Trabalho de Conclusão de Curso

Desenvolvimento de Aplicações Web Escaláveis: O Caso do Leilão Virtual

Felipe Carlos Lima dos Santos
felipeptcho@gmail.com

Orientador:
Patrick Henrique

Maceió, Março de 2011

Felipe Carlos Lima dos Santos

Desenvolvimento de Aplicações Web Escaláveis: O Caso do Leilão Virtual

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação do Instituto de Computação da Universidade Federal de Alagoas.

Orientador:

Patrick Henrique

Maceió, Março de 2011

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação do Instituto de Computação da Universidade Federal de Alagoas, aprovada pela comissão examinadora que abaixo assina.

Patrick Henrique - Orientador
Instituto de Computação
Universidade Federal de Alagoas

Ex1 - Examinador
Instituto de Computação
Universidade Federal de Alagoas

Ex2 - Examinador
Instituto de Computação
Universidade Federal de Alagoas

Maceió, Março de 2011

Resumo

Alguns sistemas Web precisam ter um bom tempo de resposta às requisições dos navegadores dos clientes, principalmente em momentos de alta demanda. Para que eles não fiquem sobrecarregados, arquiteturas distribuídas podem ser utilizadas, contudo, o processamento das requisições ainda pode ser otimizado utilizando técnicas de *cache*. Neste trabalho, esse assunto é abordado através das etapas de desenvolvimento de um *site* de leilões virtuais, chamado Kuase de Graça. Os leilões virtuais são introduzidos, tendo como base os leilões tradicionais, em especial o Leilão Inglês. As funcionalidades do sistema são especificadas detalhadamente. O processo de desenvolvimento é descrito com a ajuda de diagramas, desde o projeto do banco de dados até a construção da arquitetura. Após isso, uma solução de otimização para o processamento das requisições é apresentada. Ao final, são apresentadas capturas de tela do sistema implementado.

Abstract

Some Web systems need a good response time for requests coming from client browsers, especially at moments of high demand. In order for these systems are not overloaded, distributed architectures can be used, however, the processing of requests can still be optimized using caching techniques. In this work, we address this topic through the development stages of an online auction site called Kuase de Graça. We introduce the online auctions, based on the traditional auctions, in particular the English Auction. The functionality of the system is specified in detail. We describe the development process with the aid of diagrams, from the database design to the architecture construction. After that, an optimization solution for the requests is presented. At the end, we present screenshots of the implemented system.

Agradecimentos

Agradeço a Deus, minha família e meus amigos. À minha namorada Vanessa Dias, por todo o apoio. Ao meu sócio Sidney Vilaça, que teve a ideia de desenvolver o Kuase de Graça e me convidou para participar do projeto. Agradeço ao professor Patrick Henrique, meu orientador, e ao meu amigo Anderson Brandão, que me apresentou o CakePHP. Agradeço a Marcelo, Regina e Flor pelas informações e pela força na secretaria do Instituto de Computação.

Em especial, agradeço a meus amigos José Oswaldo, José Cavalcante (Neto), Lucas Amorim e Rodrigo Peixoto, que colaboraram diretamente na realização deste trabalho.

Felipe Santos

"Deus nunca exige que creiamos em alguma coisa sem nos dar suficientes provas sobre que fundamentemos nossa fé. Sua existência, Seu caráter, a veracidade de Sua Palavra, baseiam-se todos em testemunhos que falam à nossa razão, e esses testemunhos são abundantes. Todavia Deus não afasta a possibilidade da dúvida. Nossa fé deve repousar sobre evidências, e não em demonstrações. Os que quiserem duvidar, hão de encontrar oportunidade; ao passo que os que desejam realmente conhecer a verdade, encontrarão abundantes provas em que basear sua fé."

Ellen G. White

Conteúdo

Lista de Figuras

1

Introdução

Nos últimos três anos, tem aparecido no mercado brasileiro um novo tipo de *site* de compras: o de leilões virtuais. O principal atrativo para o cliente é a possibilidade de obter um produto por um preço significativamente menor do que o valor de mercado, bastando, basicamente, estar conectado à Internet. Além disso, os produtos leiloados geralmente são novos, no intuito de atrair ainda mais a clientela.

Num leilão tradicional os participantes ofertam valores em dinheiro pelo produto. O ato de ofertar chama-se lance. No caso do Leilão Inglês (McAfee & McMillan 1987), que é o mais conhecido, cada oferta é maior que a anterior. Se ninguém "cobrir" a última oferta dentro de um determinado tempo, o leilão termina e é dito arrematado pelo último ofertante, que, mediante o pagamento do valor que anunciou, obtém o produto. Os outros participantes, que não arrematararam, são isentos de pagar qualquer valor.

Existe uma variedade de *sites* de leilões virtuais, cada um com suas características particulares. Porém, a maioria usa como base o Leilão Inglês, com algumas adaptações como, por exemplo, a venda do direito de fazer ofertas, ou venda de lances. Isto é, cada lance dado pelos usuários é pago. Apesar de não estar presente nos leilões clássicos, essa é a principal maneira de obtenção de receita pelas companhias.

Os *sites* que mantêm esse tipo de negócio são mais dinâmicos que o comum, e os navegadores dos clientes são instruídos a realizarem, de maneira abstrata ao usuário, várias requisições ao servidor, com o objetivo de atualizar as informações na tela. Essas requisições devem ser respondidas em um curto intervalo de tempo, a fim de cumprirem seu objetivo. Contudo, se o *site* estiver com muitos visitantes em um determinado momento, ele pode não ser capaz de realizar tal tarefa, e seus usuários terão dificuldade em participar dos leilões.

Neste trabalho são apresentadas as etapas de desenvolvimento de um *site* de leilões virtuais, chamado Kuase de Graça, junto com uma solução para otimização do processamento de requisições em ambientes de alta demanda como esse.

1.1 Organização

Este trabalho está organizado da seguinte forma:

- No Capítulo 2, o aspecto, os tipos e o funcionamento de um leilão virtual do Kuase de Graça são apresentados;
- O Capítulo 3 contém a especificação do sistema. As principais páginas e funcionalidades do *site* são detalhadas;
- No Capítulo 4 são apresentadas as etapas de desenvolvimento, incluindo o projeto do banco de dados e a arquitetura do sistema;
- O Capítulo 5 trata do balanceamento de carga e da otimização do processamento das requisições através de técnicas de *cache*;
- O Capítulo 6 destina-se a apresentar as capturas de tela do sistema implementado;
- A conclusão é apresentada no Capítulo 7.

2

Leilão Virtual do Kuase de Graça

Na Internet existem diversos *sites* de leilão e vários tipos de leilão virtual. Neste capítulo serão apresentadas as características de um leilão virtual do Kuase de Graça, como os usuários podem participar, o aspecto no navegador do usuário e o seu funcionamento.

2.1 Aquisição de Lances

Como em outros *sites* de leilão virtual, antes de fazer ofertas no Kuase de Graça, o usuário precisa adquirir lances. Cada lance dá a ele o direito de fazer uma oferta em qualquer leilão disponível no *site*. O preço de um lance é R$ 1,00, contudo, esse valor pode diminuir devido a promoções ou se o cliente adquirir uma quantidade maior de lances. Isso ficará a critério da equipe de administração do *site*.

2.2 Tipos de Leilão Virtual

Existem dois tipos principais de leilão virtual no Kuase de Graça: progressivo e regressivo.

2.2.1 Leilão Progressivo

Esse tipo de leilão é usado na maioria dos *sites* de leilão existentes na Web. Ele é baseado no Leilão Inglês (McAfee & McMillan 1987). Nesse tipo de leilão, o valor de cada oferta é sempre R$ 0,01 maior que o preço atual do leilão. O preço inicial do leilão é R$ 0,00 (zero real).

2.2.2 Leilão Regressivo

O preço inicial desse tipo de leilão é 1% do preço de mercado sugerido do produto[1]. O valor de cada oferta é sempre R$ 0,01 menor que o preço atual do leilão. Se o preço final do leilão for negativo, esse valor é arredondado e convertido em lances para o arrematante. Por exemplo, se o leilão terminar no valor de R$ -2,01, esse valor é arredondado para R$ -3,00, que equivale a 3 lances para o arrematante.

2.3 Aparência do Leilão no Navegador do Usuário

O aspecto de um leilão virtual no navegador do usuário é apresentado na Figura 2.1.

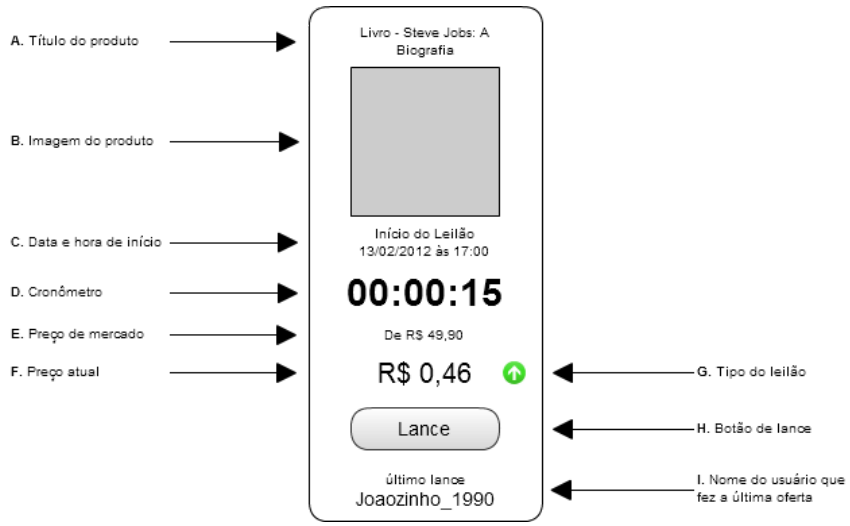

Figura 2.1: Aspecto de um leilão virtual no navegador do usuário.

- Os itens A e B são, respectivamente, o nome e a imagem do produto que está sendo leiloado;
- O item C é a data e a hora marcadas para o início do leilão;
- O item D é o cronômetro regressivo que indica quanto tempo os participantes têm para fazer uma nova oferta. Ele só começa a contar quando o leilão inicia, e é reiniciado após cada oferta. Se o cronômetro zerar (nenhum usuário tiver feito uma nova oferta), o leilão é arrematado;
- O item E é o preço de mercado sugerido do produto;
- O item F é o preço atual do leilão (também chamado Valor para Arremate), ou seja, o valor a ser pago pelo arrematante, caso o leilão termine. A cada lance, esse valor aumenta ou diminui R$ 0,01, dependendo do tipo do leilão (progressivo ou regressivo);

[1] O preço de mercado considerado ficará a critério da equipe de administração do *site*, que se baseará no valor do produto em diversas lojas onde ele pode ser adquirido.

- O item G é o indicador de tipo do leilão. Uma seta para cima significa que o leilão é progressivo. Uma seta para baixo significa que o leilão é regressivo;
- O item H é o botão a ser clicado pelos participantes que desejem fazer uma oferta. Quando o leilão termina, não é possível fazer novas ofertas e esse item desaparece. No seu lugar, surge a frase:
  - "É seu", se o usuário arrematou o leilão;
  - "Arrematado", se outro usuário arrematou o leilão;
  - "Finalizado", se não há arrematante, ou seja, ninguém fez uma oferta;
- O item I apresenta o nome do último ofertante, ou seja, o usuário que irá arrematar (ou já arrematou) o leilão, caso não haja uma nova oferta.

2.4 Funcionamento

O diagrama do funcionamento de um leilão virtual é apresentado na Figura 2.2. No primeiro estado, o leilão não está iniciado porque a data e a hora marcadas ainda não foram alcançadas, mas ele já pode receber ofertas. Uma vez alcançada a hora marcada, o leilão inicia e o seu cronômetro começa a contagem regressiva. Quando a contagem chega a zero, o leilão termina. Iniciado ou não, ao receber uma oferta, o leilão passa para o estado de validação da mesma. Nesse estado, é verificado se o usuário tem lances suficientes e se está habilitado a participar do leilão[2]. Se a oferta for válida, ela é aceita e as seguintes operações são feitas:

- um lance é subtraído do total de lances do usuário ofertante;
- o preço atual do leilão (item F da Figura 2.1) é acrescido ou decrescido de R$ 0,01, dependendo do tipo do leilão (progressivo ou regressivo);
- o nome do último ofertante (item I da Figura 2.1) é atualizado com o nome do usuário ofertante;
- se o leilão estiver iniciado, o cronômetro recomeça a contagem, dando oportunidade para outro usuário fazer uma nova oferta.

[2]Também é verificado se o usuário não estará "cobrindo" o próprio lance, pois não é permitido.

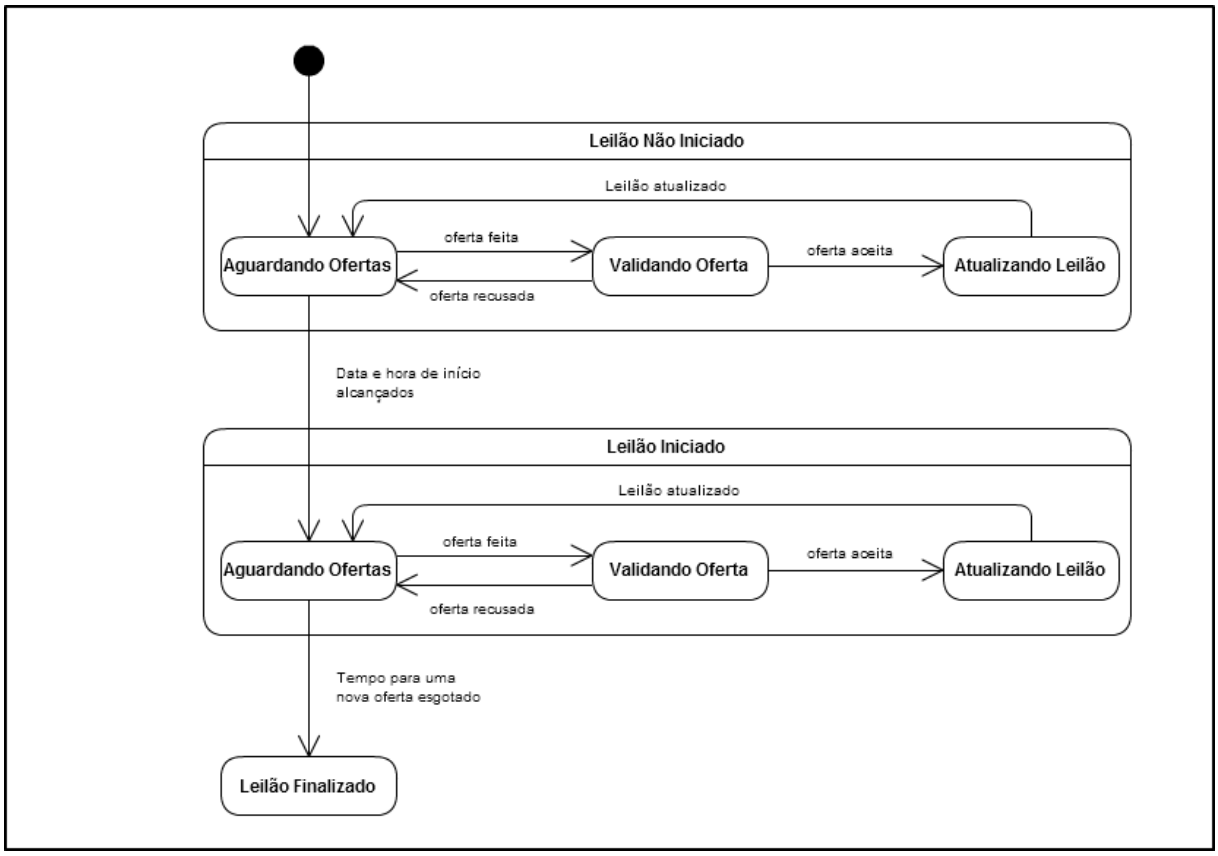

Figura 2.2: Diagrama de funcionamento de um leilão virtual.

