi dividida em 5 projetos, sendo eles *AvisaImovel* (Módulo Web), *Gerenciador* (Módulo Web), *RN* (Regras de negócio, acesso ao MySQL, acesso ao DynamoDB e Módulo *Job*), *Seeker* (Módulo *Seeker*) e *SeekerRN* (Regras de negócio, acesso ao MySQL e ao DynamoDB do módulo *Seeker*). Esses projetos atenderam a todos os requisitos especificados anteriormente.

4.3 Resultado da Ferramenta

O sistema foi desenvolvido e posteriormente testado, conforme é especificado no capítulo 5. Em seguida foi publicado em produção e seu resultado é apresentado nas próximas seções.

4.3.1 Módulo Web

A ferramenta foi desenvolvida e o módulo Web foi publicado no endereço http://www.avisaimovel.com.br, disponível para os usuários fazerem a interação com o resto do sistema. A seguir são exibidas as principais telas deste módulo. A figura 4.3 é a tela inicial do site e direciona para as telas de cadastro de usuários consumidores, cadastro de usuários vendedores, autenticação de usuários cadastrados, além de páginas de apresentação e de contato.

Figura 4.3 – Tela inicial do módulo Web

Fonte: Próprio Autor (2013)

As próximas figuras estão sem o topo e o rodapé da tela, por uma questão de espaço, porém as telas seguem o padrão da figura 4.3. Na figura 4.4 é exibida a tela de cadastro de usuários consumidores e o cadastro de preferências. Esta tela é a mesma para o cadastro e edição. Ao inserir as preferências, elas são listadas e também é possível excluí-las clicando no "x" vermelho ou editá-las clicando em cima da linha.

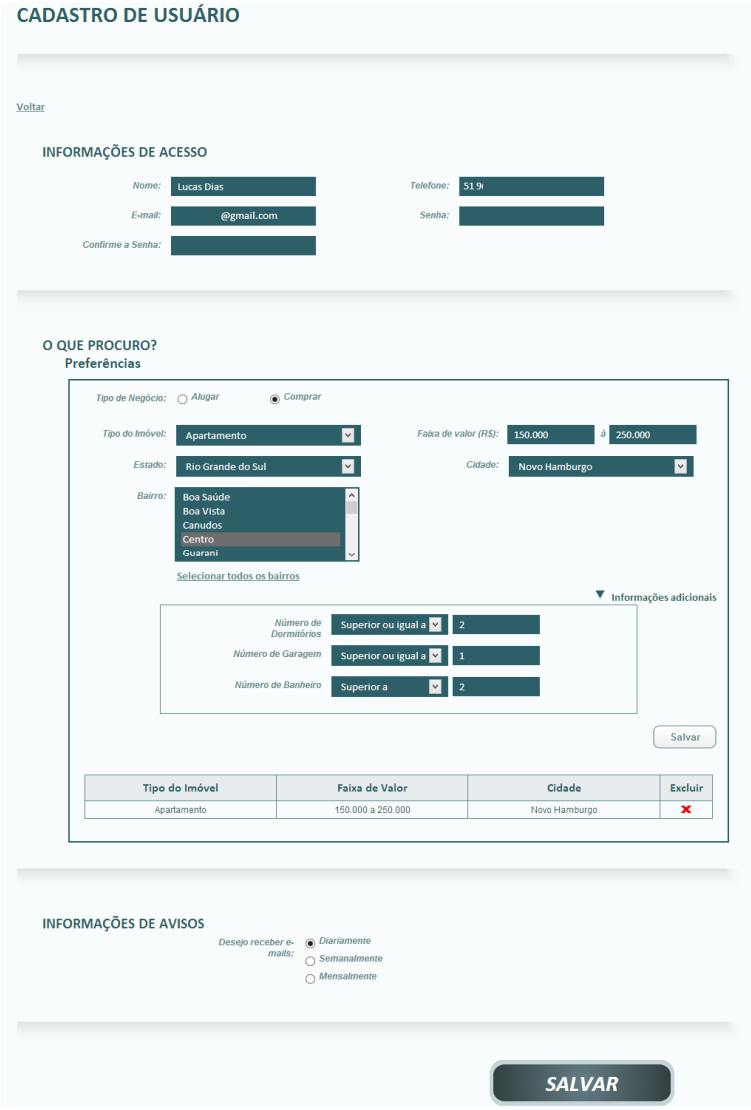


Figura 4.4 – Tela de cadastro e edição de usuários consumidores e suas preferências – Módulo Web

Fonte: Próprio Autor (2013)

A figura 4.5 apresenta parte da lista de sugestões geradas para um usuário consumidor, ordenado pelas mais recomendadas. A classificação das sugestões geradas é ilustrada pelas estrelas preenchidas. As principais informações e uma foto do imóvel são descritas de modo a auxiliar na visualização resumida dos imóveis sugeridos.

Ao clicar em *Ver Detalhe* na lista de sugestões, o imóvel é detalhado, conforme exemplo da figura 4.6. Nesta tela todas as informações obtidas do imóvel são exibidas para o usuário, incluindo todas as fotos. Ao clicar em cima de uma foto, abre-se uma janela em que é possível visualizar a foto selecionada em um tamanho maior.

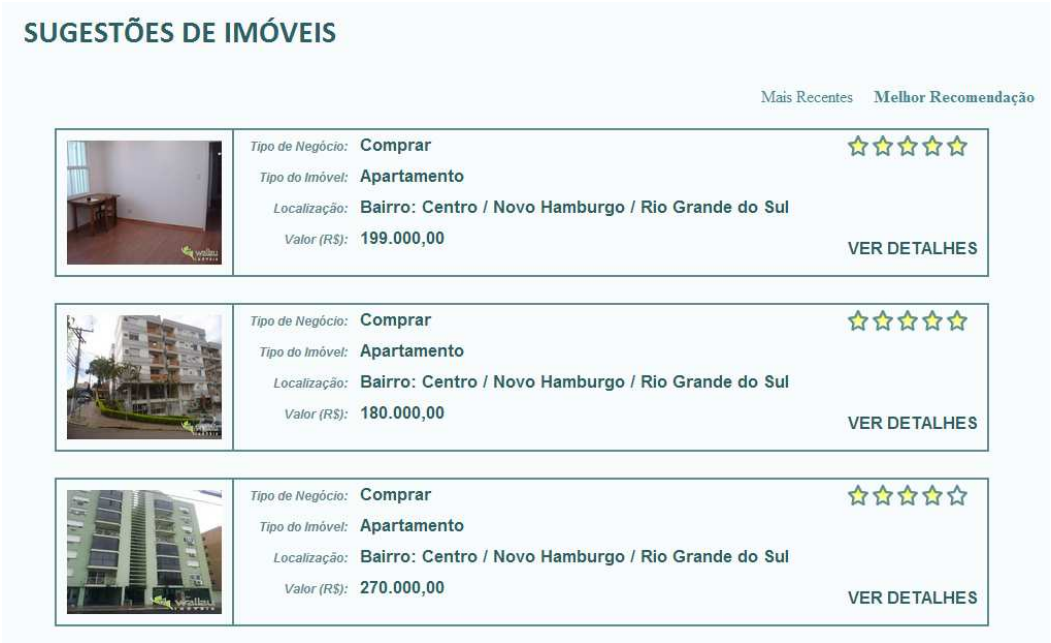


Figura 4.5 – Tela da lista de sugestões – Módulo Web

Fonte: Próprio Autor (2013)

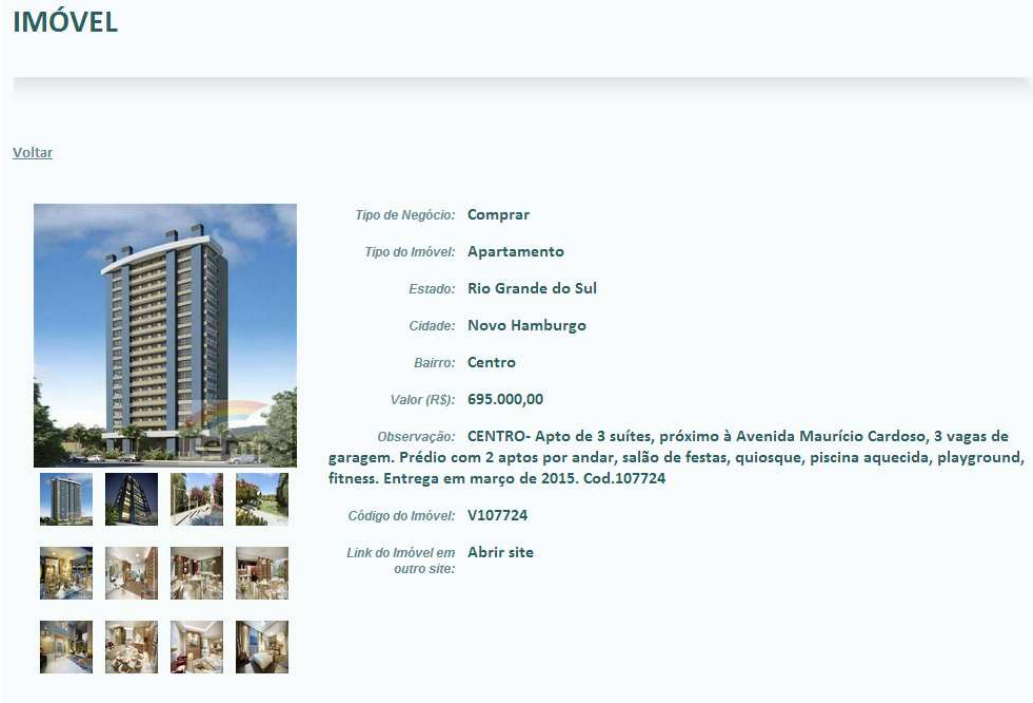


Figura 4.6 – Tela de detalhes de imóvel – Módulo Web

Fonte: Próprio Autor (2013)

Para os usuários vendedores, após fazerem o cadastro ele podem cadastrar imóveis na tela da figura 4.7. Nesta tela o usuário pode incluir as informações do imóvel, que serão posteriormente exibidas para o usuário, incluindo fotos e podendo descrever o imóvel em um campo livre.

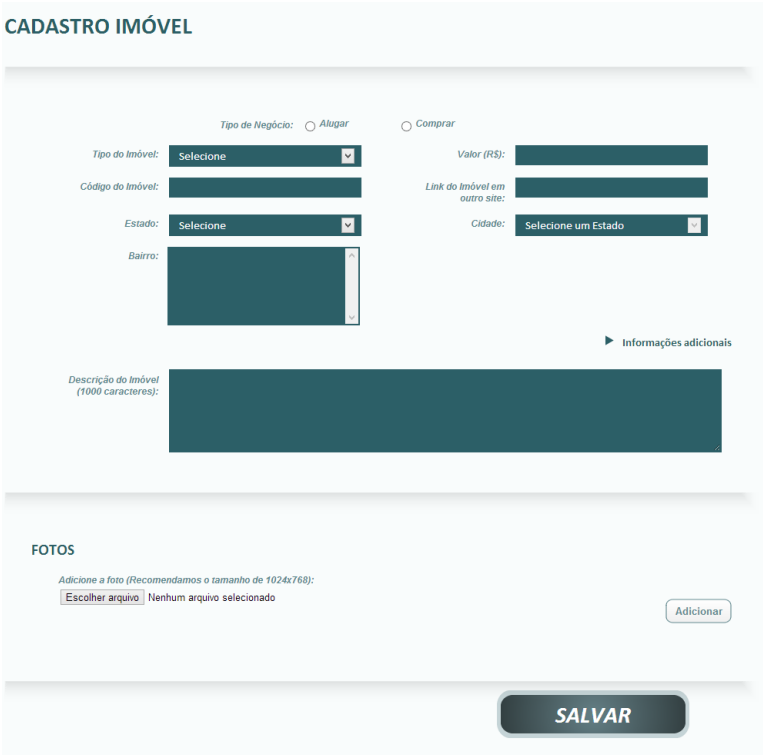


Figura 4.7 – Tela de cadastro de imóvel – Módulo Web

Fonte: Próprio Autor (2013)

A figura 4.8 apresenta a visualização da lista dos imóveis cadastrados pelo usuário vendedor. As informações resumidas são exibidas para o usuário vendedor e o sistema permite a exclusão do imóvel ao clicar no “x” vermelho.

Figura 4.8 – Tela da lista de imóveis cadastrados pelo usuário vendedor – Módulo Web

Fonte: Próprio Autor (2013)

4.3.2 Módulo *Seeker*

O módulo *Seeker* não tem visualização gráfica, porém para ilustrar o resultado da integração dos sites das imobiliárias com o sistema, a figura 4.9 exibe parte do resultado de uma consulta na tabela de persistência das execuções de cada etapa do módulo. Sendo assim é possível visualizar que os 4 primeiros itens já foram integrados e os imóveis foram inseridos ou atualizados no sistema. O restante ainda está aguardando a execução. Caso ocorra uma queda no servidor, quando este retornar, o módulo começa a integração a partir do quinto item, que ainda não tem data de utilização preenchida.

JobAuxID	URL	DataUtilizacao	Usuario_UsuarioID
91	http://www.tempoimoveis.com.br/343/Vendas/imovel-N...	2013-10-18 00:18:22	16
92	http://www.tempoimoveis.com.br/344/Vendas/imovel-N...	2013-10-18 00:18:29	16
93	http://www.tempoimoveis.com.br/351/Vendas/imovel-N...	2013-10-18 00:18:36	16
94	http://www.tempoimoveis.com.br/360/Vendas/imovel-N...	2013-10-18 00:18:43	16
95	http://www.tempoimoveis.com.br/367/Vendas/imovel-N...	*NULL*	16
96	http://www.tempoimoveis.com.br/372/Vendas/imovel-N...	*NULL*	16
97	http://www.tempoimoveis.com.br/373/Vendas/imovel-N...	*NULL*	16
98	http://www.tempoimoveis.com.br/376/Vendas/imovel-N...	*NULL*	16
99	http://www.tempoimoveis.com.br/377/Locacao/imovel-...	*NULL*	16

Figura 4.9 – Tabela de persistência - Módulo *Seeker*

Fonte: Próprio Autor (2013)

4.3.3 Módulo *Job*

Para ilustrar o resultado do módulo *Job*, a figura 4.10 apresenta uma lista das sugestões geradas por este módulo. Esses dados foram consultados no DynamoDB. Nesta figura cada linha é uma sugestão, que associa um usuário a um imóvel, tem a data em que foi criada e se a sugestão ainda está ativa ou não e qual a sua classificação.

A figura 4.11 exibe um exemplo do e-mail, que é enviado para os usuários consumidores com as últimas sugestões geradas. Este e-mail pode ter até 10 imóveis, de acordo com a quantidade de novas sugestões, caso o usuário consumidor deseje visualizar todas as sugestões, um *link* para o módulo Web é apresentado neste e-mail.

Usuário ID	Imóvel ID	Data Sugestão	Inativo	Recomendado
58	909	17/10/2013 03:11:50	0	4
58	907	17/10/2013 03:11:50	0	4
51	907	17/10/2013 03:11:43	0	4
90	907	17/10/2013 03:12:33	0	4
58	906	16/10/2013 03:38:23	0	4
73	906	16/10/2013 03:38:37	0	4
90	906	16/10/2013 03:38:52	0	4
58	905	16/10/2013 03:38:22	0	4
90	905	16/10/2013 03:38:52	0	3
58	902	16/10/2013 03:38:22	0	4
45	902	16/10/2013 03:38:08	0	4
51	902	16/10/2013 03:38:15	0	4
90	902	16/10/2013 03:38:51	0	4
58	901	16/10/2013 03:38:22	0	4
51	901	16/10/2013 03:38:16	0	3
90	901	16/10/2013 03:38:51	0	4
58	900	16/10/2013 03:38:21	0	4
58	899	16/10/2013 03:38:21	0	4
45	899	16/10/2013 03:38:09	0	3
51	899	16/10/2013 03:38:15	0	4

◄ 1 2 3 4 5 ►

Figura 4.10 – Sugestões geradas – Módulo *Job*

Fonte: Próprio Autor (2013)

Figura 4.11 – E-mail enviado com sugestões – Módulo *Job*

Fonte: Próprio Autor (2013)

Sendo assim, o objetivo de desenvolver uma ferramenta utilizando banco de dados relacional e um banco de dados NoSQL foi alcançado. Todos os requisitos funcionais e não funcionais definidos e os casos de uso foram programados e entregues na ferramenta exibida nas figuras anteriores.

5 TESTES E AVALIAÇÃO

Neste capítulo serão mostrados os testes executados em todo o sistema, detalhando os níveis utilizados, para que a ferramenta pudesse ser disponibilizada para todos os usuários. Também é avaliado o uso do DynamoDB apresentando as vantagens e desvantagens encontradas. Após, através de um questionário, será apresentada a opinião dos usuários da ferramenta. Por fim, será feita uma reflexão sobre os problemas e as dificuldades encontradas no decorrer deste trabalho.

5.1 Testes

Ao todo este trabalho utilizou quatro níveis de testes, todos eles foram baseados nos fluxos dos casos de uso. O teste unitário foi utilizado durante a fase de desenvolvimento, de acordo com Myers (2004), o seu objetivo é testar as partes do sistema separadas ou pequenas funções de um todo, para garantir o completo funcionamento da parte. Após isso, foi feito o teste de integração, onde todos os três módulos do sistema foram unidos e testados em um ambiente local, com o objetivo de validar se os módulos estavam integrados corretamente (MYERS, 2004).

Em seguida, o sistema foi publicado em um ambiente igual ao de produção, com os módulos distribuídos, este foi o teste de sistema, segundo Myers (2004) este teste tem como objetivo testar a ferramenta integrada e verificar se cumpria todos os requisitos definidos. Por fim, foi realizado o teste de aceitação, no qual foram convidadas 9 pessoas a se cadastrarem e utilizarem o sistema, com o objetivo de verificar falhas não encontradas anteriormente (MYERS, 2004).

Em todos os níveis de testes foi utilizada uma planilha para registrar as falhas encontradas para posterior correção. No quadro 5.1 são apresentadas as principais pendências encontradas de cada nível.

Quadro 5.1 – Principais pendências encontradas após realização dos testes

Nível de Teste	Status	Caso de Uso	Descrição da pendência	Observações
Unitário	Concluído	Cadastro de consumidor	Não está validando se e-mail já cadastrado	
Unitário	Concluído	Cadastro de preferências	Erro de referência de objeto ao editar filtro e	Estava dando erro ao salvar, pois existia uma

			ao salvar	referência errada ao objeto dmtipoimovel
Unitário	Concluído	Integração Imobiliárias	Deve ser tratado o Valor sob consulta	Quando tiver valor sob consulta coloca o valor zero pro imóvel
Integrado	Concluído	Sugestão	O módulo *Job* não está inativando Sugestões	
Integrado	Concluído	Sugestão	Estão sendo criadas sugestões de imóveis inativos ou com data de validade vencida	
Sistema	Concluído	Módulo *Seeker*	A hospedagem está derrubando o processo do *Seeker* e o sistema não está auto reiniciando	Foi desenvolvido para o sistema identificar quando houve queda do processo e reiniciar de onde parou de acordo com a persistência dos dados
Sistema	Concluído	Sistema	Não está sendo possível conectar no banco de dados MySQL	Foi preciso colocar algumas DLL do MySQL na Bin de cada projeto
Aceitação	Concluído	Cadastro de consumidor	Usuário seleciona estado, cidade, a lista de bairros é exibida. Ao trocar de estado a lista de bairros não é apagada.	
Aceitação	Concluído	Cadastro de consumidor	Ao exibir o erro quando vai salvar, deve desmarcar o *checkbox*	
Aceitação	Concluído	Cadastro de preferências	Quando um usuário autenticado cadastrar, atualizar ou excluir uma preferência deve imediatamente persistir no banco de dados	O módulo Web estava programado para somente persistir no momento de clicar no botão Salvar no final da página

Fonte: Próprio Autor. 2013.