3

Especificação do Sistema

Neste capítulo trata-se de detalhar o aspecto e as principais funcionalidades do *site* Kuase de Graça. Todo o desenvolvimento é baseado nas especificações aqui descritas.

3.1 Cadastro de Usuários e Autenticação

O Kuase de Graça é um sistema multiusuário, ou seja, vários usuários podem estar acessando-o num certo momento. Para atender determinadas solicitações, o sistema precisa identificar o usuário que fez a requisição. Por exemplo, ao receber da Internet uma solicitação de compra de trinta lances, é necessário saber quem fez o pedido, de maneira que os lances sejam atribuídos ao usuário correto. Por esse motivo, para participar das principais atividades do *site*, os usuários precisarão realizar um cadastro, informando, entre outros dados, um *e-mail* e uma senha, de forma que o sistema possa identificá-los corretamente através do processo de autenticação (Figura 3.1).

O processo de autenticação consiste em determinar se alguém é realmente quem ele alega ser. Sempre que necessário, o sistema irá solicitar informações que, teoricamente, só o usuário verdadeiro pode fornecer. Essas informações geralmente são um *login* e uma senha. O *login* serve para identificar o usuário em meio a outros. A senha, informada corretamente, prova que o usuário é quem ele diz ser[1].

Cada usuário deve possuir um *login* único no sistema. No Kuase de Graça, o campo de *e-mail* informado no cadastro será usado como o *login*. Ele é suficiente para isso, pois não será permitido que dois usuários sejam cadastrados com o mesmo *e-mail* no sistema.

Além do *e-mail* e senha, o formulário de cadastro requisitará que o usuário aceite os termos de uso (Seção 3.16) e preencha alguns dados pessoais, bem como o endereço para onde devem ser enviados os produtos que ele arrematar. Todos os campos que devem ser informados pelo usuário podem ser divididos em 3 grupos:

[1] É assumido que o conhecimento da senha garante a autenticidade desse usuário.

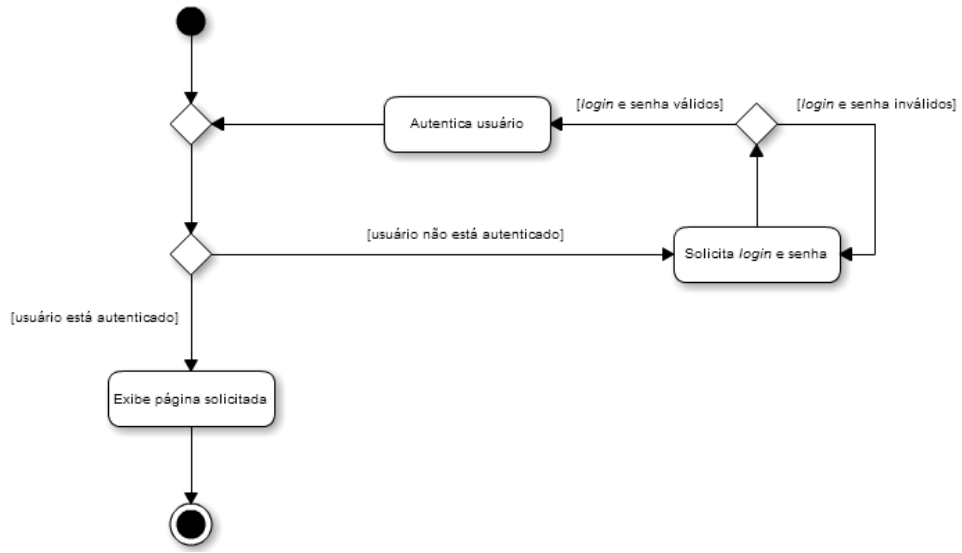

Figura 3.1: Processo de autenticação de um usuário quando ele requisita uma página restrita a usuários autenticados.

- **Dados Gerais**:
  - Nome Completo;
  - Sexo;
  - Telefone;
  - Data de Nascimento;
  - CPF;
- **Dados para identificação no sistema**
  - Nome de Usuário;
  - *E-mail*;
  - Senha;
- **Dados para Entrega de Produtos**
  - CEP;
  - Endereço;
  - Número;
  - Complemento (opcional);
  - Bairro;
  - Cidade;
  - Estado.

O campo “Nome de Usuário” é o nome que irá aparecer nos leilões em que esse usuário fizer uma oferta (item I da Figura 2.1). Ele deve ter entre quatro e quinze caracteres. É obrigatório que esses caracteres sejam alfanuméricos ou *underscores* (_).

Após a submissão do formulário, o usuário terá sua conta no Kuase de Graça criada. Ela conterá inicialmente 5 lances, contudo, antes de poder acessá-la, ele deverá ativar seu cadastro, conforme explicado na Seção 3.3.

3.2 Redefinição de Senha

Se ocorrer de um usuário esquecer a sua senha, ele ficará impossibilitado de acessar o sistema. Para reverter essa situação, ele poderá solicitar a redefinição de sua senha. O sistema pedirá que ele se identifique, informando seu *login*. Se o *login* informado realmente estiver cadastrado, uma mensagem será enviada para o endereço de *e-mail* desse usuário com um *link* que, quando acessado, permitirá que ele crie uma nova senha. Esse *link* deve ser único, de uma forma que não possa ser reproduzido por usuários mal intencionados. Ele também deve ser invalidado após a nova senha ter sido criada ou após três dias da solicitação.

Mesmo que não tenham esquecido a senha, os usuários poderão solicitar a alteração da mesma sem terem que receber um *e-mail* com o *link*. Basta que eles informem a senha atual e a nova senha desejada.

3.3 Ativação de Cadastro

É necessário garantir que um usuário que está se cadastrando é capaz de receber mensagens no endereço de *e-mail* informado. Se, por exemplo, o *e-mail* não existir e o usuário precisar redefinir sua senha, ele não conseguirá concluir o processo, pois não receberá a mensagem com o *link* para a criação da nova senha.

Para evitar esse problema, logo após o cadastro, uma mensagem de confirmação será enviada para o endereço de *e-mail* informado. Ela conterá um *link* que o usuário deve acessar. Isso irá confirmar ao sistema que o usuário é capaz de receber mensagens. Somente após essa confirmação, o usuário terá sua conta ativada e poderá acessá-la.

3.4 Convite de Usuários

O sistema deve fornecer meios pelos quais os usuários possam facilmente convidar seus amigos para participarem do *site*. Após a ativação de seu cadastro, o usuário receberá acesso a um *link* pessoal que ele poderá distribuir entre seus contatos. Todos os amigos que acessarem esse *link* e se cadastrarem (e realizarem a ativação do cadastro) serão considerados convidados desse usuário e poderão ser visualizados por ele numa lista específica. Essa lista

estará acessível através da página “Minha Conta” (Seção 3.13). Para cada convidado que realizar sua primeira compra de lances, o usuário que convidou receberá uma recompensa de dez lances.

Integradas ao sistema, devem estar ferramentas que os usuários poderão utilizar para distribuir seus *links* pessoais de convite nas redes sociais e via *e-mail* com facilidade.

3.5 Uso de CAPTCHA

CAPTCHA (*Completely Automated Public Turing Test to Tell Computers and Humans Apart*) é o mecanismo que usa imagens geradas aleatoriamente, contendo palavras ou caracteres, com o intuito de verificar se um usuário é humano (von Ahn et al. 2004). Um exemplo de CAPTCHA com caracteres pode ser visto na Figura 3.2. Esse mecanismo é usado, por exemplo, em alguns formulários que possuem certo nível de vulnerabilidade a programas de computador maliciosamente implementados para executar uma determinada tarefa diversas vezes por segundo.

No caso do Kuase de Graça, um usuário mal intencionado poderia programar um algoritmo para submeter repetidas vezes o formulário de *login* do *site*, utilizando várias combinações de senha, no intuito de descobrir a senha de outro usuário e poder se autenticar como ele no sistema. Por evitar isso, no momento do *login*, um usuário terá 5 tentativas para acertar sua senha. Se não acertar, ele poderá fazer novas tentativas, contudo, digitando corretamente a sequência de caracteres ou palavras do CAPTCHA, provando não ser um algoritmo malicioso.

Figura 3.2: Exemplo de CAPTCHA.

3.6 Compra de Lances

Inicialmente, a conta de cada usuário possui apenas 5 lances. Mais lances podem ser adquiridos através de uma página de compra. Os lances não poderão ser comprados individualmente, mas serão vendidos em “pacotes”. Cada pacote possui uma quantidade de lances e preço específicos. Após selecionar o pacote desejado, o usuário será redirecionado a uma próxima página, onde poderá escolher a forma de pagamento desejada e dar as informações

financeiras necessárias. Se o pagamento for aprovado, os lances são adicionados à sua conta no Kuase de Graça.

O usuário ainda poderá visualizar numa página, o andamento de todos os pagamentos feitos para o Kuase de Graça.

O processo de pagamento deverá ficar por conta de uma solução para pagamentos online, o PagSeguro[2]. Desse modo, além de contar com várias formas de pagamento, o usuário não precisará se preocupar com o sigilo de seus dados financeiros, pois eles não terão que ser informados diretamente ao Kuase de Graça. E ainda contará com a garantia de que o seu dinheiro será devolvido na ocasião de algum problema. Além disso, a adição de novas formas de pagamento é feita pelo PagSeguro de maneira abstrata para o Kuase de Graça, e os pagamentos feitos a prazo pelos clientes são recebidos de uma só vez pelo *site*.

3.7 Códigos Promocionais

O sistema deverá fornecer um meio de gerar códigos (ou cupons) de desconto para os usuários utilizarem na compra de pacotes de lances. Esses códigos podem ser digitados pelos usuários no momento da compra ou serem enviados por *e-mail* na forma de *links*. Após a utilização de um código de desconto, o mesmo deverá ser invalidado, evitando o seu reuso. Somente se a transação de pagamento não for aprovada pela instituição financeira, o código poderá voltar a ser usado pelo usuário para uma nova tentativa.

3.8 Depoimentos

Quando um usuário receber em sua residência um produto que ele arrematou no Kuase de Graça, ele poderá enviar um depoimento por *e-mail*. Em troca, ele receberá vinte lances. Um depoimento consiste numa foto do usuário com o produto e num texto que conte a experiência que ele teve no *site*.

Uma página deverá reunir todos os depoimentos recebidos dos usuários, ordenados pela data do recebimento.

3.9 *Layout* das Páginas

O *layout* padrão para as páginas do *site* deve seguir o formato apresentado na Figura 3.3.

- O item A é o cabeçalho do *site*. Ele deve conter a logomarca e um formulário (*e-mail* e senha) para *login* dos usuários. Se o usuário já estiver autenticado no sistema, ao invés do formulário de *login*, a quantidade de lances atual dele deve ser mostrada em

[2] https://pagseguro.uol.com.br/

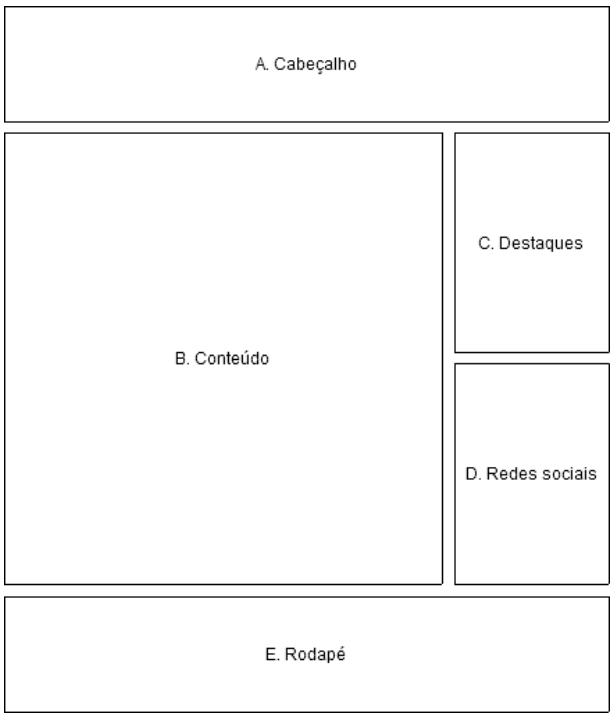

Figura 3.3: *Layout* padrão para as páginas do Kuase de Graça.

destaque, junto com o nome de usuário, um *link* para a realização do *logout*, outro para a página de compra de lances e outro para a página "Minha Conta" (Seção 3.13), conforme a Figura 3.4. Se a quantidade de lances do usuário for menor que cinquenta, ela deve ser mostrada na cor vermelha, com o objetivo de alertar o usuário da baixa quantidade;

- O item B é o conteúdo específico da página;
- O item C é um espaço para mostrar no máximo 2 leilões atualmente em destaque no *site*, ordenados por data e hora de início dos mesmos;
- O item D é um espaço para *plug-ins* de redes sociais. Pode ser vista na Figura 3.5, por exemplo, a aparência de um *plug-in* do Twitter[3] que mostra as últimas atualizações de um determinado perfil, e do *plug-in Like Box*[4] do Facebook[5], que mostra a quantidade de pessoas que "curtiram" uma dada página dessa rede;
- O item E é o rodapé do *site*, que deve conter uma lista de *links* para diversas páginas do *site*.