Todas as pendências encontradas foram corrigidas e testadas novamente, em nível unitário e depois em nível de sistema. Portanto, a etapa de testes foi considerada encerrada e o sistema foi publicado em produção e divulgado ao público em geral.

5.2 Avaliação do uso de NoSQL

A ferramenta funciona com dois conceitos de armazenagem de dados diferentes. Este trabalho teve como objetivo utilizar um banco de dados relacional, quando a funcionalidade exigisse relacionamentos e um banco de dados NoSQL, quando exigisse escalabilidade e disponibilidade.

Após pesquisas, a definição do banco de dados NoSQL a ser utilizado foi o DynamoDB, que oferece um SDK para o desenvolvimento de aplicações, desta forma abstraindo funcionalidades e oferecendo ao desenvolvedor várias facilidades em seu uso. A utilização do SDK faz com que a experiência de desenvolvimento seja muito semelhante à utilização do LINQ, trazendo como vantagem uma baixa curva de aprendizado. Além disso, a grande vantagem é que ao utilizar o DynamoDB, a Amazon oferece um banco de dados escalável sem necessidade de administrá-lo.

Entretanto, o DynamoDB oferece duas funções de pesquisa nas tabelas, que são o *Query* e o *Scan*. A função *Query* oferece um melhor desempenho em relação à *Scan* (AWS, 2013). Mas a utilização da função *Query* está condicionada a utilizar obrigatoriamente o filtro de igualdade para o *Hash Key*, ou seja, ao fazer consultas com essa função é necessário conhecer o *Hash Key*. Caso seja preciso executar mais filtros na tabela, então é necessário utilizar a função *Scan*, que faz uma varredura em todos os dados para retornar o que é procurado.

No caso deste trabalho isso não é um problema para a tabela de Sugestão, pois ao ser modelada, o *Hash Key* utilizado foi o identificador do usuário e esta informação o módulo Web tem devido a autenticação do usuário. Já na tabela Log o *Hash Key* é um valor aleatório e único, sendo assim toda vez que é preciso carregar algum registro é necessário utilizar a função *Scan*. Por existir poucas consultas do módulo Web na tabela Log, isso não se torna um problema de desempenho.

Portanto, este pode ser um ponto negativo do DynamoDB e deve ser considerado na modelagem dos dados, sendo verificado o quanto que a tabela será requisitada e se ao executar a consulta à tabela o sistema tem conhecimento do valor do *Hash Key* a ser pesquisado.

5.3 Avaliação dos usuários

Com o sistema publicado em produção, os usuários consumidores se cadastraram na ferramenta e começaram a receber sugestões de imóveis. Com o objetivo de avaliar a aceitação dos usuários com a aplicação, foi desenvolvido um questionário com questões objetivas para analisar o perfil do usuário, a visão do usuário sobre o Avisa Imóvel, a usabilidade das principais telas, o funcionamento e desempenho do sistema. Ao final, dois campos de textos livres são optativos para que o usuário possa deixar um comentário e seu e-mail.

A população considerada neste questionário são os usuários que se cadastraram no sistema, sendo assim todos esses tem a mesma probabilidade de ser incluído na amostra, portanto será uma amostragem probabilística do tipo casual simples. Com esta, abordagem quantitativa, será possível avaliar a aceitação dos usuários em relação ao sistema.

Após o questionário ser desenvolvido e revisado foi realizado um pré-teste. Ele foi aplicado em um grupo de 5 usuários próximos ao autor deste trabalho, para verificar se existia alguma incoerência ou ajustes que deveriam ser feitos no questionário. Após verificar o pré-teste, foram incluídas duas novas questões, para que uma melhor análise do funcionamento do sistema fosse feita.

Uma das questões adicionadas é se o Avisa Imóvel havia gerado sugestões para ele. Desta forma é possível identificar se houve falha, caso o usuário não tenha recebido um e-mail com as sugestões. A outra questão, de múltipla escolha, deseja saber em que cidades o usuário procura imóveis, pois se for alguma cidade diferente de Novo Hamburgo, a eficiência do sistema é reduzida, uma vez que atualmente as duas imobiliárias participantes deste trabalho são desta cidade.

O questionário final foi enviado por e-mail para todos os usuários cadastrados no sistema, o qual é apresentado no Apêndice E. No total 27 usuários responderam a pesquisa de avaliação e o resultado é apresentado e analisado em seguida.

Em relação ao perfil dos usuários que responderam o questionário 85% tem interesse em comprar/alugar/vender algum imóvel e somente 7% já trabalhou/trabalha no mercado imobiliário. Na questão de múltipla escolha das cidades em que os usuários procuram imóveis 50% afirmaram que buscam imóveis em Novo Hamburgo. A figura 5.1 apresenta esta distribuição.

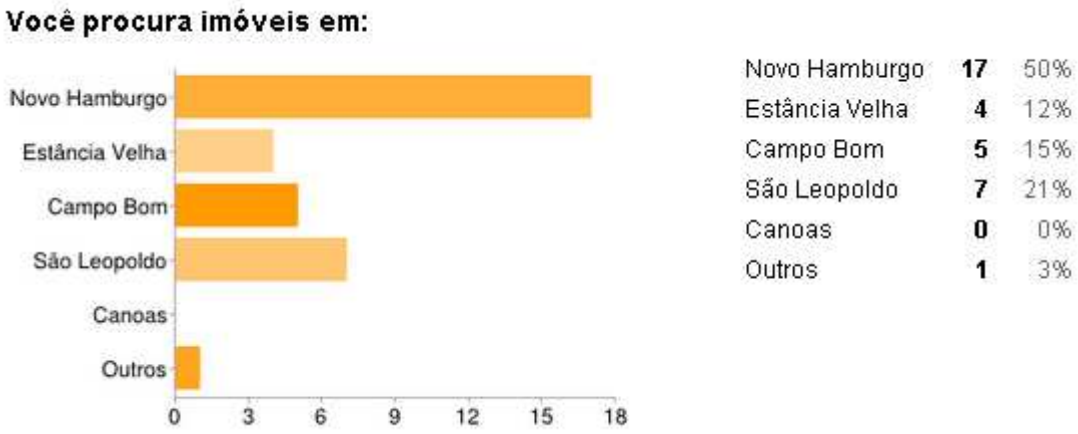


Figura 5.1 – Cidades em que os usuários procuram imóveis

Fonte: Próprio Autor (2013)

Em relação à opinião dos usuários sobre o Avisa Imóvel, o retorno foi positivo, pois 85% consideram que a ferramenta atinge o seu objetivo. Todos os usuários afirmam ser importante a sua atuação e 82% definem o Avisa Imóvel como muito relevante ou relevante para si. A figura 5.2 apresenta a visão dos usuários sobre a aplicação em percentuais e a quantidade de usuários que escolheram cada resposta.

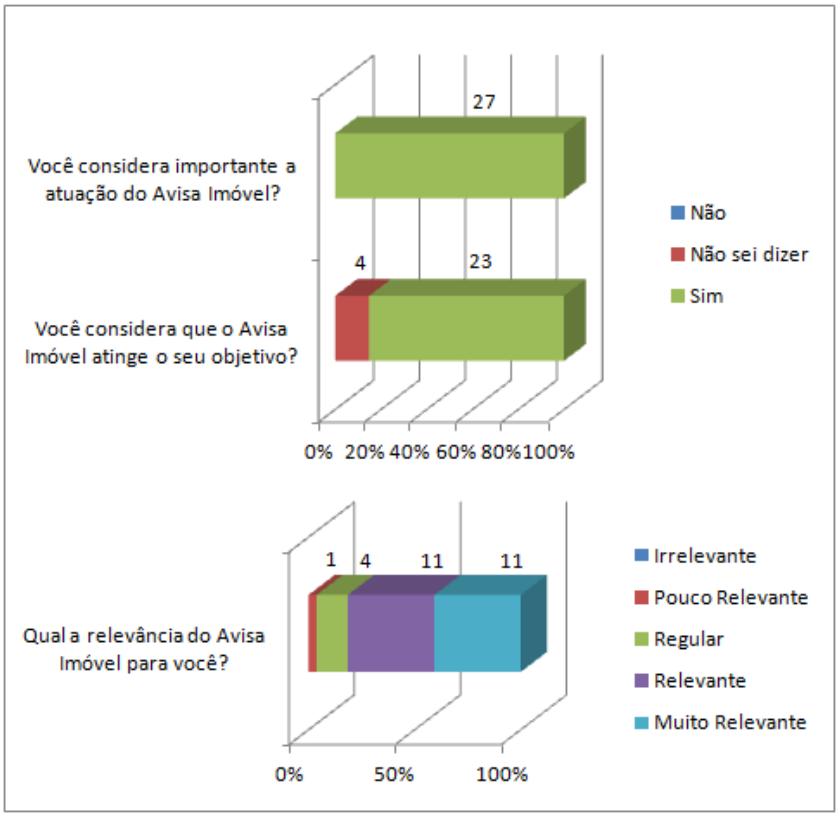


Figura 5.2 – Visão dos usuários sobre a aplicação

Fonte: Próprio Autor (2013)

Nas questões de usabilidade das principais telas do módulo Web, os que responderam o questionário consideraram em geral que o sistema é de fácil uso. Dos que responderam o questionário 37% consideram o sistema muito fácil e 41% afirmam ser fácil. Porém a tela de cadastro de preferências foi a tela em que os usuários encontraram maior dificuldade em utilizar. Na figura 5.3 são detalhadas as considerações dos usuários para cada tela.

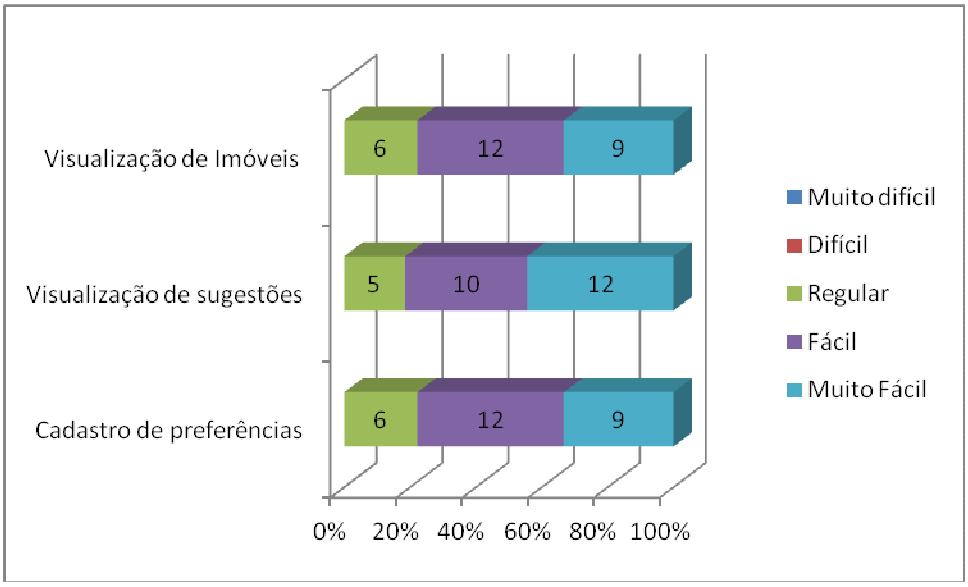


Figura 5.3 – Avaliação da usabilidade do módulo Web

Fonte: Próprio Autor (2013)

Quanto ao funcionamento do sistema, quatro usuários afirmaram que o Avisa Imóvel não gerou sugestões para eles. Isto ocorreu, pois conforme respondido posteriormente no questionário eles estavam procurando imóveis em São Leopoldo, Campo Bom e Porto Alegre, assim o Avisa Imóvel não encontrou os imóveis para estes usuários, pois a maioria dos imóveis cadastrados estão localizados em Novo Hamburgo. Desta forma, para avaliar este quesito, esses quatros usuários foram excluídos da análise. Sendo assim 100% dos usuários restantes afirmaram que receberam os e-mails enviados pelo Avisa Imóvel e 91% concordaram totalmente ou parcialmente de que as sugestões geradas estavam de acordo com o que eles procuravam. Este resultado é muito bom, pois demonstra que o Avisa Imóvel está funcional e conseguindo atingir o objetivo de conectar os consumidores e as imobiliárias automaticamente. Na figura 5.4 é possível visualizar o detalhamento das respostas dos usuários para cada questão.

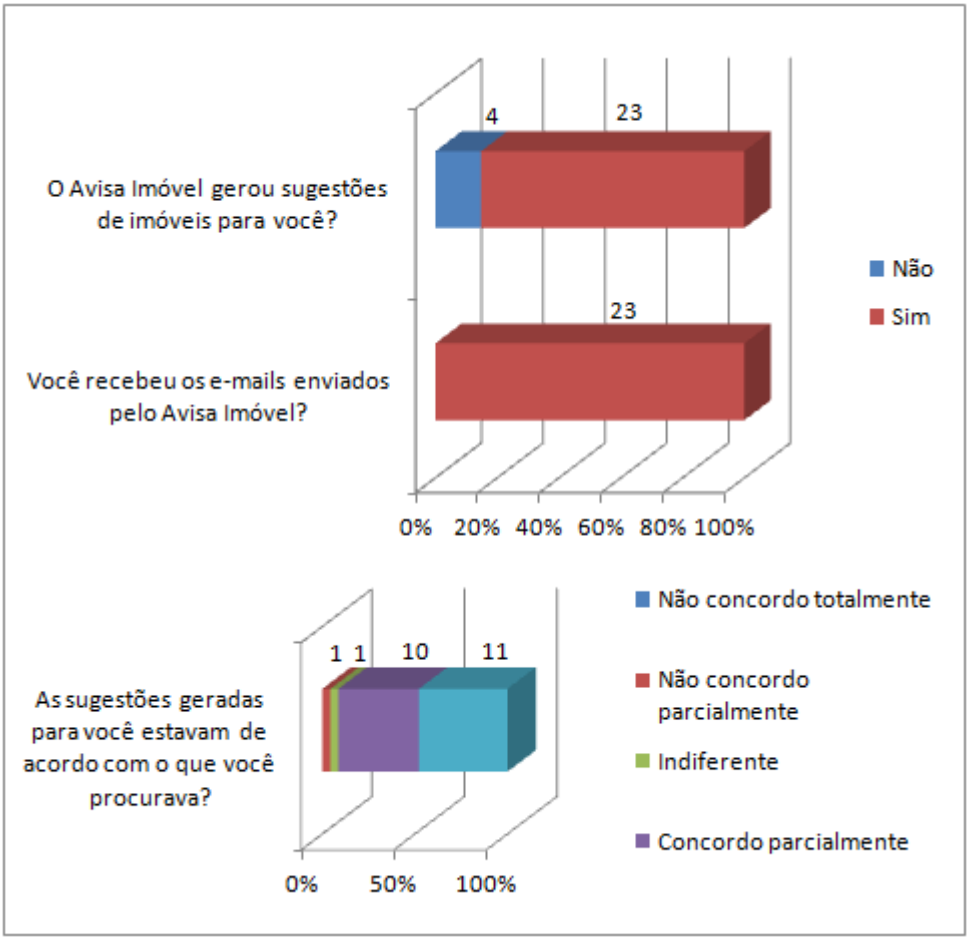


Figura 5.4 – Avaliação do funcionamento do sistema

Fonte: Próprio Autor (2013)

Quanto ao desempenho do Módulo Web, 63% consideram que ele está rápido, 26% muito rápido e 11% que está regular. Com esta avaliação sobre este trabalho, considera-se que o objetivo de ter um sistema disponível foi atingido e é notado pelo usuário.

Ao término desta análise quantitativa, conclui-se que a avaliação dos usuários sobre o sistema foi positiva e que os usuários reconhecem que o Avisa Imóvel atingiu o seu objetivo de ter uma ferramenta capaz de conectar as imobiliárias e os consumidores de forma automática. Ainda na pesquisa foram recebidos alguns comentários anônimos sobre o Avisa Imóvel, que podem ser visualizados na figura 5.5, cada retângulo cinza é um comentário de um usuário.

ÓTIMO SITE! PARABÉNS! Como estou em um momento de busca de um novo imovel e estava efetuando buscas em sites de imobiliarias o Avisaimovel veio de encontro e parece facilitar em muito o trabalho. No entanto, recebi somente um email com sugestões o que me faz crer que ainda há poucas opções cadastradas. Mas a ideia é sensacional. Muito útil, já que reúne informações de diversos sites de imobiliárias em um mesmo ambiente economizando tempo e paciência quando você esta procurando por imóveis. O Avisa Imóvel traz a praticidade que faltava para quem está pesquisando imóvel. Seu diferencial em reunir as diversas ofertas disponíveis num só site, proporciona um grande ganho de tempo para quem o utiliza. O grande diferencial do site é que você não recebe avisos indesejados. Você só vai receber aquelas informações que te interessam e que foram pré-selecionadas por você mesmo. A falta da opção "esqueci a senha" dificultou o acesso aos imoveis indicados pela ferramenta.

Figura 5.5 – Comentários dos usuários sobre a ferramenta

Fonte: Próprio Autor (2013)

5.4 Plano de Ação em função das avaliações dos usuários

Após a avaliação dos usuários e a análise das respostas, foi constatado que seriam necessários alguns ajustes no sistema. Dos usuários que responderam o questionário, 22% consideraram que a usabilidade no cadastro de preferências é regular. O cadastro de preferências exigia do usuário preencher as informações do imóvel procurado e clicar no botão adicionar, deste modo permitindo adicionar vários imóveis, porém isto causou confusão no primeiro contato com a ferramenta.

Então, para facilitar, foi feita uma alteração na página inicial, em que o usuário preenche o que procura e depois preenche somente os seus dados de contato e já está cadastrado. Esta alteração é apresentada na figura 5.6, onde o usuário preenche os dados do imóvel e na figura 5.7 em que o usuário preenche seus dados de contato. Porém, caso o usuário deseje, ele pode fazer o cadastro completo, como ocorria anteriormente, clicando no *link* desta ação.

Outra questão foi enviada nos comentários: de que faltava a funcionalidade de “esqueci minha senha”. Portanto foi desenvolvida esta funcionalidade, em que o usuário preenche o seu e-mail e o sistema gera uma nova senha e a envia por e-mail. Por fim, selecionaram-se três comentários positivos enviados no questionário para serem apresentados na página inicial do site com a opinião de usuários. Esta alteração pode ser visualizada na figura 5.6.