[3] http://twitter.com/
[4] http://developers.facebook.com/docs/reference/plugins/like-box/
[5] http://www.facebook.com/

Figura 3.4: Informações mostradas no cabeçalho do *site* quando o usuário está autenticado.

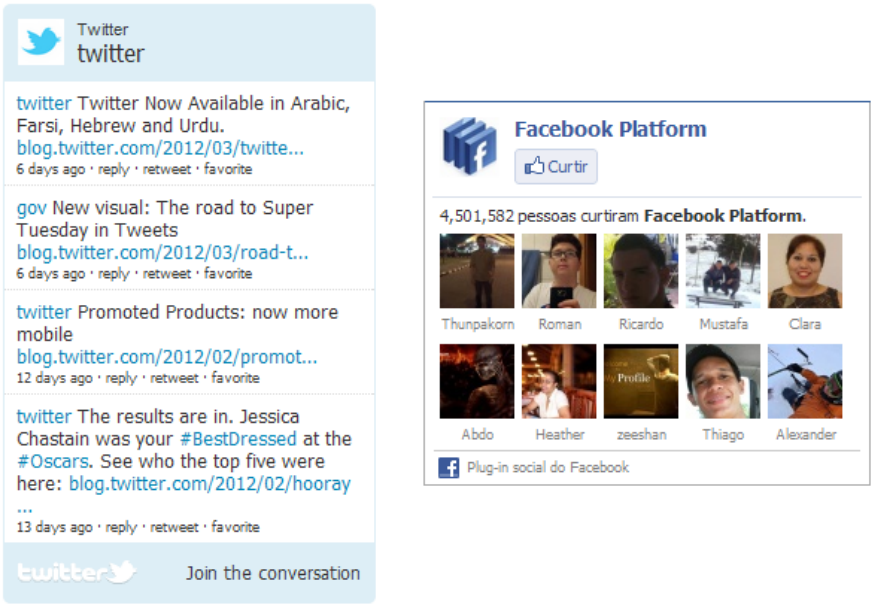

Figura 3.5: Um *plug-in* do Twitter à esquerda e um do Facebook à direita.

3.10 Página Inicial

O conteúdo da página inicial deve ser composto por no máximo 8 leilões que ainda vão começar ou que estão em execução, ordenados por data e hora de início. Se não existir leilões cadastrados no sistema, uma mensagem explicativa deve ser mostrada. Cada leilão deve seguir o formato apresentado na Figura 2.1.

3.11 Detalhes de um Leilão

O espaço para cada leilão na página inicial é limitado e nem todas as informações do mesmo podem ser exibidas, por isso todo leilão deverá ter uma página específica onde o usuário poderá visualizar o histórico dos dez últimos usuários que fizeram uma oferta e os detalhes do produto que está sendo leiloado (mais imagens e informações técnicas), conforme a Figura 3.6.

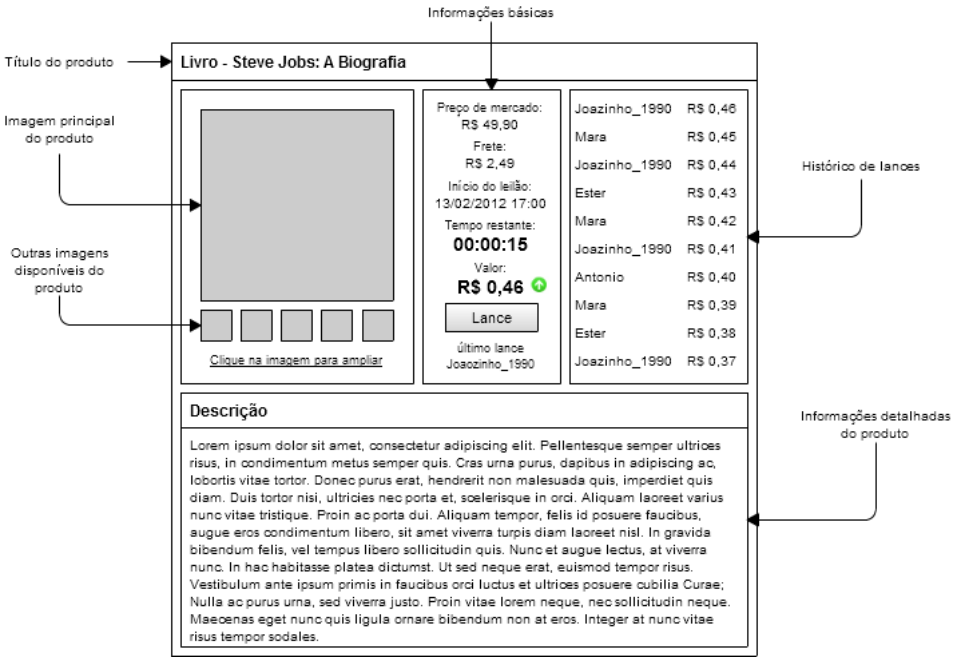

Figura 3.6: Página de detalhes de um leilão.

3.12 Leilões Arrematados

A página inicial exibe somente os leilões que vão iniciar ou que estão em execução (Seção 3.10). Os leilões que já foram arrematados devem ser exibidos numa página específica, ordenados por data e hora de arremate, conforme a Figura 3.7. Os leilões que finalizaram sem um arrematante não devem aparecer nessa página.

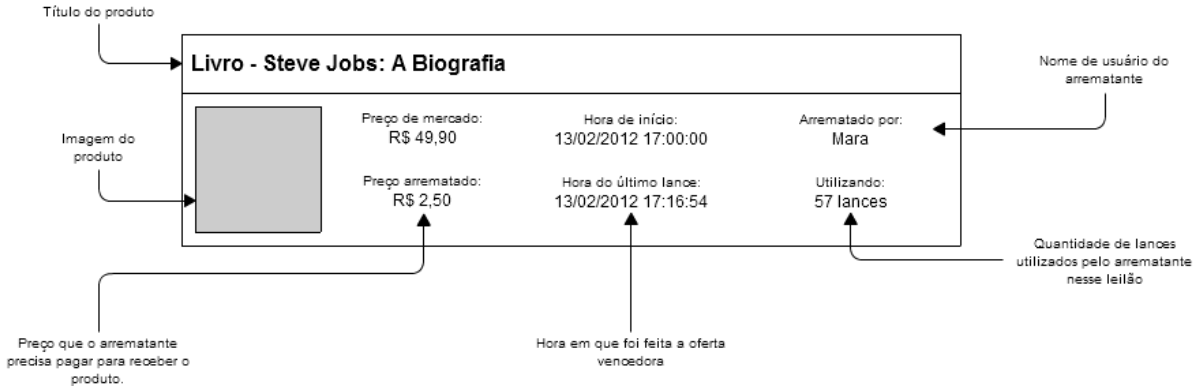

Figura 3.7: Aspecto de um leilão na página de leilões arrematados.

3.13 Página "Minha Conta"

É onde o usuário deverá ter acesso a informações pertinentes à sua conta no Kuase de Graça. Ele poderá:

- enviar convites para seus contatos;

- visualizar seus convidados;
- visualizar o extrato de seus lances;
- adquirir mais lances;
- visualizar leilões que ele participou;
- visualizar leilões que ele arrematou;
- realizar o pagamento dos leilões arrematados;
- visualizar seus pagamentos;
- alterar seus dados de cadastro.

3.14 Página “Fale Conosco”

Trata-se de uma página que os usuários podem usar para entrar em contato com a equipe de atendimento do Kuase de Graça. Essa página deve conter um formulário para o pre-enchimento do nome, *e-mail*, assunto e mensagem do usuário. Quando o formulário for submetido, esses campos devem ser enviados por *e-mail* para a equipe de atendimento do *site*.

3.15 Simulador

Os leilões virtuais são novidade para muitas pessoas. Por isso, é necessário a existência de uma página para demonstrar como eles funcionam. Através de um simulador, os usuários podem participar de um leilão fictício e tirar a maioria de suas dúvidas por conta própria. O simulador deve:

- permitir que os usuários escolham que tipo de leilão eles desejam simular (progressivo ou regressivo);
- explicar as partes de um leilão como mostrado na Figura 2.1;
- explicar como funciona o tipo de leilão escolhido (Seção 2.2);
- iniciar o leilão e instruir o usuário fazer uma oferta.

Quando o leilão terminar, o simulador deve explicar por quanto o produto foi arrematado e como o usuário deveria proceder para recebê-lo em sua residência.

3.16 Termos de Uso

São as regras de utilização do *site*. Especificam quem pode se cadastrar, o funcionamento dos leilões, o procedimento para pagamento e entrega dos produtos arrematados e as responsabilidades do *site* para com seus usuários. Elas devem ser mostradas no momento do cadastro e todo usuário deve concordar com as mesmas para poder criar sua conta. Também deve existir uma página específica contendo os termos de uso para que os usuários possam consultar posteriormente.

3.17 Administração

O sistema deve possuir uma área administrativa designada para usuários com a devida autorização. Essa área deve permitir:

- Cadastrar, editar e excluir códigos promocionais (Seção 3.7);
- Cadastrar, editar e excluir os depoimentos enviados por usuários que receberam produtos arrematados (Seção 3.8);
- Cadastrar, editar e excluir os produtos do *site*;
- Cadastrar, editar e excluir os leilões. O cadastro dos leilões deve ser feito a partir dos produtos;
- Reiniciar um determinado leilão, devolvendo todos os lances dos participantes e o agendando para uma nova data. Essa funcionalidade pode ser útil se houver algum problema na execução do leilão que o invalide. Por exemplo, se o *site* saísse do ar, ficaria impossibilitado de receber ofertas e, consequentemente, os leilões em execução seriam finalizados, sem que os participantes tivessem oportunidade "cobrir" a última oferta computada;
- Cadastrar, editar e excluir os pacotes de lances a serem vendidos na página de compra de lances (Seção 3.6).

Os usuários que têm acesso à área administrativa são semelhantes a usuários clientes, contudo, eles não podem fazer ofertas nos leilões do *site*.

3.18 Casos de Uso do Sistema

De acordo com a Seção 3.17, existem dois tipos de usuário no sistema: usuário cliente e administrador. É apresentado na Figura 3.8, o diagrama de casos de uso de um usuário genérico do sistema (cliente ou administrador).

O diagrama de casos de uso para um usuário cliente do sistema, que não tem privilégios administrativos, pode ser visto na Figura 3.9. Pode-se notar pelo diagrama que ele herda os casos de uso do usuário genérico e tem a capacidade de fazer ofertas.

Já o diagrama de casos de uso para um usuário que é administrador do sistema, é apresentado na Figura 3.10. Ele também herda os casos de uso do usuário genérico, mas ao contrário do cliente, não pode participar dos leilões.

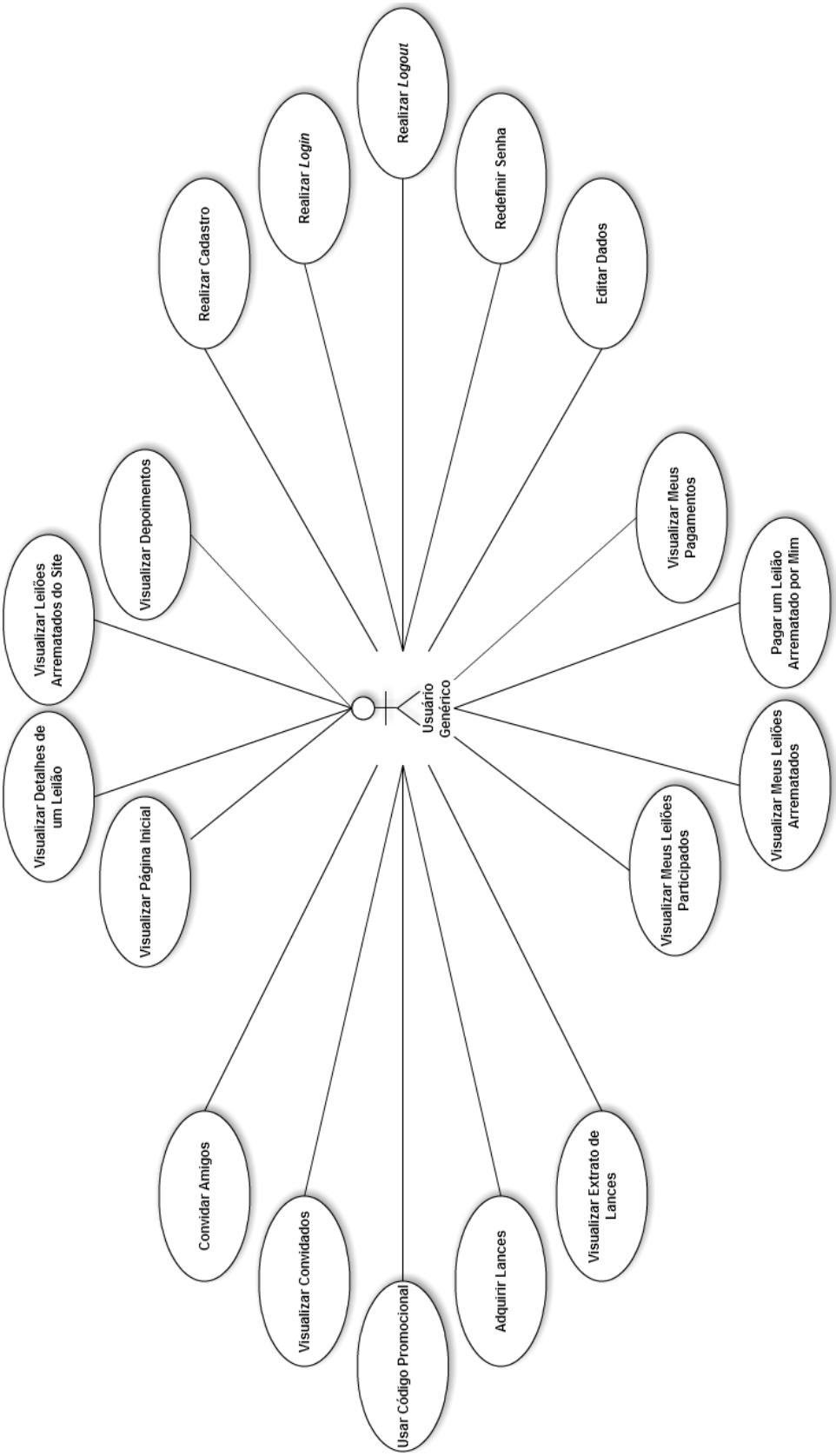

Figura 3.8: Diagrama de casos de uso de um usuário genérico do sistema.