Figura 5.6 – Novo método de cadastro e a opinião dos usuários na página inicial

Fonte: Próprio Autor (2013)

Figura 5.7 – Novo método de cadastro etapa dos dados de contato

Fonte: Próprio Autor (2013)

Desta forma, este trabalho aplicou as melhorias nos principais pontos levantados pelos usuários na pesquisa de avaliação. Algumas questões pontuais, como os usuários que consideraram o desempenho regular e a falta de imóveis cadastrados no site serão tratadas em uma próxima etapa, migrando a aplicação para um servidor dedicado e contatando mais imobiliárias para se tornarem parceiras, respectivamente.

5.5 Dificuldades

Durante o desenvolvimento da ferramenta várias dificuldades foram encontradas. A seguir as principais são listadas e quais foram as ações para contornar os problemas.

Em relação às dificuldades encontradas com o banco de dados relacional encontrou-se problemas ao utilizar procedimentos armazenados (*Stored Procedures*) do MySQL com LINQ. A funcionalidade para inserir ou atualizar os procedimentos armazenados não estavam funcionando automaticamente pela ferramenta de Design do Visual Studio e, portanto, foi necessário fazer este processo manualmente editando o código-fonte dos arquivos XML e adicionando os procedimentos armazenados no arquivo, conforme explica Wickiy (2011).

No DynamoDB a principal dificuldade encontrada foi a falta de um banco de dados instalado localmente no computador do desenvolvedor, pois este banco de dados é um serviço Web. Sendo assim, ao contratá-lo é disponibilizado uma instância do serviço online em que é possível criar as tabelas. Essa única instância teve que ser utilizada no desenvolvimento, testes e posteriormente em produção. Este problema agora pode ser solucionado, pois no dia 18 de setembro de 2013 a Amazon anunciou a disponibilidade de uma ferramenta para desenvolvimento que trabalha como se fosse o DynamoDB localmente.

Por fim, durante a publicação e a distribuição do sistema nos servidores foi preciso utilizar dois métodos na programação para contornar problemas dos módulos estarem em hospedagens compartilhadas. Pois conforme citado na seção 2.2.5, esse tipo de hospedagem derruba a aplicação quando ela está ocupando muita memória ou utilizando o processamento do servidor acima do permitido.

Essas dificuldades não ocorreriam se o servidor fosse dedicado ao sistema, porém o custo para aquisição deste tipo de serviço é elevado. Uma solução foi salvar a sessão do usuário no banco de dados e não no servidor web (IIS). Desta forma caso o servidor derrube a aplicação, a sessão do usuário está persistida. A outra foi no módulo *Seeker*, armazenar cada

passo da integração dos sites das imobiliárias com o sistema, pois caso ocorresse alguma queda, o módulo reiniciasse do ponto onde parou.

Deste modo, o sistema foi testado e foi possível disponibilizar para o grande público um sistema que utiliza um SGBD relacional e um banco de dados NoSQL, posteriormente com a avaliação dos usuários concluiu-se que a ferramenta teve uma boa aceitação e 89% consideram ela entre rápido e muito rápido.

CONCLUSÃO

No Brasil, o setor da construção está crescendo e com isto a demanda de vendas nas imobiliárias também aumenta. Para alcançar o consumidor, as imobiliárias investem em diversos meios de publicidade. Atualmente, o consumidor está conectado na internet e, portanto este se torna um grande meio de publicidade. Por outro lado, diante de tantas oportunidades o cliente precisa pesquisar por novos imóveis repetitivas vezes.

Os sites existentes atualmente, sites das imobiliárias ou portais de classificados de imóveis, eles não se conectam com o consumidor de forma automática, isto é eles aguardam o consumidor acessar o site e fazer buscas. Portanto, em relação aos sites existentes no mercado o grande diferencial é que o Avisa Imóvel armazena o que o consumidor está procurando de imóveis e o avisa por e-mail sempre que surgir novas oportunidades, de acordo com o perfil do usuário.

Diante deste cenário, avaliaram-se algumas imobiliárias da cidade de Novo Hamburgo e duas imobiliárias disponibilizaram os imóveis cadastrados no seu respectivo site ao sistema a ser desenvolvido. Em paralelo a esta definição, pesquisou-se as linguagens de programação e o banco de dados relacional que se adequariam a este trabalho, concluindo-se que a linguagem a ser utilizada seria o C# com *Framework .NET* e o banco de dados MySQL.

Além disso, foi necessário definir um banco de dados NoSQL, para cumprir o objetivo de desenvolver uma ferramenta Web que utilize este tipo de banco, conforme pesquisado esta tecnologia oferece alta disponibilidade e escalabilidade. Com isso, analisou-se como melhor alternativa o DynamoDB, que oferece ótimo desempenho, armazenamento seguro, escalabilidade e facilidade na administração.

O sistema deste trabalho ficou dividido em três módulos: Web, *Seeker* e *Job*. O módulo Web faz a interação dos usuários com os dados do sistema. O módulo *Seeker* integra as imobiliárias selecionadas com este trabalho, obtendo os imóveis nos sites das imobiliárias. Por fim, o módulo *Job* analisa e cruza os dados com o objetivo de criar sugestões para os usuários e em seguida enviar um e-mail sugerindo novas oportunidades de negócios.

Após descrever as funcionalidades que o sistema deveria ter e suas integrações entre os módulos, foi especificado os requisitos funcionais e não-funcionais de cada módulo. Em seguida, especificaram-se os requisitos através de casos de uso dos diagramas desenvolvidos

anteriormente. Desta forma, foi possível documentar como seria o sistema a ser desenvolvido posteriormente.

Com a documentação concluída e as ferramentas de desenvolvimento definidas, iniciou-se a codificação dos módulos. Em paralelo a esta etapa, testes foram executados para identificação de problemas. Ao término da codificação, passou-se por mais três níveis de testes a fim de encontrar mais erros que não foram levantados anteriormente, os problemas encontrados foram documentados e posteriormente corrigidos.

Com o sistema desenvolvimento e testado, ele foi liberado e divulgado ao público em geral poder fazer o cadastro e com isso ter acesso a todas as funcionalidades existentes. Logo após, foi desenvolvido um questionário de avaliação, que foi enviado a estes usuários cadastrados. Desta forma, foi possível obter a opinião dos usuários sobre a ferramenta.

Com esta avaliação, conclui-se que 100% dos usuários que responderam o questionário consideram importante a atuação do Avisa Imóvel e 85% destes afirmam que o objetivo desta ferramenta foi atingido. Ainda, 89% consideraram o desempenho do site entre muito rápido ou rápido e de um modo geral 79% avaliaram a usabilidade entre muito fácil e fácil.

Diante deste retorno dos usuários verificou-se uma posição muito positiva da ferramenta desenvolvida neste trabalho, por outro lado também surgiram novas sugestões de funcionalidades ou melhorias que deveriam ser executadas. Sendo assim, desenvolveu-se a funcionalidade de “esqueci minha senha” e foram feitas alterações na página inicial do site para melhorar a usabilidade do cadastro e expor algumas opiniões dos usuários avaliados.

Entretanto, no decorrer deste trabalho surgiram várias novas ideias para aprimorar a ferramenta, sendo estes futuros trabalhos a serem desenvolvidos. A principal melhoria a ser executada no sistema é fazer a migração para um servidor dedicado, aumentando assim a qualidade do módulo *Job*, que não sofrerá com quedas do servidor compartilhado e também agregando um maior desempenho a todo o sistema. Em paralelo, deverão ser contatadas mais imobiliárias para integrações, possibilitando para os usuários uma maior base de dados de imóveis.

Em relação a novas funcionalidades, ficou de fora do escopo inicial a possibilidade de edição dos imóveis cadastrados diretamente no site, sendo possível somente a sua exclusão. Além disso, será desenvolvido no futuro relatórios para o acompanhamento da

quantidade de sugestões e visualizações em cada imóvel, podendo este ser acessado pelas imobiliárias parceiras.

Para qualificar ainda mais a comunicação com os usuários consumidores, serão avaliados novos métodos de aviso de novas sugestões como, por exemplo, o envio de mensagens de texto para o celular e a possibilidade de integrações com redes sociais. Neste estudo também será levado em consideração a possibilidade do desenvolvimento de um aplicativo para celulares, que poderia trabalhar com a funcionalidade de notificações do aparelho.

Por fim, verificou-se com a pesquisa de avaliação, que 20% dos usuários que responderam a pesquisa consideraram regular a usabilidade das telas de sugestões e imóveis. Com isto, devem ser reavaliadas essas telas, uma das ideias seria abrir os detalhes do imóvel na mesma página de sugestões. Desta forma, o usuário tem acesso mais informações do imóvel sem perder o contexto da lista de sugestões. Já na lista de sugestões, o desenvolvimento de filtros dos resultados e novos tipos de ordenação facilitaria a busca do usuário. Ainda poderá ser desenvolvida a possibilidade do usuário descartar uma sugestão, vai fazer com que ela não apareça mais na tela, gerando uma quantidade menor de informação visível.

Com este trabalho, foi possível evidenciar que ao desenvolver uma ferramenta utilizando dois conceitos de banco de dados, o sistema poderá ter as três características do teorema CAP (Consistência, Disponibilidade e Tolerância ao Particionamento) disponíveis para uso, de acordo com a necessidade de cada funcionalidade (escalabilidade e disponibilidade ou consistência).

Para garantir a consistência utilizou-se o SGBD de mercado MySQL. Já para permitir a ferramenta oferecer alta escalabilidade e disponibilidade utilizou-se o DynamoDB, concluindo-se que este banco de dados NoSQL supri esta demanda e tem como vantagem uma baixa administração do banco. Como desvantagem deste tipo de banco de dados NoSQL é que a busca com melhor desempenho só pode ser utilizada se for conhecido o identificador da tabela, o que nem sempre ocorre dependendo da funcionalidade.

Sendo assim, este trabalho atingiu os objetivos propostos desenvolvendo um sistema com alta escalabilidade, consistência e disponibilidade para conectar consumidores com o mercado imobiliário de forma automática, rápida e inteligente.

REFERÊNCIAS BIBLIOGRÁFICAS

ALMEIDA, R. C., BRITO, P. F. **Utilização da Classe de Banco de Dados NOSQL como Solução para Manipulação de Diversas Estruturas de Dados.** In: Encontro de Computação e Informática do Tocantins, 14., 2012, Palmas. **Anais...** Palmas: CEULP/ULBRA, 2012. p. 152-161.

AMAZON. **Amazon DynamoDB.** Disponível em: <http://aws.amazon.com/pt/dynamodb/>. Acesso em: 20 mar. 2013.

AMAZON. **Amazon DynamoDB:** Developer Guide. Disponível em: <http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html>. Acesso em: 12 mai. 2013.

ANDERSON, Dale. **Column Oriented Database Technologies.** Disponível em: <http://dbbest.com/blog/column-oriented-database-technologies/ >. Acesso em: 20 jun. 2013.

ATTON, Chris**. Approaching Alternative Media: Theory and Methodology**. Scotland: Napier University, 2001.

ÁVILA, Ana Luiza, SPÍNOLA, Rodrigo Oliveira. **Introdução a Engenharia de Requisitos.** In: Engenharia de Software Magazine, 46, 2007, Rio de Janeiro.

BRITO, Ricardo W. **Bancos de Dados NoSQL x SGBDs Relacionais:** Análise Comparativa. In: InfoBrasil, 3, 2010, Fortaleza. **Anais...** Fortaleza, 2010.

BUFITHIS, Gregory P.. **NoSQL, graph databases and Big Data.** Disponível em: <http://thecloudandediscovery.com/?p=847>. Acesso em: 20 jun. 2013.

CBIC DADOS. **Mercado Imobiliário**. Disponível em: <http://www.cbicdados.com.br/menu/mercado-imobiliario/>. Acesso em: 25 mar. 2013.

DIANA, Mauricio De; GEROSA, Marco Aurélio. **NOSQL na Web 2.0:** Um Estudo Comparativo de Bancos Não-Relacionais para Armazenamento de Dados na Web 2.0. In: Workshop de Teses e Dissertações em Banco de Dados, 9, 2010, Belo Horizonte. **Anais...** Belo Horizonte, 2010.

DORDOR, Xavier. **Mídia/mídia alternativa:** a escolha de uma estratégia global de comunicação para a empresa. São Paulo, SP: Nobel, 2007.

FAYAD, M. E., SCHMIDT, D. C. **Object-oriented Application frameworks. Communications of the ACM**, Vol. 40, 10 p., 1997.

FERREIRA, Edmar. **Introdução ao NoSQL parte I.** Disponível em: <http://escalabilidade.com/2010/03/08/introducao-ao-nosql-parte-i/ >. Acesso em: 05 mai. 2013.

GRAVES, Steve. **Multi-Core Software:** To Gain Speed, Eliminate Resource Contention. Disponível em: <http://www.rtcmagazine.com/articles/view/101612>. Acesso em: 10 mai. 2013.

GRIGORIK, Ilya. **Schema-Free MySQL vs NoSQL.** Disponível em: <http://www.igvita.com/2010/03/01/schema-free-mysql-vs-nosql/>. Acesso em: 05 mai. 2013.

HARRISON, Guy. **Consistency models in Non relational Databases.** Disponível em: <http://guyharrison.squarespace.com/blog/2010/6/13/consistency-models-in-non-relational-databases.html>. Acesso em: 05 mai. 2013.

IAB BRASIL. **Mercado digital cresce 32% em 2012, atingindo R$ 4,5 bi em publicidade.** Disponível em: <http://iabbrasil.net/portal/mercado-digital-cresce-32-em-2012-atingindo-r-45-bi-em-publicidade/>. Acesso em: 28 mai. 2013.

INTERMEIOS. **Relatórios de Investimentos.** Disponível em: <http://www.projetointermeios.com.br/relatorios-de-investimento>. Acesso em: 22 mai. 2013.

KENDZERSKI, Paulo Roberto. **Publicidade Offline X Publicidade On-line.** Disponível em: <http://www.biblioteca.sebrae.com.br/bds/bds.nsf/DowContador?OpenAgent&unid=C3632CC8511EE9D78325727B007E0C07>. Acesso em: 25 mai. 2013.

KHANH, Duy Nguyen. **NoSql-like approach for Gideros offline save game files.** Disponível em: <http://www.guava7.com/2012/nosql-like-approach-for-gideros-saved-files/>. Acesso em: 20 jun. 2013.

LOCAWEB. **Hospedagem de sites.** Disponível em: < http://www.locaweb.com.br/produtos/hospedagem-de-sites/planos.html>. Acesso em: 17 mai. 2013.

LÓSCIO, Bernadette Farias; OLIVEIRA, Hélio Rodrigues de; PONTES, Jonas César de Sousa. **NoSQL no desenvolvimento de aplicações Web colaborativas.** Disponível em: <http://www.addlabs.uff.br/sbsc_site/SBSC2011_NoSQL.pdf>. Acesso em: 20 mar. 2013.

MARUCCI, Fernando. **Oportunidades e desafios na Construção Civil.** Disponível em: < http://www.asapexec.com.br/2012/04/oportunidades-e-desafios-na-construcao-civil/>. Acesso em: 29 mar. 2013.

MICROSOFT. **Introdução ao .NET Framework.** Disponível em: <http://msdn.microsoft.com/library/vstudio/hh425099.aspx>. Acesso em: 15 mai. 2013.

MICROSOFT. **Visão geral sobre o .NET Framework.** Disponível em: <http://msdn.microsoft.com/library/vstudio/zw4w595w>. Acesso em: 15 mai. 2013.

MYERS, Glenford J. **The art of software testing.** New Jersey, EUA: Wiley, 2004.

NIELSEN. **Global Advertising Trends.** Disponível em: <http://www.nielsen.com/us/en/reports/2012/global-advertising-view---q2-2012.html>. Acesso em: 25 mai. 2013.

ORACLE. **JAVA.** Disponível em: <http://www.oracle.com/br/technologies/java/overview/index.html>. Acesso em: 15 mai. 2013.

ORACLE. **MySQL.** Disponível em: <http://www.oracle.com/br/products/mysql/index.html>. Acesso em: 20 mai. 2013.

PEIRIS, Prabath. **Document Oriented Database.** Disponível em: <http://prabathsql.blogspot.com.br/2013/02/document-oriented-database_14.html>. Acesso em: 20 jun. 2013.

PEREZ, Clotilde; BARBOSA, Ivan Santo. **HIPERPUBLICIDADE:** atividades e tendências. São Paulo, SP: Thomson Learning Edições, 2008.

PRODANOV, C. C.; FREITAS, E. C.; **Metodologia do Trabalho Científico – Métodos e Técnicas da Pesquisa e do Trabalho Acadêmico**. 2. ed. Editora Feevale, Novo Hamburgo, 2013.

SADALAGE, Pramod J; FOWLER, Martin. **NoSQL Distilled:** A Brief Guide to the Emerging World of Polyglot Persistence. 1. ed. Addison-Wesley Professional, Indiana, 2012.

SHALOM, Nati. **The Common Principles Behind the NOSQL Alternatives.** Disponível em: <http://natishalom.typepad.com/nati_shaloms_blog/2009/12/the-common-principles-behind-the-nosql-alternatives.html >. Acesso em: 10 mai. 2013.

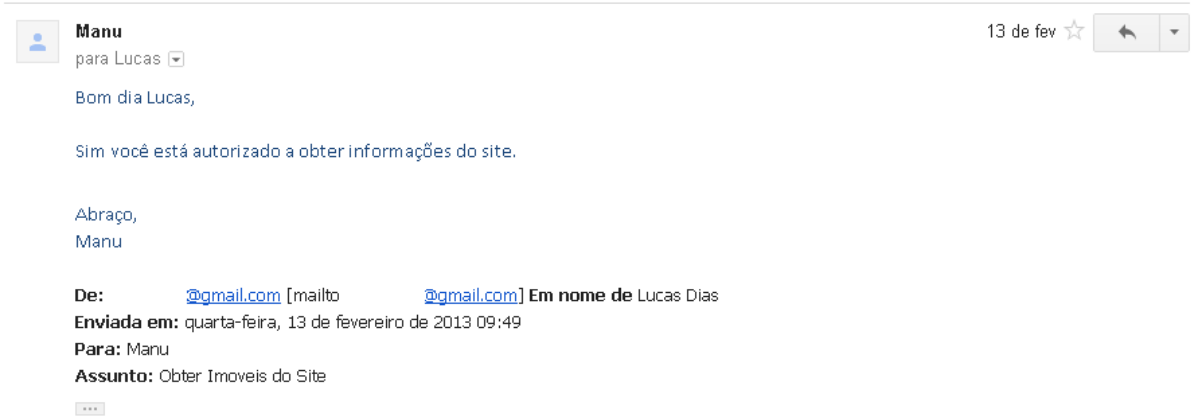
TAMANAHA, Paulo. **Planejamento de mídia: teoria e experiência.** São Paulo, SP: Pearson Prentice Hall, 2006.

VOGELS, Werner. **Eventually Consistent - Revisited.** Disponível em: <http://www.allthingsdistributed.com/2008/12/eventually_consistent.html>. Acesso em: 12 mai. 2013.