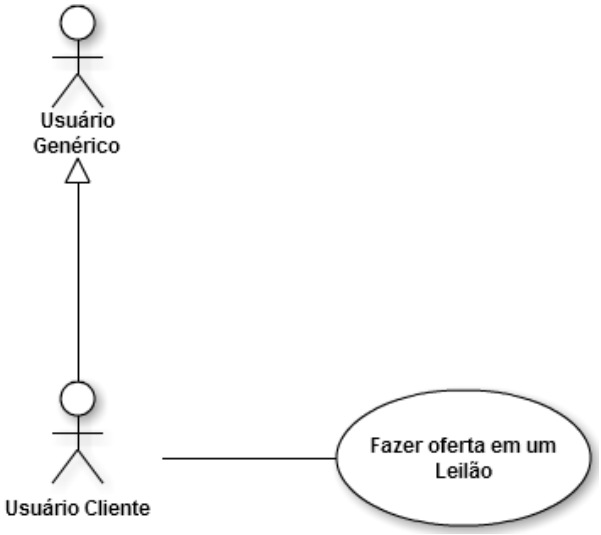

Figura 3.9: Diagrama de casos de uso de um usuário cliente do sistema.

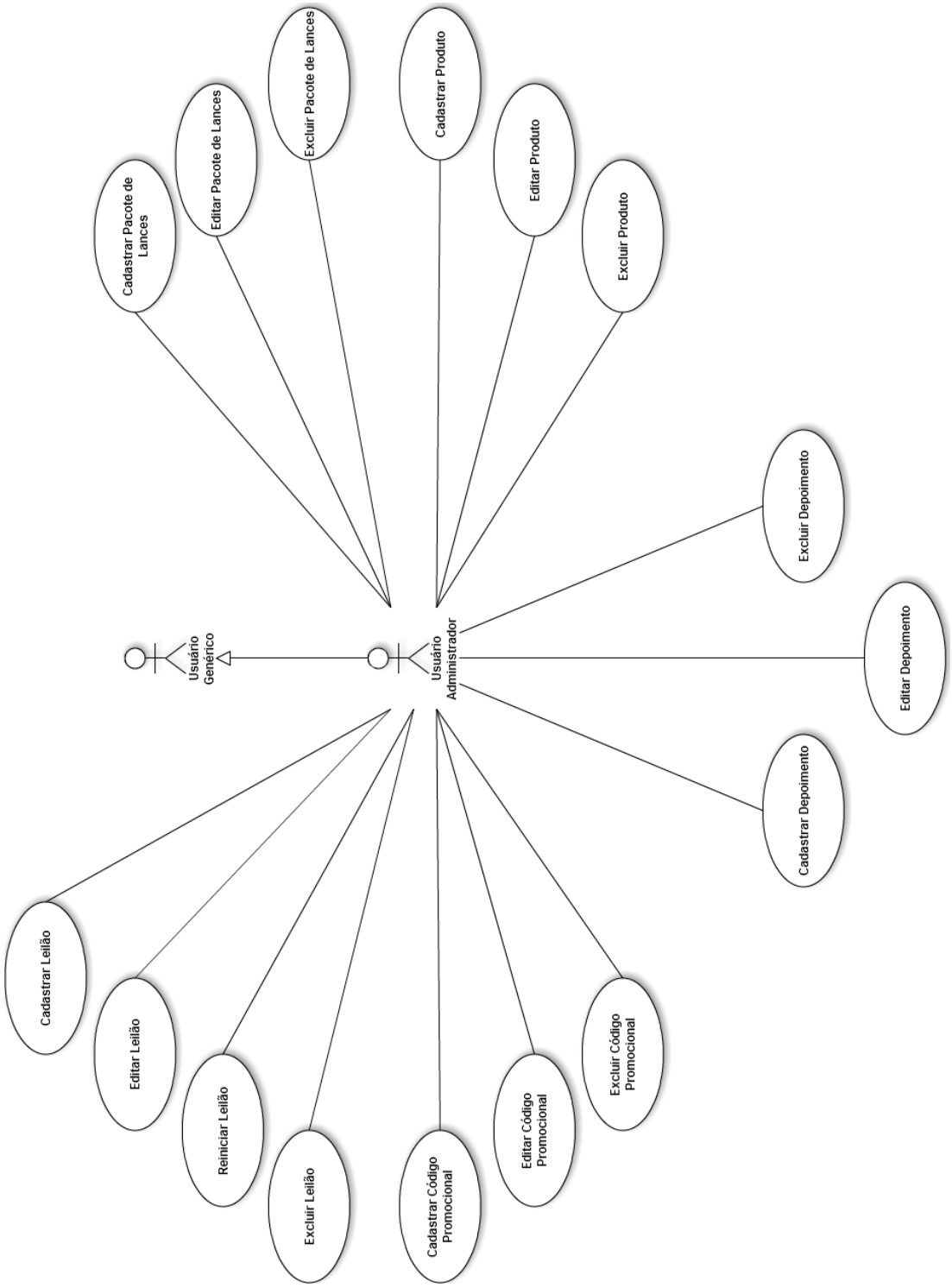

Figura 3.10: Diagrama de casos de uso de um usuário administrador do sistema.

4

Desenvolvimento

Serão apresentadas neste capítulo as etapas do desenvolvimento do Kuase de Graça, baseadas nas especificações do capítulo anterior.

4.1 Projeto de Banco de Dados

4.1.1 Diagrama Entidade Relacionamento

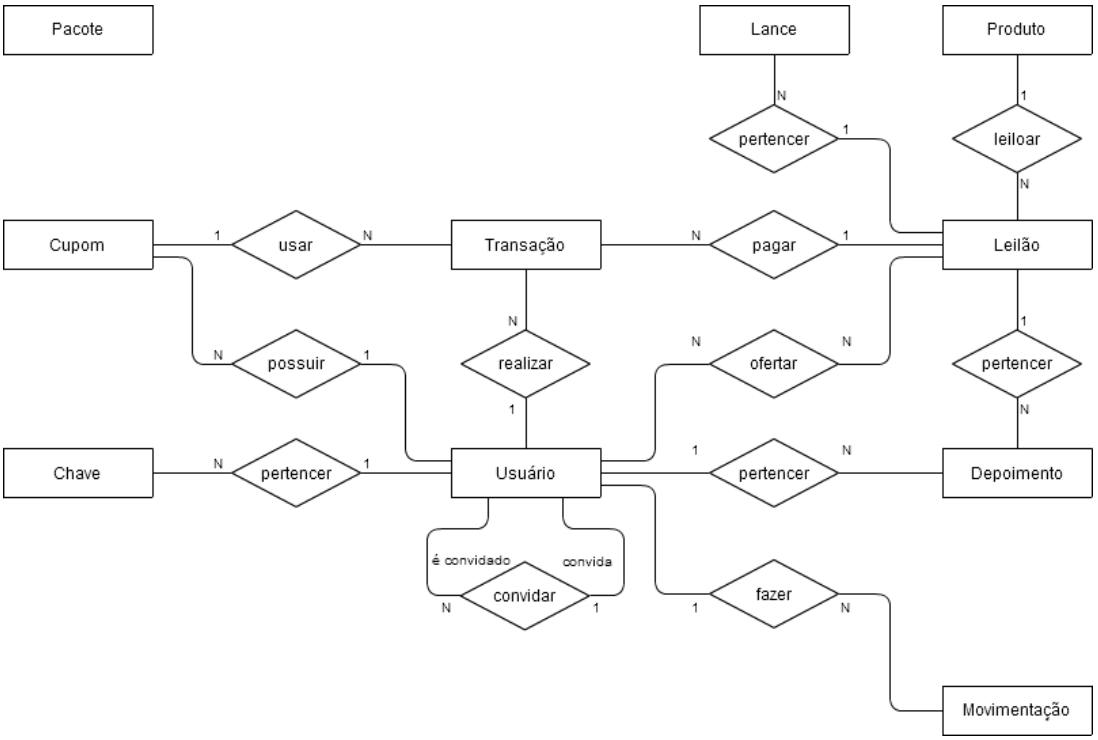

Figura 4.1: Diagrama entidade relacionamento sem os atributos.

O diagrama entidade relacionamento baseado na especificação do sistema no capítulo anterior pode ser visto na Figura 4.1. Esse diagrama é fundamentado num modelo de dados que tem como objetivo representar conceitualmente os dados de um sistema de informação. Ele consegue incorporar algumas das informações semânticas sobre o mundo real (Chen 1975), o que o torna uma ferramenta ideal para o projeto do banco de dados desse sistema. Nessa figura, os atributos foram ocultados, contudo, serão apresentados os atributos das entidades mais importantes: Usuário e Leilão.

Entidade Usuário

Essa entidade representa um usuário do sistema. Seus atributos são apresentados na Figura 4.2.

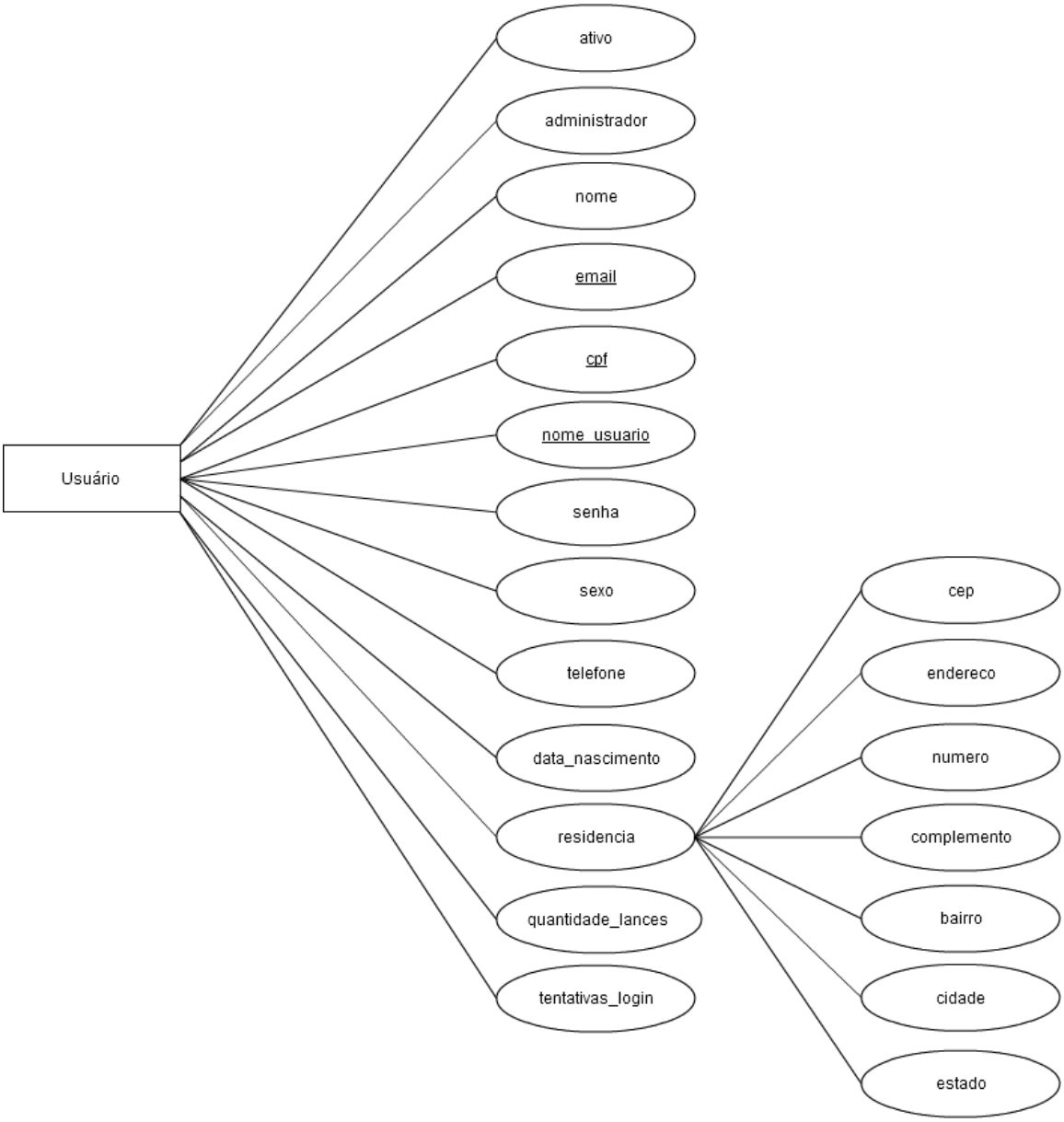

Figura 4.2: Atributos da entidade usuário.

A maioria dos atributos dessa entidade são dados de cadastro do usuário. O restante é explicado a seguir:

- `ativo`: indica se o usuário já fez a ativação do seu cadastro (Seção 3.3);
- `administrador`: indica se o usuário pode administrar o sistema (Seção 3.17);
- `quantidade_lances`: armazena a quantidade de lances que o usuário possui para participar dos leilões;
- `tentativas_login`: indica quantas vezes o usuário tentou realizar login e não conseguiu, com o objetivo de requisitar a digitação de um CAPTCHA (Seção 3.5).

Entidade Leilão

Representa um leilão do sistema. Seus atributos são apresentados na Figura 4.3.

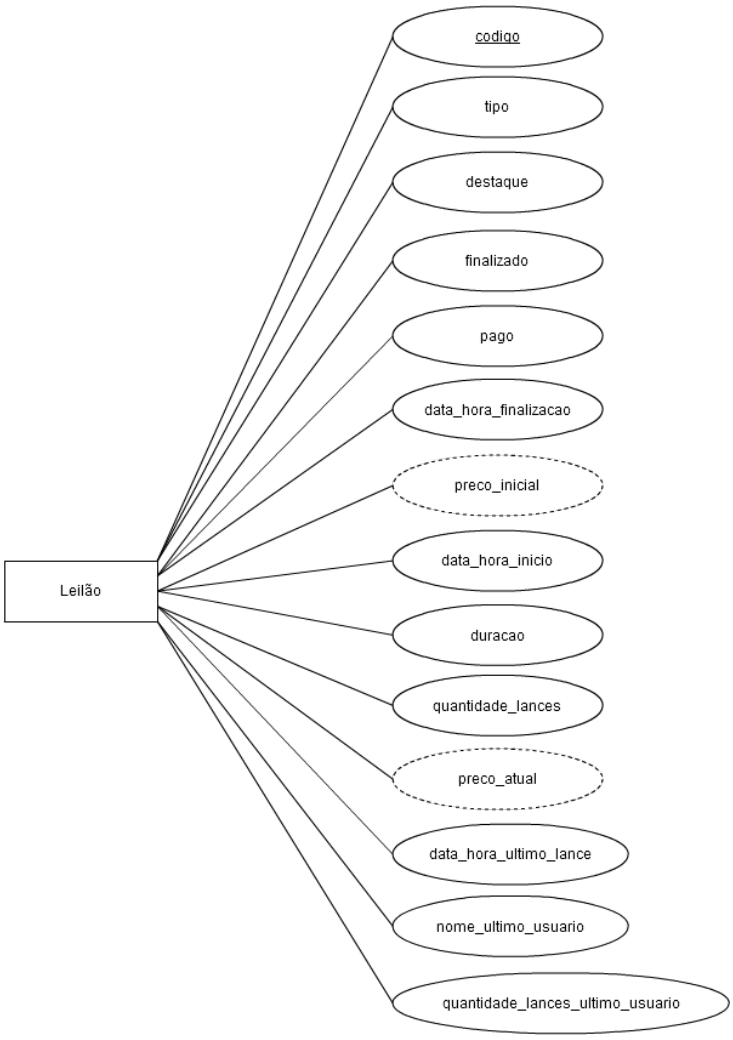

Figura 4.3: Atributos da entidade leilão.

- `tipo`: informa se o leilão é progressivo ou regressivo;
- `destaque`: informa se o leilão pode aparecer no espaço de leilões em destaques (item C da Figura 3.3);
- `finalizado`: informa se o leilão foi arrematado, encerrou sem arrematante, ou se não iniciou, de modo a ser exibido na página inicial ou na página de leilões arrematados;
- `pago`: informa se o arrematante já realizou o pagamento desse leilão;
- `data_hora_finalizacao`: informa a data e a hora em que o cronômetro do leilão zerou;
- `preco_inicial`: informa o preço do leilão antes do mesmo receber qualquer oferta. Esse valor é deduzido a partir do tipo do leilão e do preço de mercado sugerido do produto (Seção 2.2);
- `data_hora_inicio`: informa a data e hora marcadas para o início do leilão;
- `duracao`: informa o tempo estabelecido para o cronômetro do leilão. A cada lance, o cronômetro voltará a esse valor;
- `quantidade_lances`: informa quantos lances o leilão já recebeu;
- `preco_atual`: informa o preço do leilão de acordo com o preço inicial, o tipo do leilão e a quantidade de lances recebida. Quando o leilão terminar, esse será o valor a ser pago pelo arrematante (se o preço for positivo) ou a ser devolvido a ele na forma de lances (se o preço for negativo);
- `data_hora_ultimo_lance`: informa a data e a hora do último lance recebido no leilão. Esse será o lance vencedor se não houver novas ofertas até o cronômetro zerar. Essa informação será usada na visualização desse leilão na página de leilões arrematados, conforme a Figura 3.7;
- `nome_ultimo_usuario`: informa o nome de usuário do dono do último lance recebido no leilão. Essa informação será usada no item I da Figura 2.1 e na visualização desse leilão na página de leilões arrematados;
- `quantidade_lances_ultimo_usuario`: informa a quantidade de lances do dono do último lance recebido no leilão. Após o arremate, essa informação será usada na visualização desse leilão na página de leilões arrematados.

4.1.2 Modelo Lógico

Um modelo lógico é uma descrição de um banco de dados no nível de abstração visto pelo usuário do SGBD (Heuser 2008). Neste projeto, utilizou-se um SGBD relacional (Seção 4.2), onde os dados são organizados na forma de tabelas. Pode ser vista na Figura 4.4, a estrutura das tabelas projetada a partir do diagrama entidade relacionamento.

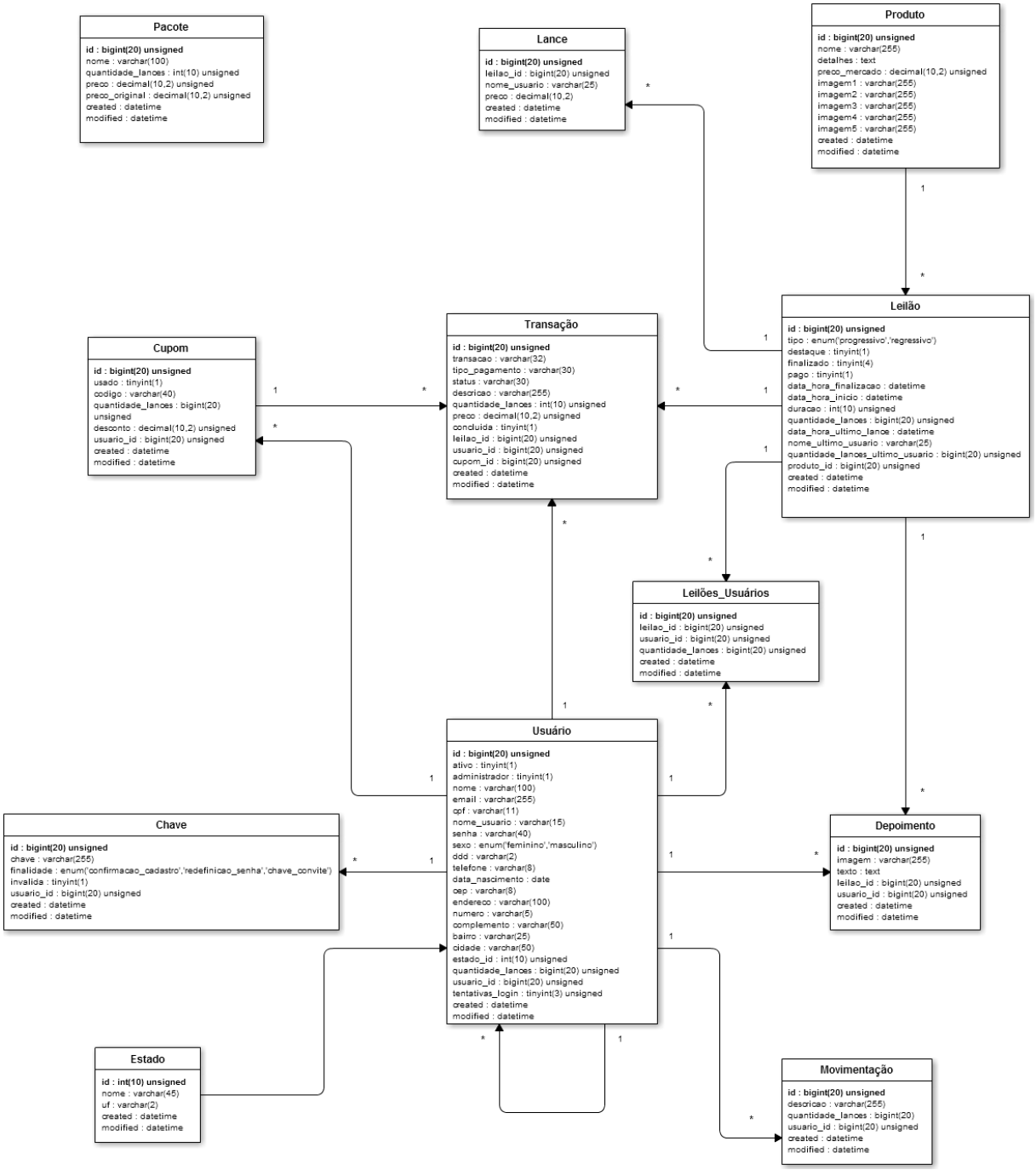

Figura 4.4: Estrutura das tabelas do banco de dados.

4.2 Tecnologias Utilizadas

Para o desenvolvimento do Kuase de Graça, foram utilizadas as seguintes tecnologias:

- Sistema Operacional: Ubuntu Server 10.04 LTS[1]
  (http://www.ubuntu.com/);
- Servidor HTTP: Apache 2.2.14
  (http://www.apache.org/);
- Linguagem de programação do lado servidor: PHP[2] 5.3.8 (Welling & Thomson 2005, Schlossnagle 2004)
  (http://www.php.net/);
- *Framework* para desenvolvimento Web: CakePHP 1.2.11
  (http://cakephp.org/);
- Editor de código fonte: Notepad++ 5.9.6.2
  (http://notepad-plus-plus.org/);
- Sistema de Gerenciamento de Banco de Dados (SGBD): MySQL 5.1.61
  (http://www.mysql.com/);
- Administrador de banco de dados: phpMyAdmin 3.3.2
  (http://www.phpmyadmin.net/);
- Linguagem de programação do lado cliente: JavaScript (Goodman & Morrison 2007)
  (https://developer.mozilla.org/en/JavaScript);
- Linguagem de estilo: CSS[3] 2.1
  (http://www.w3.org/Style/CSS/);
- Serviço de CAPTCHA: reCAPTCHA
  (http://www.google.com/recaptcha);
- Teste de carga: ApacheBench 2.3
  (http://httpd.apache.org/docs/2.0/programs/ab.html).

[1] *Long-Term Support.*
[2] *PHP: Hypertext Preprocessor.*
[3] *Cascading Style Sheets.*

4.3 Características do *Framework* CakePHP

O CakePHP é um *framework* de desenvolvimento rápido[4] para PHP. Seu objetivo principal é permitir que o trabalho seja feito de uma maneira rápida e estruturada, contudo, sem perda da flexibilidade. O programador conta com todas as ferramentas que precisa para começar a programar o que realmente o interessa: a lógica específica de sua aplicação, ao invés de, a cada novo projeto, ter que "reinventar a roda" (envio de e-mails, *cookies* do navegador, segurança, gerenciamento de sessão e tratamento de requisições).

As vantagens de utilizar o CakePHP incluem:

- Comunidade ativa e amigável;
- Licença flexível, abrangendo *software* livre e proprietário;
- Geração de código baseada na estrutura das tabelas do banco de dados;
- Arquitetura MVC (Seção 4.3.1);
- Validação embutida;
- Mecanismos de *cache* flexíveis;
- Pouca ou nenhuma configuração no Apache envolvida.

4.3.1 Arquitetura MVC

O CakePHP segue o padrão de projeto de *software* MVC (Burbeck 1997), que separa a aplicação em três camadas principais.

Model

Pode ser vista como a primeira camada de interação com o banco de dados. Responsável por recuperar dados do banco de dados, convertê-los em conceitos significantes para a aplicação, bem como processar, validar, associar e qualquer outra tarefa relativa ao tratamento desses dados.

View

É responsável por utilizar as informações de que dispõe para produzir qualquer interface de apresentação que a aplicação precisar. Por exemplo, ela poderia usar um conjunto de dados retornados da camada *Model* para renderizar uma página HTML contendo esses dados.

[4] *Rapid Application Development* (RAD) é uma metodologia de desenvolvimento de *software* que envolve desenvolvimento iterativo e prototipação (McConnell 1996).

Controller

Essa camada lida com as requisições dos usuários. É responsável por renderizar de volta uma resposta com a ajuda das camadas *Model* e *View*.

4.3.2 Ciclo de Requisição do CakePHP

O tratamento típico de uma requisição do usuário no CakePHP é apresentado na Figura 4.5. Ele inicia com um usuário requisitando uma página ou recurso à aplicação. Essa requisição é processada primeiramente por um *dispatcher* (expedidor), que irá selecionar a classe *controller* correta para tratá-la. Uma vez que o pedido chega ao *controller* ele se comunicará com a camada *Model* para processar qualquer busca de dados ou operação de salvamento necessária. Depois que essa comunicação termina, o *controller* irá proceder delegando à classe *view* a tarefa de gerar uma saída resultante dos dados fornecidos pela camada *Model*. Finalmente, quando essa saída é gerada, é imediatamente renderizada para o usuário.

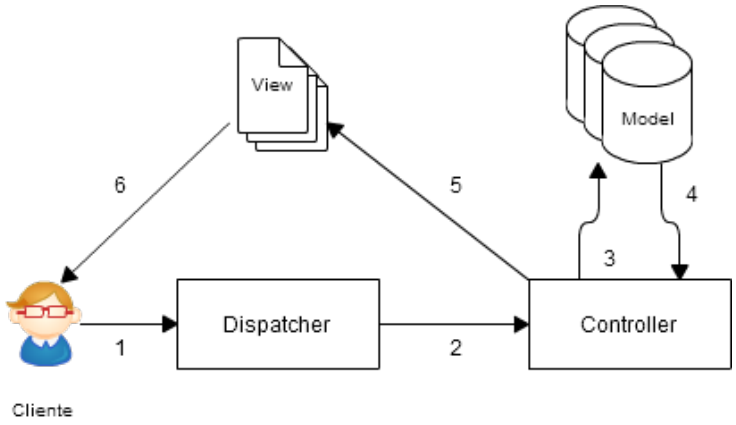

Figura 4.5: Ciclo de requisição do CakePHP.

4.4 Arquitetura do Kuase de Graça

A arquitetura do Kuase de Graça pode ser vista na Figura 4.6. Para cada tabela do modelo lógico (Seção 4.1.2), criou-se uma classe *model* para interagir com os dados armazenados na tabela.

Os casos de uso apresentados na Seção 3.18 são as requisições que podem ser feitas pelo usuário através do navegador. Essas requisições, como explicado na Seção 4.3.2, são tratadas por uma classe *controller* específica. Foram criadas várias classes *controller* para tratar essas requisições, distribuindo-as de acordo com a entidade principal que elas referenciam. Por exemplo, a requisição "Visualizar Depoimentos" diz respeito à entidade "Depoimento". Portanto, a classe *controller* designada para tratar essa requisição é "*Controller* Depoimentos".

Para melhor exemplificar, todas as requisições tratadas por "*Controller* Usuários" e "*Controller* Leilões" são apresentadas na Figura 4.7. Para cada requisição da esquerda, um método é definido na classe *controller* da direita. Esse método é chamado automaticamente pelo CakePHP sempre que a requisição é realizada.