WICKIY. **Doesnt import arguments for store procedures on mysql with entity frameworks.** Disponível em: <http://social.msdn.microsoft.com/Forums/en-US/9c1e1e58-0531-4bbc-8206-eeeb6bc60fef/doesnt-import-arguments-for-store-procedures-on-mysql-with-entity-frameworks?forum=adodotnetentityframework >. Acesso em: 15 ago. 2013.

APÊNDICES

Apêndice A – E-mails de autorização para integração com as imobiliárias parceiras



Manu 13 de fev

para Lucas

Bom dia Lucas,

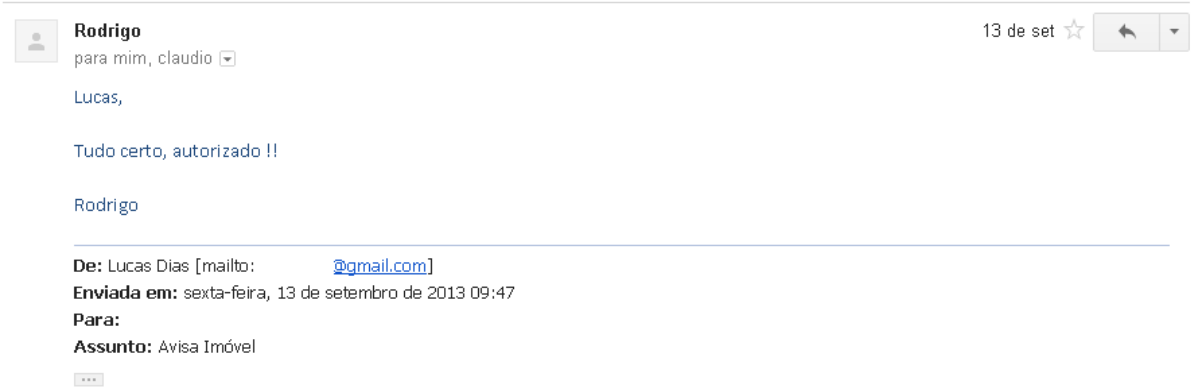
Sim você está autorizado a obter informações do site.

Abraço,
Manu

De: @gmail.com [mailto @gmail.com] **Em nome de** Lucas Dias
Enviada em: quarta-feira, 13 de fevereiro de 2013 09:49
Para: Manu
Assunto: Obter Imoveis do Site

Figura Apêndice 1 – E-mail de autorização para obter informações do site – Imobiliária 1

Fonte: Próprio Autor (2013)



Rodrigo 13 de set

para mim, claudio

Lucas,

Tudo certo, autorizado !!

Rodrigo

De: Lucas Dias [mailto: @gmail.com]
Enviada em: sexta-feira, 13 de setembro de 2013 09:47
Para:
Assunto: Avisa Imóvel

Figura Apêndice 2 – E-mail de autorização para obter informações do site – Imobiliária 7

Fonte: Próprio Autor (2013)

Apêndice B – Casos de uso do Módulo Web

Quadro Apêndice 1 – Caso de uso Visualizar sugestões

UCW003 – Visualizar sugestões	
Descrição	O usuário autenticado tem como página inicial sempre a lista de sugestões geradas para ele, de acordo com o seu perfil. A listagem é paginada exibindo 20 itens por página, podendo o usuário ordenar pelas sugestões mais recentes ou aquelas que são as mais recomendadas.
Requisitos	RFW4 RFW5 RNFW1 RNFW3 RNFW4 RNFW5
Ator	Usuário Consumidor
Pré-condição	Usuário autenticado
Fluxo Principal	- Sistema consulta as sugestões geradas no DynamoDB - Sistema lista as sugestões que os imóveis foram desativados e passa para o UCW009 - Sistema exibe a listagem de sugestões ativas paginada e ordenada pelas mais recentes - Sistema permite ordenar entre as mais recentes e as mais recomendadas
Fluxo Alternativo 1	- Sistema consulta as sugestões geradas no DynamoDB - Não existem sugestões para o usuário, sistema exibe a mensagem: “Ainda não encontramos nenhuma sugestão. Caso ainda não tenha adicionado filtros no seu perfil, clique em IMÓVEIS QUE PROCURA.”
Pós-condição	Lista de sugestões paginada

Fonte: Próprio Autor. 2013.

Quadro Apêndice 2 – Caso de uso Visualizar imóveis

UCW004 – Visualizar imóveis	
Descrição	Somente usuários autenticados podem visualizar imóveis. A página de detalhe do imóvel pode exibir as seguintes informações:

	Fotos, tipo de negócio, tipo do imóvel, localização, valor, observações, código do imóvel, *link* com acesso para um site externo.
Requisitos	RFW6 RNFW1 RNFW3 RNFW4 RNFW5
Ator	Usuário Consumidor
Pré-condição	Usuário autenticado
Fluxo Principal	- Sistema recebe por *Query String* o identificador do imóvel - Sistema carrega os dados do imóvel - Sistema registra no DynamoDB que o usuário visualizou o imóvel
Fluxo Alternativo 1	- Sistema recebe por *Query String* o identificador do imóvel - Sistema não encontra o imóvel - Sistema exibe a mensagem “Este imóvel não está mais disponível”
Fluxo Alternativo 2	- Usuário consumidor clica para abrir site externo - Sistema registra no DynamoDB que o usuário acessou o imóvel
Pós-condição	Imóvel exibido para o usuário

Fonte: Próprio Autor. 2013.

Quadro Apêndice 3 – Caso de uso Autenticar

UCW005 – Autenticar	
Descrição	O sistema deve garantir que os usuários tenham acesso a suas informações e sugestões geradas pelo sistema, através da autenticação do usuário. A autenticação é feita através da senha cadastrada pelo usuário.
Requisitos	RFW1 RNFW1 RNFW2 RNFW3 RNFW4
Ator	Usuário Consumidor
Fluxo Principal	- Usuário digita e-mail e senha

	- Sistema codifica a senha digitada em MD5 - Sistema verifica se e-mail e senha são iguais - Sistema autentica o usuário e direciona para o UCW003 - Sistema atualiza a cada interação, o tempo para mais 20 minutos, em que o usuário ficará autenticado.
Fluxo Alternativo 1	- Usuário digita e-mail e senha - Sistema codifica a senha digitada em MD5 - Sistema verifica se e-mail e senha são iguais - Sistema exibe a mensagem: “Usuário/senha incorretos”
Pós-condição	Usuário autenticado

Fonte: Próprio Autor. 2013.

Quadro Apêndice 4 – Caso de uso Cadastrar Vendedor

UCW006 – Cadastrar Vendedor	
Descrição	Na página de cadastro de usuário consumidor tem que ser informado o nome, telefone, e-mail, senha, confirmação de senha e aceitação dos termos de uso do site.
Requisitos	RFW2 RNFW1 RNFW3 RNFW4
Ator	Usuário Vendedor
Fluxo Principal	- Usuário acessa a página para cadastrar um novo usuário vendedor - Usuário preenche todos os campos corretamente e clica em finalizar - Sistema cadastra o novo usuário - Sistema autentica o novo usuário
Fluxo Alternativo 1	- Usuário autenticado clica para alterar o seu cadastro - Usuário preenche todos os campos corretamente e clica em finalizar

	- Sistema atualiza o cadastro do usuário
Fluxo Alternativo 2	- Usuário não aceita os termos de uso - Sistema não permite que o usuário clique no botão finalizar
Fluxo Alternativo 3	- Usuário não preenche qualquer um dos campos e clica em finalizar - Sistema exibe ao lado do campo a mensagem “Campo Obrigatório” na cor vermelha
Fluxo Alternativo 4	- Usuário preenche com valores diferentes o campo senha e confirmação de senha - Sistema exibe ao lado do campo a mensagem “Senhas não conferem” na cor vermelha
Fluxo Alternativo 5	- Usuário preenche um e-mail que já está cadastrado no sistema e clica em finalizar - Sistema exibe ao lado do campo a mensagem “E-mail já cadastrado” na cor vermelha
Fluxo Alternativo 6	- Usuário preenche o telefone com letras - Sistema não permite a inclusão de letras
Pós-condição	Os dados do usuário estão armazenadas no bancos de dados relacional

Fonte: Próprio Autor. 2013.

Quadro Apêndice 5 – Caso de uso Cadastrar imóveis

UCW007 – Cadastrar imóveis	
Descrição	Usuário cadastrar o imóvel obrigatoriamente preenchendo as informações: Tipo de negociação, tipo do imóvel, valor e localização. Opcionalmente o usuário pode inserir as informações: código do imóvel, endereço em outro site, observação, número de dormitórios, número de garagem, número de banheiro e fotos.

Requisitos	RFW7 RNFW1 RNFW3 RNFW4
Ator	Usuário vendedor
Pré-condição	Usuário autenticado
Fluxo Principal	- Usuário preenche todos os campos corretamente e clica em finalizar - O imóvel fica ativo durante os próximos 30 dias - Sistema cadastra o novo imóvel - Sistema exibe mensagem de sucesso
Fluxo Alternativo 1	- Usuário não preenche qualquer um dos campos e clica em finalizar - Sistema exibe ao lado do campo a mensagem “Campo Obrigatório” na cor vermelha
Fluxo Alternativo 2	- Ocorre algum erro ao inserir o imóvel - Sistema exibe a mensagem: “Ocorreu um erro para executar sua ação. Tente novamente mais tarde.”
Pós-condição	Imóvel cadastrado

Fonte: Próprio Autor. 2013.