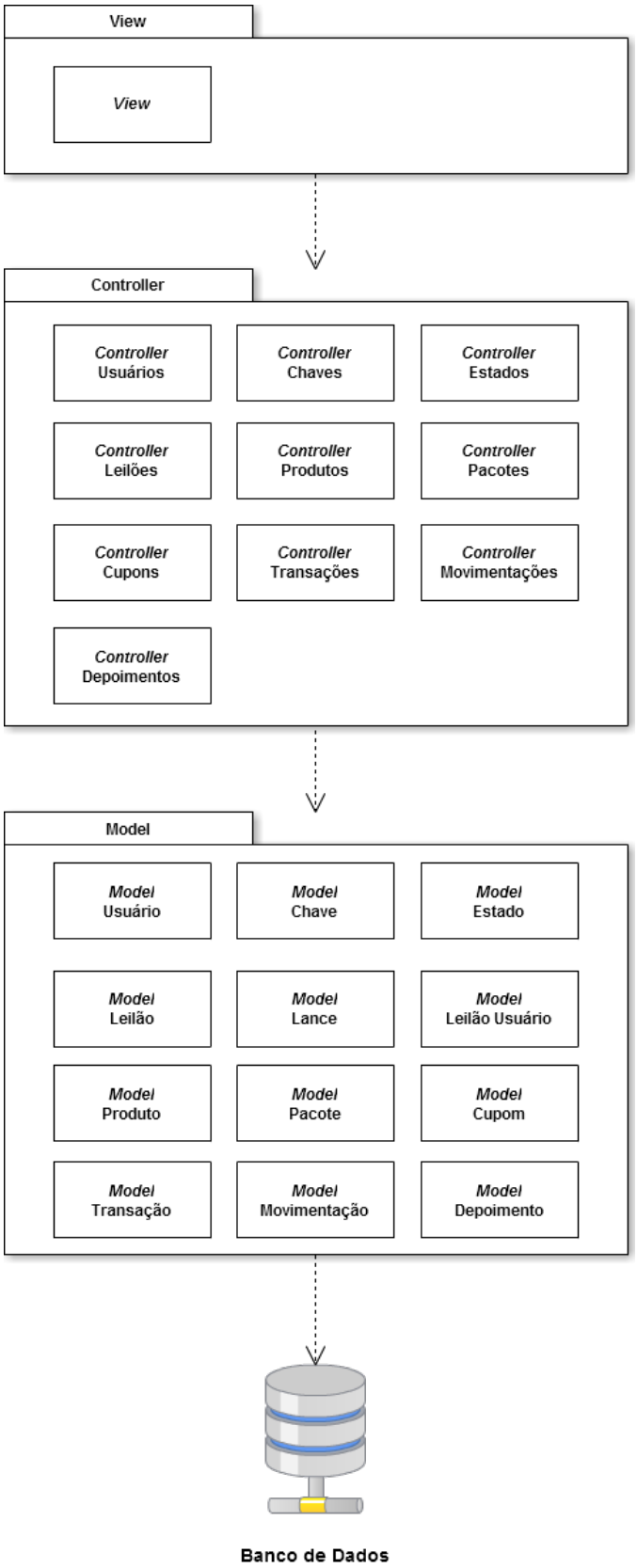

Figura 4.6: Arquitetura do Kuase de Graça.

Figura 4.7: Requisições tratadas por "*Controller* Usuários" e "*Controller* Leilões".

5

Otimização do Processamento de Requisições

5.1 Balanceamento de Carga

Balanceamento de carga é a habilidade de fazer vários servidores participarem no mesmo serviço e fazerem o mesmo trabalho. De acordo com (Schlossnagle 2004), existem dois principais motivos que levam um *site* além um único servidor Web:

- **Redundância:** Se um *site* tem um propósito crítico e não pode permitir nem mesmo uma interrupção breve de serviço, ele precisa usar múltiplos servidores Web para redundância. Por mais caro que seja o *hardware* de um servidor, ele irá eventualmente falhar, precisar ser trocado, ou precisar de manutenção física;
- **Capacidade:** Por outro lado, os *sites* são frequentemente movidos para uma configuração de múltiplos servidores para atender suas crescentes demandas de tráfego.

No caso do Kuase de Graça, se o serviço for interrompido, mesmo que rapidamente, durante a execução de um leilão, o sistema ficará incapaz de receber lances dos participantes e o produto provavelmente será arrematado incorretamente. Além disso, uma arquitetura escalável permitirá a adição de novos servidores conforme o número de usuários participantes dos leilões cresça.

Uma configuração de exemplo, com três servidores Web com carga balanceada, pode ser vista na Figura 5.1. Quando uma requisição chega da Internet, ela é direcionada para um dos três servidores, de acordo com um determinado algoritmo. O algoritmo *Round Robin*, por exemplo, usa um servidor por vez, como uma fila circular.

É importante observar que alguns desses algoritmos são não-determinísticos, o que significa que duas requisições consecutivas de um mesmo usuário têm uma grande chance

de atingirem dois servidores diferentes. Se as informações de sessão dos usuários são armazenadas nos servidores da aplicação, elas serão perdidas entre essas duas requisições. Contudo, no caso do Kuase de Graça, o CakePHP fornece um mecanismo embutido que permite armazenar as informações de sessão numa tabela do banco de dados, evitando esse problema.

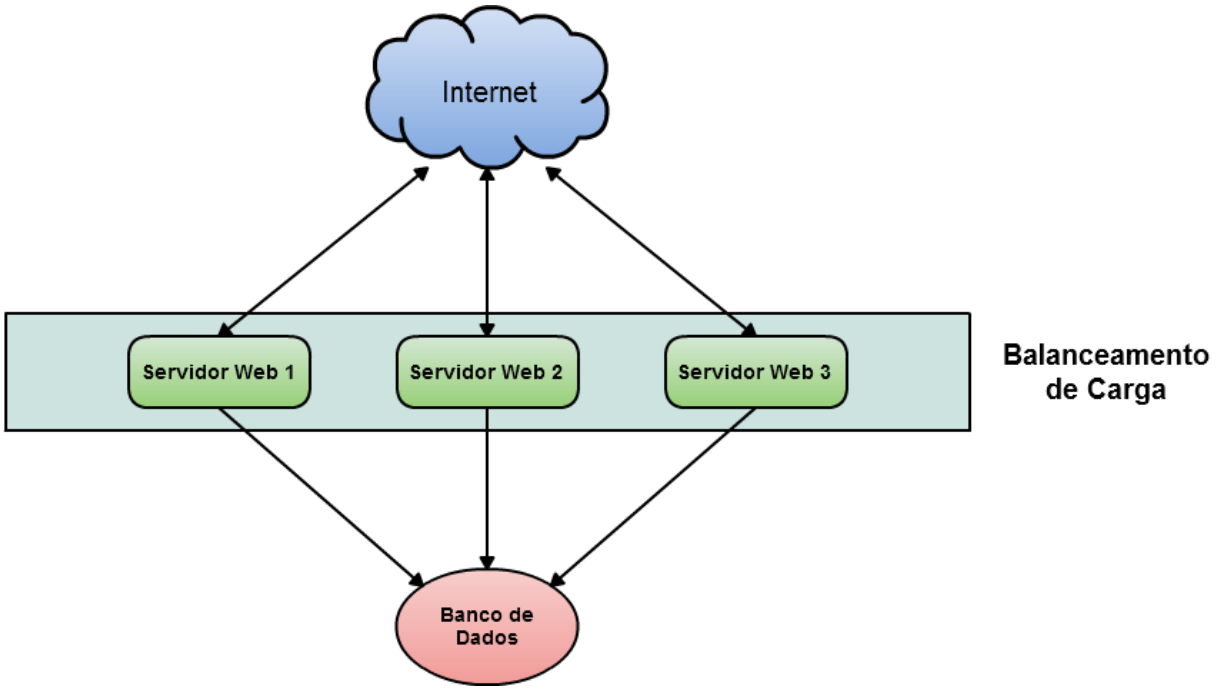

Figura 5.1: Balanceamento de carga em três servidores Web.

5.2 *View Caching*

Algumas páginas do *site* exibem o mesmo conteúdo toda vez que são acessadas. Por exemplo, a página "Fale Conosco" sempre exibirá um formulário de envio de mensagem. Em outras páginas, o conteúdo muda após o cadastro ou edição de algum registro. Por exemplo, a página de depoimentos vai sempre exibir os mesmos depoimentos até que um novo seja cadastrado. Apesar disso, sempre que uma requisição é recebida, todo o ciclo de requisição do CakePHP (Seção 4.3.2) é executado, consumindo recursos do servidor.

View Caching é um mecanismo embutido no CakePHP que, quando ativado e configurado, consegue responder a uma requisição reutilizando a resposta de uma requisição anterior compatível, evitando a execução de todo ciclo da Figura 4.5. A compatibilidade das requisições é verificada de acordo com a URL requisitada, conforme a Figura 5.2. Se a resposta de uma requisição compatível não é encontrada em *cache*, o ciclo de requisição do CakePHP é executado por completo, e então, a resposta gerada é armazenada, conforme a Figura 5.3.

O CakePHP também cuida de limpar o *cache* quando acontece alguma mudança nos dados e as páginas precisam ser geradas novamente. Por exemplo, quando um novo depoimento é cadastrado no sistema, todas as respostas em *cache* relacionadas à entidade "Depoimento" são removidas, inclusive `/depoimentos`. Assim, quando o próximo cliente fizer uma requisição `/depoimentos`, a resposta será gerada com o novo depoimento cadastrado e armazenada em *cache* até a próxima alteração.

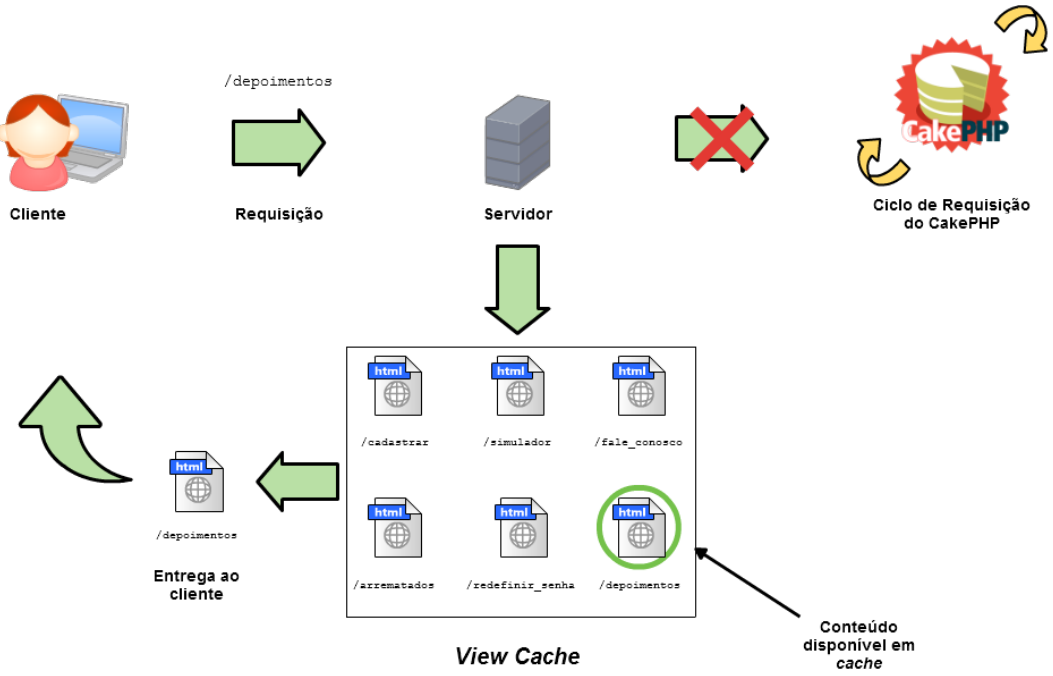

Figura 5.2: Mecanismo de *View Caching*.

5.2.1 Teste de Carga

Podem ser vistos na Figura 5.4, os resultados de um teste de carga realizado para verificar a eficiência desse mecanismo. Foram feitas cem requisições concorrentes da página principal do Kuase de Graça com o *View Caching* ativado. E mais cem requisições concorrentes com o *View Caching* desativado. O tempo levado para responder todas as requisições no segundo caso foi mais que o dobro que no primeiro caso, o que faz esse mecanismo útil para que o *site* forneça um bom tempo de resposta para os usuários num momento em que o mesmo esteja recebendo muitas visitas.

5.3 *View Caching* Distribuído

Originalmente, o mecanismo de *View Caching* armazena as páginas num diretório do servidor que está tratando a requisição, sem qualquer relação com os outros servidores que

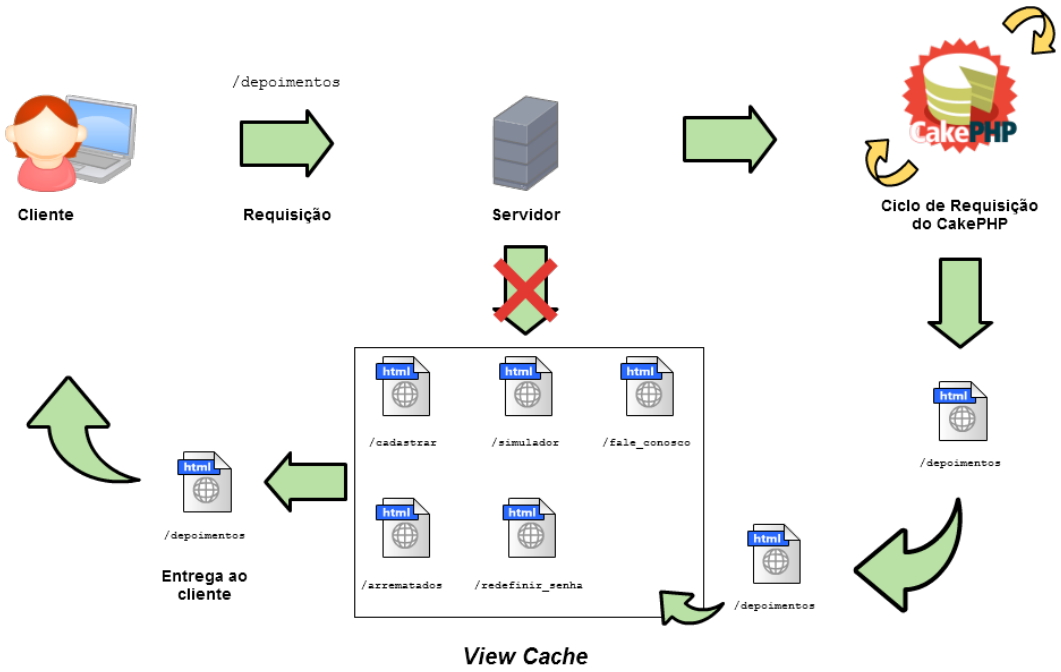

Figura 5.3: Armazenamento da resposta de uma requisição no *View Cache*.

possam estar participando do balanceamento de carga. Ou seja, esse diretório não é compartilhado entre eles, e cada servidor tem seu próprio diretório. Portanto, se, por exemplo, duas requisições da página de depoimentos são feitas, e essa página não estiver em *cache* em nenhum servidor, e cada uma for tratada por um servidor diferente, então as duas passarão, desnecessariamente, por todo o ciclo de requisição do CakePHP, como apresentado na Figura 5.5.

Pior acontece quando a limpeza do *View Cache* é realizada somente em um dos servidores. Por exemplo, suponhamos que dois servidores estejam com a página de depoimentos em *cache*. Se um novo depoimento é cadastrado, as páginas relacionadas à entidade "Depoimento" que estão armazenadas no *View Cache* devem ser excluídas, de maneira a serem geradas com os novos dados, como explicado na Seção 5.2. Contudo, como são dois servidores, somente as páginas do *View Cache* do servidor que tratou a requisição de cadastro serão excluídas. Se, nesse momento, o outro servidor receber uma requisição da página de depoimentos, ele irá retornar a versão que está em seu *View Cache*, desatualizada.

Para resolver esses problemas, deve existir um *View Cache* único que todos os servidores usarão. Neste trabalho, será utilizado um sistema distribuído de *cache* de objetos na memória: Memcached[1]. Os objetos, nesse caso, serão as páginas do *View Cache*. O Memcached é mais um sistema de armazenamento de chave-valor na memória, contudo, ele consegue utilizar a memória de todos os servidores como se fosse uma só, como pode ser visto na Figura 5.6. Se, ao invés de 2, houvessem cinquenta servidores Web, ainda haveria 64MB de tamanho útil no primeiro quadro da figura, e aproximadamente 3.2GB no segundo.

[1] http://memcached.org/

No CakePHP, o *View Cache* só consegue ser armazenado em arquivos, sendo necessário alterar algumas linha de código do *framework*, que dizem respeito à escrita e leitura dos mesmos, para ter essa funcionalidade. Ao final, obtém-se um sistema que pode ser facilmente incrementado com novos servidores, de acordo com a necessidade e com o uso de *View Cache*.

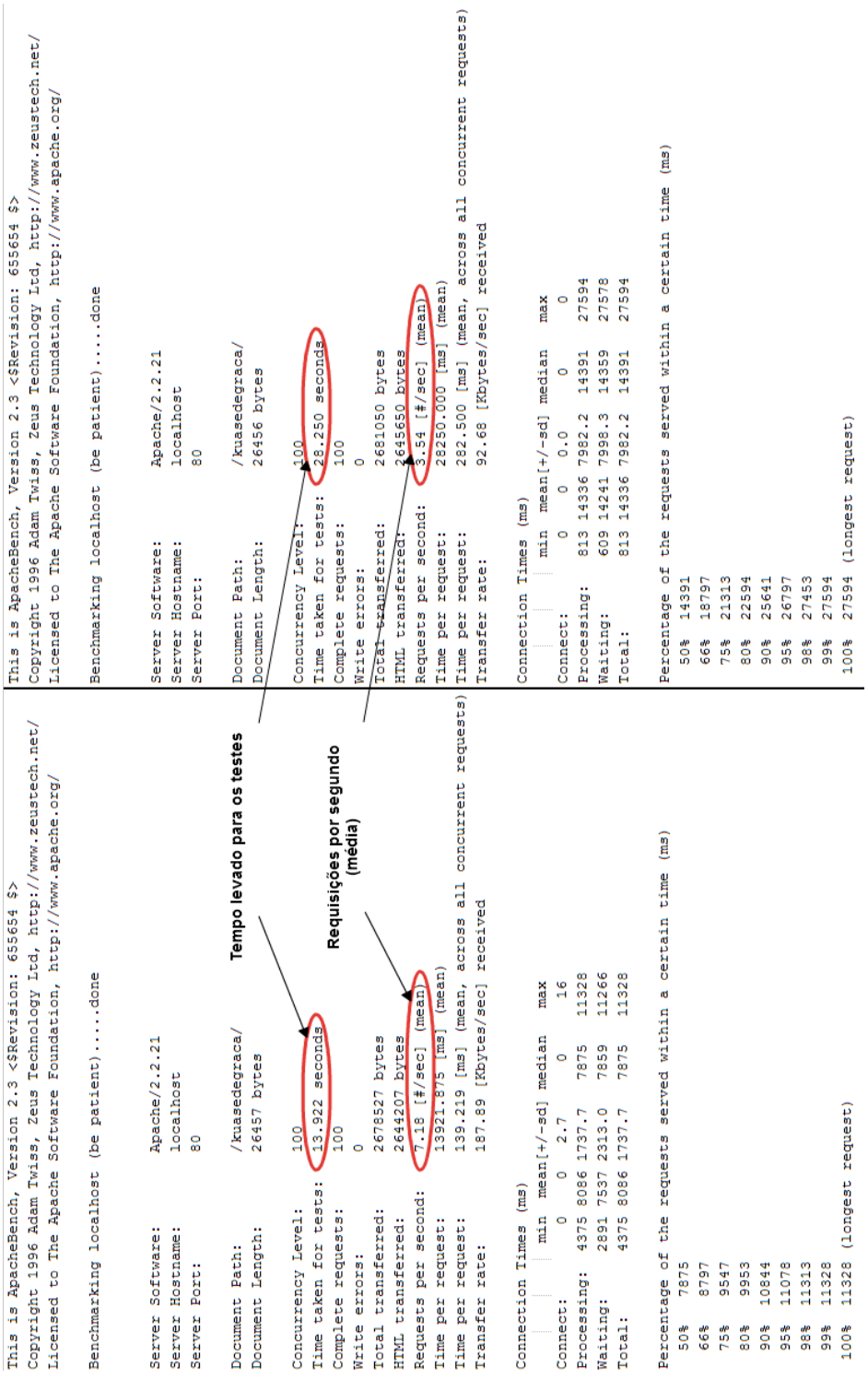

Figura 5.4: Resultados do teste de carga realizado para verificar a eficiência do mecanismo de *View Caching*.

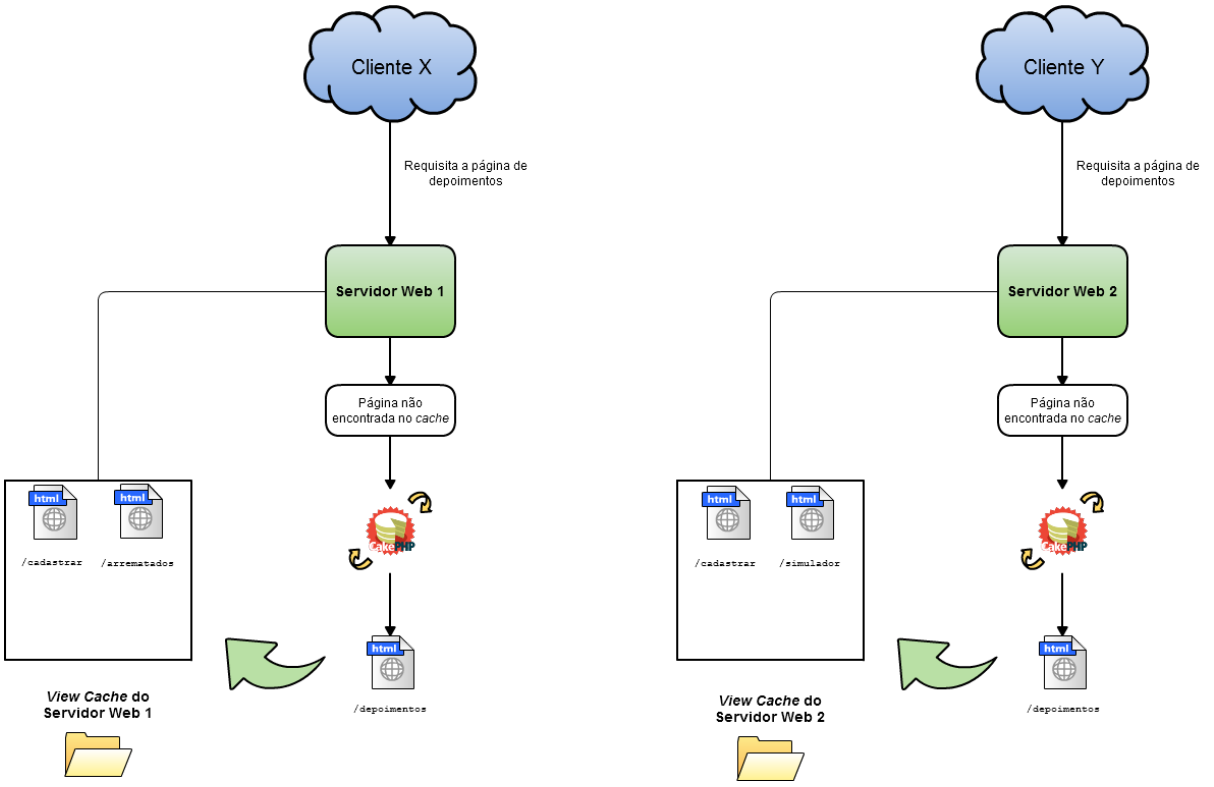

Figura 5.5: Mecanismo de *View Caching* não compartilhado entre os servidores.

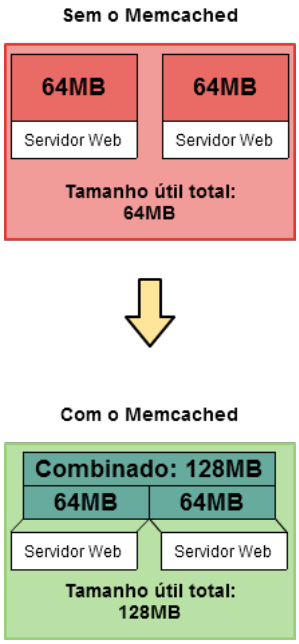

Figura 5.6: Utilização do Memcached.

6

Capturas de Tela

Neste capítulo são apresentadas capturas de tela das principais páginas do *site* Kuase de Graça, quando o mesmo estava prestes a completar um mês *online*.

- A Figura 6.1 é uma captura de tela da página inicial;
- A Figura 6.2 é uma captura de tela da página de detalhes de um leilão;
- A Figura 6.3 é uma captura de tela da página de cadastro;
- A Figura 6.4 é uma captura de tela da página de depoimentos;
- A Figura 6.5 é uma captura de tela da página de simulação de leilão virtual;
- A Figura 6.6 é uma captura de tela da página de leilões arrematados.

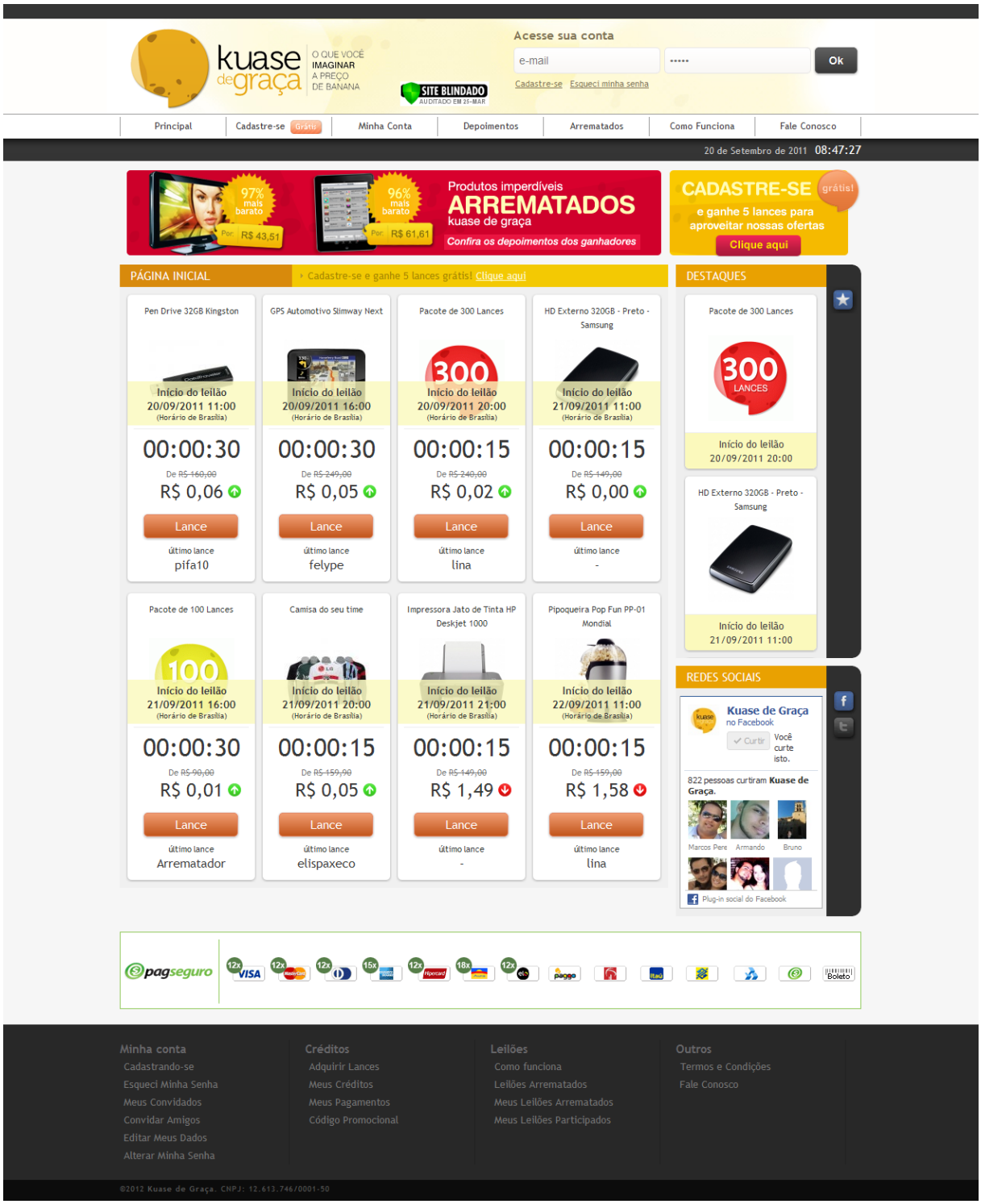

Figura 6.1: Captura de tela da página inicial.

Figura 6.2: Captura de tela da página de detalhes de um leilão.