Quadro Apêndice 6 – Caso de uso Deletar imóveis

UCW008 – Deletar imóveis	
Descrição	Lista de todos os imóveis ativos cadastrados pelo usuário com a possibilidade de excluir logicamente
Requisitos	RFW8 RNFW1 RNFW3 RNFW4
Ator	Usuário vendedor
Pré-condição	Usuário autenticado
Fluxo Principal	- Sistema exibe a lista de todos os imóveis ativos - Usuário seleciona qual quer excluir - Sistema pergunta se o usuário tem certeza

	- Sistema exclui logicamente o imóvel
Fluxo Alternativo 1	- Usuário não tem imóveis cadastrados - Sistema exibe a mensagem: “Você não tem imóveis cadastrados”
Pós-condição	Imóvel excluído logicamente

Fonte: Próprio Autor. 2013.

Quadro Apêndice 7 – Caso de uso Deletar sugestões

UCW009 – Deletar sugestões	
Descrição	O sistema envia as sugestões que devem ser deletadas no banco de dados NoSQL
Requisitos	RFW9 RNFW3 RNFW4 RNFW5
Ator	Usuário Consumidor
Pré-condição	Lista de sugestões a serem deletadas
Fluxo Principal	- Sistema deleta logicamente as sugestões
Fluxo Alternativo 1	- Sistema tenta deletar as sugestões e ocorre um erro - Sistema registra *log* no banco de dados NoSQL com a mensagem: “Erro ao deletar sugestão para o ‘Código identificador do Usuário’ com o imóvel ‘Código identificador do Imóvel’”
Pós-condição	Sugestões deletadas

Fonte: Próprio Autor. 2013.

Apêndice C – Casos de uso do Módulo Seeker

Quadro Apêndice 8 – Caso de uso Inserir Imóveis

UCS003 – Inserir Imóveis	
Descrição	Após obter todas as informações de cada imóvel, o sistema envia as informações para que o imóvel seja inserido no banco de dados relacional
Requisitos	RFS2 RNFS1 RNFS2
Ator	Sistema *Seeker*
Pré-condição	Informações do imóvel obtido
Fluxo Principal	- Sistema define a data de cadastro do imóvel com a data atual - Sistema salva o imóvel no banco de dados relacional
Fluxo Alternativo 1	- Sistema define a data de cadastro do imóvel com a data atual - Ao salvar o imóvel ocorre algum erro, sistema registra *log* no banco de dados NoSQL de falha ao salvar: “Erro ao salvar o imóvel – endereço do imóvel”
Pós-condição	O imóvel é inserido

Fonte: Próprio Autor. 2013.

Quadro Apêndice 9 – Caso de uso Atualizar imóveis

UCS004 – Atualizar imóveis	
Descrição	Após obter todas as informações de cada imóvel, o sistema envia as informações para que o imóvel já cadastrado seja atualizado no banco de dados relacional
Requisitos	RFS2 RNFS1 RNFS2
Ator	Sistema *Seeker*
Pré-condição	Informações do imóvel obtido
Fluxo Principal	- Sistema define a data de atualização do imóvel com a data atual - Sistema atualiza as informações do imóvel no banco de dados relacional
Fluxo	- Sistema define a data de atualização do imóvel com a data atual

Alternativo 1	- Ao salvar o imóvel ocorre algum erro, sistema registra *log* no banco de dados NoSQL de falha ao salvar: “Erro ao salvar o imóvel – endereço do imóvel”
Pós-condição	O imóvel é atualizado

Fonte: Próprio Autor. 2013.

Quadro Apêndice 10 – Caso de uso Deletar imóveis

UCS005 – Deletar imóveis	
Descrição	Após inserir e atualizar os imóveis da imobiliária em execução, o sistema deleta logicamente os imóveis que não foram atualizados.
Requisitos	RFS2 RNFS1 RNFS2
Ator	Sistema *Seeker*
Pré-condição	Imóveis inseridos e atualizados
Fluxo Principal	- Sistema pesquisa os imóveis que não foram atualizados - Sistema deleta logicamente os imóveis
Fluxo Alternativo 1	- Ao deletar o imóvel ocorre algum erro - Sistema registra *log* no banco de dados NoSQL com a mensagem: “Erro ao deletar imóvel – Código identificador do Imóvel”
Pós-condição	Os imóveis são deletados logicamente

Fonte: Próprio Autor. 2013.

Apêndice D – Casos de uso do Módulo Job

Quadro Apêndice 11 – Caso de uso Inserir sugestões

UCJ002 – Inserir sugestões	
Descrição	O sistema envia as novas sugestões que devem ser inseridas no banco de dados NoSQL
Requisitos	RFJ2 RNFJ2 RNFJ4
Ator	Sistema *Job*
Pré-condição	Lista de sugestões a serem inseridas
Fluxo Principal	- Sistema coloca a data de cadastro igual à data atual para as sugestões - Sistema insere as novas sugestões
Fluxo Alternativo 1	- Sistema tenta inserir as novas sugestões e ocorre um erro - Sistema registra *log* no banco de dados NoSQL com a mensagem: “Erro ao inserir sugestão para o ‘Código identificador do Usuário’ com o imóvel ‘Código identificador do Imóvel’”
Pós-condição	Sugestões inseridas

Fonte: Próprio Autor. 2013.

Quadro Apêndice 12 – Caso de uso Atualizar sugestões

UCS003 – Atualizar sugestões	
Descrição	O sistema envia as sugestões que devem ser atualizadas no banco de dados NoSQL
Requisitos	RFJ2 RNFJ2 RNFJ4
Ator	Sistema *Job*
Pré-condição	Lista de sugestões a serem atualizadas
Fluxo Principal	- Sistema coloca a data de cadastro igual à data atual para as sugestões - Sistema atualiza sugestões com as novas informações

Fluxo Alternativo 1	- Sistema tenta atualizar as sugestões e ocorre um erro - Sistema registra *log* no banco de dados NoSQL com a mensagem: “Erro ao atualizar sugestão para o ‘Código identificador do Usuário’ com o imóvel ‘Código identificador do Imóvel’”
Pós-condição	Sugestões atualizadas

Fonte: Próprio Autor. 2013.

Quadro Apêndice 13 – Caso de uso Enviar e-mail

UCS006 – Enviar e-mail	
Descrição	Sistema deve consultar a lista de e-mails que ainda não foram enviados e envia.
Requisitos	RFJ5 RNFJ2 RNFJ3 RNFJ5
Ator	Sistema *Job*
Pré-condição	E-mail pronto para ser enviado
Fluxo Principal	- Sistema obtém a lista de e-mails a serem enviados - Sistema faz o disparo do e-mail - Sistema atualiza que o e-mail foi enviado com sucesso - Registra *log* no banco de dados NoSQL com a mensagem: “Email enviado com sucesso! ‘endereço do destinatário’
Fluxo Alternativo 1	- Sistema obtém a lista de e-mails a serem enviados - Ocorre algum erro no momento do disparo do e-mail - Registra *log* no banco de dados NoSQL com a mensagem: “Email com erro! ‘endereço do destinatário’
Pós-condição	E-mail enviado para o usuário consumidor

Fonte: Próprio Autor. 2013.

Apêndice E – Questionário da pesquisa de avaliação

Disponível em: <http://goo.gl/HJuaRi> acessado em 18 outubro de 2013

Avaliação Avisa Imóvel

O Avisa Imóvel visa conectar imobiliárias e consumidores automaticamente, de forma inteligente e ativa, oferecendo bom desempenho.
Este site foi elaborado a partir do Trabalho de Conclusão do Curso de Graduação em Sistemas de Informação da Universidade Feevale, sendo desenvolvido por Lucas Demuti Dias com a orientação de Juliano Varella de Carvalho.

Gostaríamos de saber qual a sua avaliação do Avisa Imóvel por meio do seguinte questionário.

*Obrigatório

Você se cadastrou no Avisa Imóvel? *

- Sim
- Não

Você tem interesse em comprar/alugar/vender algum imóvel? *

- Sim
- Não
- Talvez

Você trabalha/trabalhou com o Mercado Imobiliário? *

- Sim
- Não

Você considera que o Avisa Imóvel atinge o seu objetivo? *

O objetivo é conectar através de uma ferramenta digital, as imobiliárias e o consumidor, de forma inteligente e ativa. A ligação entre as duas pontas do processo é realizada aplicando soluções tecnológicas que permitem o acesso rápido a informação.

- Sim
- Não
- Não sei dizer

Você considera importante a atuação do Avisa Imóvel? *

Você acredita ser necessária ou útil uma ferramenta que avise quando está disponível no mercado um imóvel conforme os seus interesses

- Sim
- Não
- Não sei dizer

Qual a relevância do Avisa Imóvel para você? *

- Muito Relevante
- Relevante
- Regular
- Pouco Relevante
- Irrelevante

Como você avalia as telas do Avisa Imóvel em relação a sua facilidade de uso? *

	Muito difícil	Difícil	Regular	Fácil	Muito Fácil
Cadastro de preferências					
Visualização de sugestões					
Visualização de Imóveis					

O Avisa Imóvel gerou sugestões de imóveis para você?

- Sim
- Não

Você recebeu os e-mails enviados pelo Avisa Imóvel? *

- Sim
- Não

As sugestões geradas para você estavam de acordo com o que você procurava? *

- Concordo totalmente
- Concordo parcialmente
- Indiferente
- Não concordo parcialmente
- Não concordo totalmente

Em relação ao desempenho/velocidade do site, você considera: *

- Muito rápido
- Rápido
- Regular
- Devagar
- Muito Devagar

Você procura imóveis em: *

- [ ] Novo Hamburgo
- [ ] Estância Velha
- [ ] Campo Bom
- [ ] São Leopoldo
- [ ] Canoas
- [ ] Outro:

Deixe os seus comentários sobre a ferramenta:

Caso deseja ser identificado, digite o seu e-mail:

Enviar