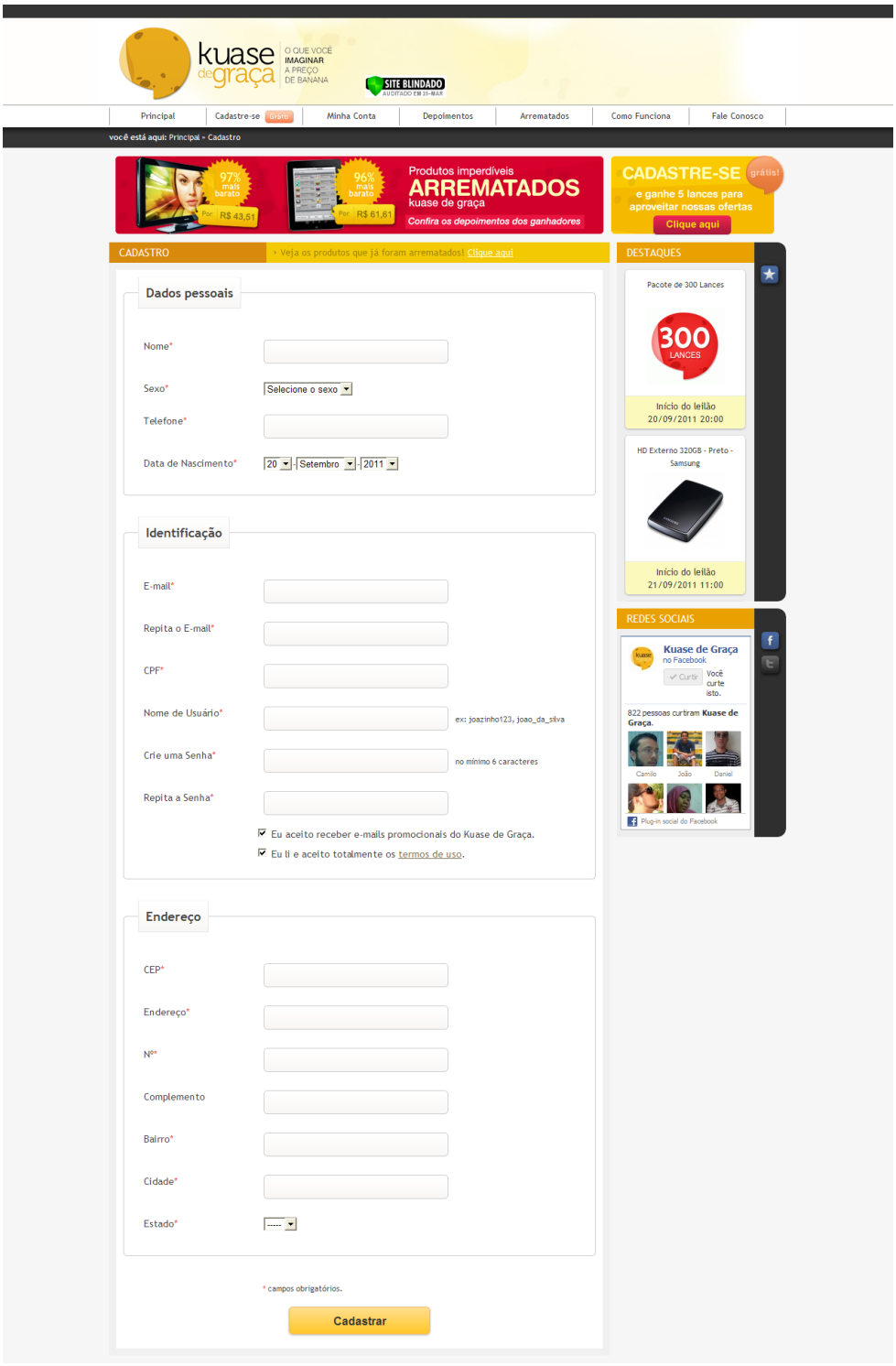

Figura 6.3: Captura de tela da página de cadastro.

Figura 6.4: Captura de tela da página de depoimentos.

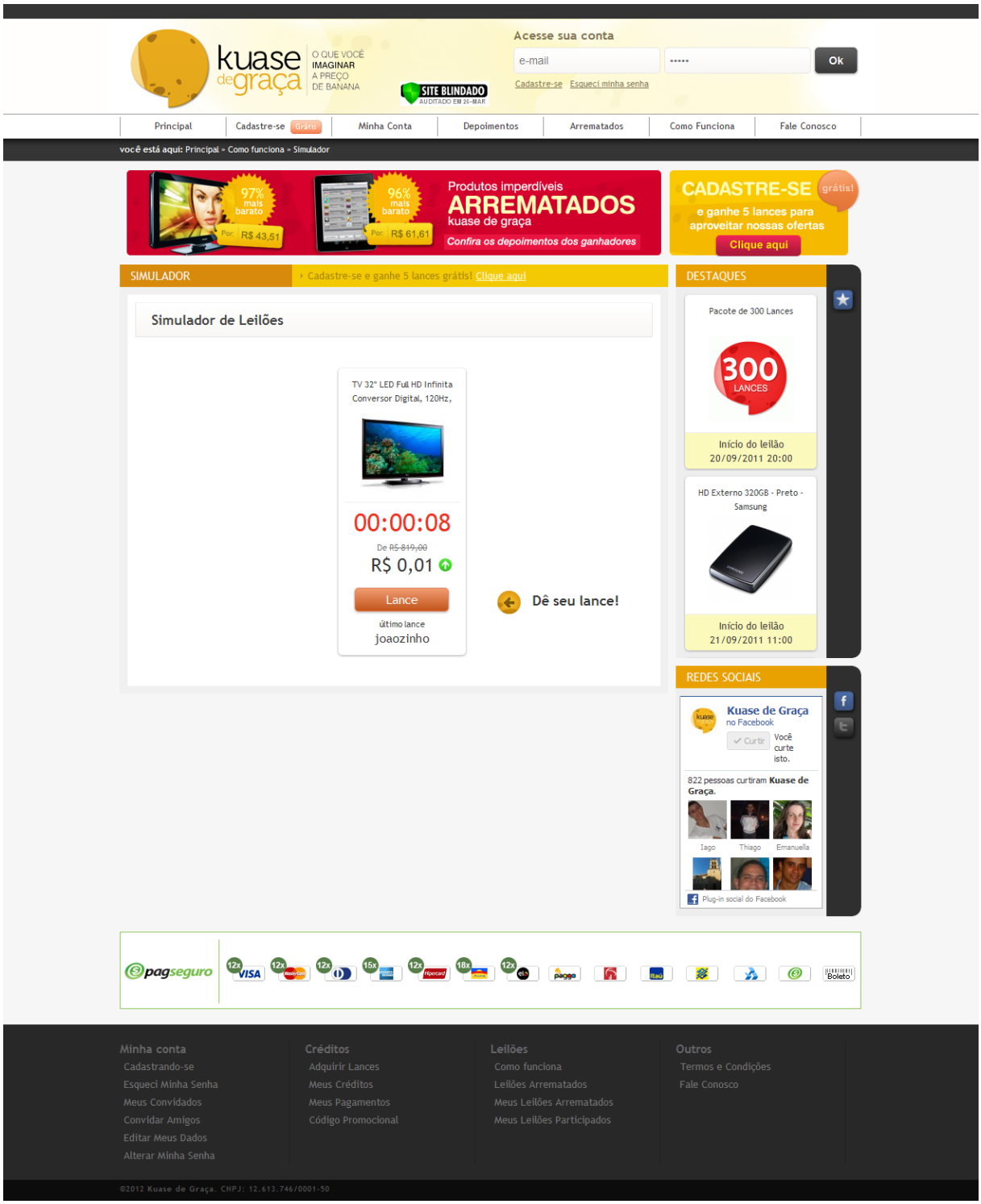

Figura 6.5: Captura de tela da página do simulador.

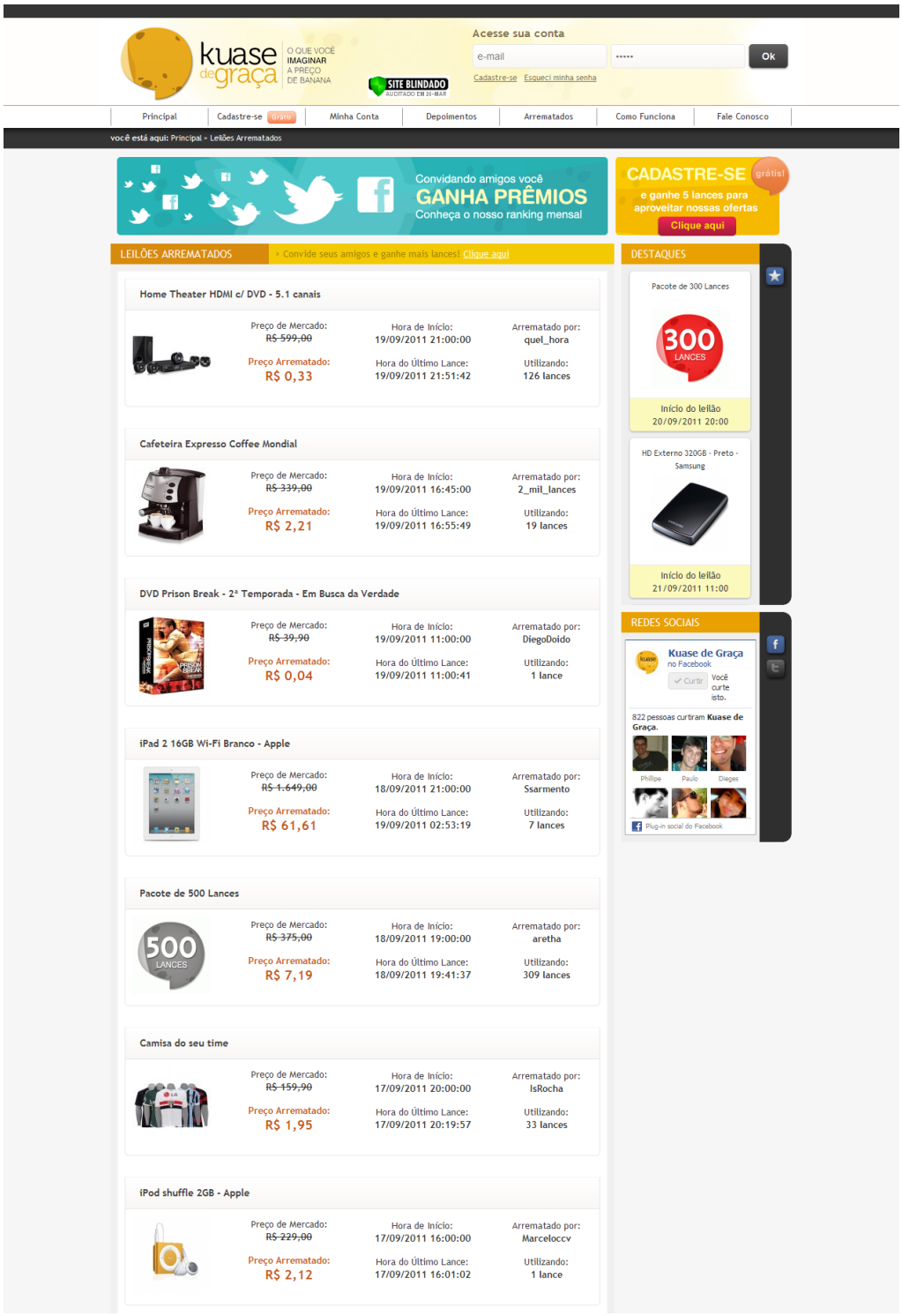

Figura 6.6: Captura de tela da página de leilões arrematados.

7

Conclusão

Os leilões tradicionais deram origem aos leilões virtuais, que se tornaram populares nos últimos anos, vendendo a ideia de adquirir produtos por um preço significativamente menor que o valor de mercado. Os *sites* que os mantêm devem oferecer um bom tempo de resposta para seus usuários, tendo em vista que esse tipo de aplicação é bastante dinâmico.

Apresentou-se neste trabalho as etapas de desenvolvimento de um *site* de leilões virtuais chamado Kuase de Graça. Esse *site* possui dois tipos de leilão: progressivo e regressivo. O leilão progressivo já é usado em outros *sites*. Já o leilão regressivo é uma inovação do Kuase de Graça por dar aos usuários a oportunidade de arrematar produtos e ainda ganhar lances para usar em outros leilões.

A partir da especificação das funcionalidades desse sistema, foi criado um modelo conceitual dos dados utilizando o diagrama entidade relacionamento, que deu origem a um modelo lógico de um banco de dados relacional. Utilizou-se o *framework* de desenvolvimento rápido CakePHP, que possui arquitetura MVC, separando a aplicação em três camadas principais: *Model*, *View*, e *Controller*. Para cada tabela do banco de dados, uma classe *model* foi criada para interagir com a mesma. Os casos de uso foram distribuídos entre classes *controller*, de acordo com a entidade que referenciavam.

Foi apresentado o mecanismo de *View Caching*, que filtra as requisições que devem passar pelo ciclo de requisição do CakePHP e, de acordo com o teste de carga realizado, é útil no propósito de diminuir o tempo de resposta aos usuários do *site*. Esse mecanismo foi adaptado para funcionar corretamente num ambiente distribuído.

Por fim, foram apresentadas capturas de tela do *site* em funcionamento.

Referências bibliográficas

Burbeck, S. (1997), 'Application programming in smalltalk-80: How to use model-view-controller (mvc).', *University of Illinois in Urbana-Champaign (UIUC) Smalltalk Archive* .
URL: *http://st-www.cs.illinois.edu/users/ smarch/st-docs/mvc.html*

Chen, P. P. S. (1975), The entity-relationship model: toward a unified view of data, *in* 'Proceedings of the 1st International Conference on Very Large Data Bases', VLDB '75, ACM, New York, NY, USA, pp. 173–173.
URL: *http://doi.acm.org/10.1145/1282480.1282492*

Goodman, D. & Morrison, M. (2007), *JavaScript Bible*, Wiley Publishing, Inc.

Heuser, C. A. (2008), *Projeto de Banco de Dados*, number 4 *in* 'Livros Didaticos - Instituto de Informatica da UFRGS', 4 edn, Editora Sagra Luzzatto.

McAfee, R. P. & McMillan, J. (1987), 'Auctions and bidding', *Journal of Economic Literature* **25**(2), 699–738.
URL: *http://www.jstor.org/stable/2726107*

McConnell, S. (1996), *Rapid Development: Taming Wild Software Schedules*, Microsoft Press Books.

Schlossnagle, G. (2004), *Advanced PHP Programming*, Sams Publishing.

von Ahn, L., Blum, M. & Langford, J. (2004), 'Telling humans and computers apart automatically', *Commun. ACM* **47**(2), 56–60.
URL: *http://doi.acm.org/10.1145/966389.966390*

Welling, L. & Thomson, L. (2005), *PHP and MySQL Web Development*, Sams Publishing.

Este trabalho foi redigido em LaTeX utilizando uma modificação do estilo IC-UFAL. As referências bibliográficas foram preparadas no JabRef e administradas pelo BibTeX com o estilo LaCCAN. O texto utiliza fonte Fourier-GUT*enberg* e os elementos matemáticos a família tipográfica Euler Virtual Math, ambas em corpo de 12 pontos. A numeração dos capítulos segue com a família tipográfica Art Nouveau Caps.