

**UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CAMPUS PATO BRANCO
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE
SISTEMAS**

**JAISON SCHEITT STEFANELLO
RAFAEL ALBANI**

SISTEMA PARA GERENCIAMENTO DE VINÍCOLAS

**PATO BRANCO
2011**

**JAISON SCHEITT STEFANELLO
RAFAEL ALBANI**

SISTEMA PARA GERENCIAMENTO DE VINÍCOLAS

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Diplomação, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná, Campus Pato Branco, como requisito parcial para obtenção do título de Tecnólogo.

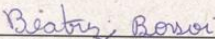
Orientadora: Profa. Beatriz T. Borsoi

**PATO BRANCO
2011**

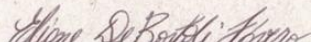
ATA Nº: 187

DEFESA PÚBLICA DO TRABALHO DE DIPLOMAÇÃO DOS ALUNOS JAISON SCHEITT STEFANELLO e RAFAEL ALBANI.

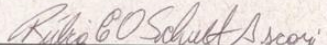
Às 11:10 hrs do dia 7 de julho de 2011, Bloco S da UTFPR, Campus Pato Branco, reuniu-se a banca avaliadora composta pelos professores Beatriz Terezinha Borsoi (Orientadora), Eliane Maria de Bortoli Fávero (Convidada) e Rúbia E. de Oliveira Schultz Ascari (Convidada), para avaliar o Trabalho de Diplomação do aluno Jaison Scheitt Stefanello, matrícula 949825 e do aluno Rafael Albani, matrícula 949876, sob o título **Sistema para Gerenciamento de Vinícolas**; como requisito final para a conclusão da disciplina Trabalho de Diplomação do Curso Superior de Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Coordenação de Informática. Após a apresentação os candidatos foram entrevistados pela banca examinadora, e a palavra foi aberta ao público. Em seguida, a banca reuniu-se para deliberar considerando o trabalho **APROVADO**. Às 12:10 hrs foi encerrada a sessão.



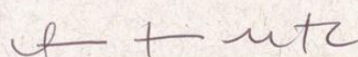
Prof. Beatriz Terezinha Borsoi, Dr.
Orientadora



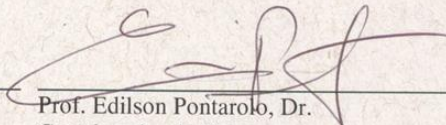
Prof. Eliane Maria de Bortoli Fávero, M.Sc.
Convidada



Prof. Rúbia E. de Oliveira Schultz Ascari, M.Sc.
Convidada



Prof. Omero Francisco Bertol, M.Sc.
Coordenador do Trabalho de Diplomação



Prof. Edilson Pontarolo, Dr.
Coordenador do Curso

RESUMO

STEFANELLO, Jaison Scheitt; ALBANI, Rafael. Sistema para gerenciamento de vinícolas. 2011. 58 f. Monografia de Trabalho de Conclusão de Curso. Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas. Universidade Tecnológica Federal do Paraná. Pato Branco, 2011.

O uso dos serviços Internet, *web* por exemplo, como meio de execução de aplicativos computacionais apresenta diversas vantagens, incluindo a não necessidade de instalação do referido *software* no cliente e o acesso de qualquer computador com conexão à Internet. Essas condições podem favorecer empresas e indústrias de pequeno porte no uso de sistemas que possam auxiliá-las a obter vantagens competitivas. Assim, esse trabalho propõe o desenvolvimento de um sistema para gerenciamento de vinícolas, provendo acompanhamento das principais atividades realizadas, incluindo do fornecimento da matéria prima até a venda. O controle é gerencial, portanto, não são atendidas funcionalidades relacionadas ao chão de fábrica. É fornecida uma ferramenta de auxílio ao gestor de uma vinícola voltado para o controle das atividades realizadas. O sistema foi desenvolvido como planejado e as tecnologias utilizadas, destacando-se a linguagem PHP e o *framework* Zend que implementa o *Model-View-Controller (MVC)*, foram adequadas para a implementação das funcionalidades definidas. Assim, esse trabalho é, também, um exemplo de uso dessas tecnologias.

Palavras-chave: Zend Framework. Linguagem PHP. *Model-View-Controller*. Gerenciamento de vinícolas.

LISTA DE FIGURAS

Figura 1 – Esquema básico do modelo MVC.....	16
Figura 2 – Principais componentes do Zend	19
Figura 3 – Diagrama de classes (parte 1)	25
Figura 4 – Diagrama de classes (parte 2)	26
Figura 5 – Diagrama de classes (parte 3)	27
Figura 6 – Diagrama entidade relacionamento (parte 1)	28
Figura 7 – Diagrama entidade relacionamento (parte 2)	29
Figura 8 – Diagrama entidade relacionamento (parte 3)	30
Figura 9 – Tela de login	31
Figura 10 – Tela inicial do sistema	32
Figura 11 – Cadastro de uma nova vinícola	33
Figura 12 – Cadastro de uma vinícola (2)	34
Figura 13 – Edição de uma vinícola	34
Figura 14 – Exclusão de uma vinícola.....	35
Figura 15 – Cadastro de clientes.....	35
Figura 16 – Menu de cadastros.....	36
Figura 17 – Cadastro de usuário	37
Figura 18 – Menu de estoque	38
Figura 19 – Edição de estoque de ingrediente	38
Figura 20 – Edição de estoque de produto	39
Figura 21 – Gerenciamento de ordens de produção	39
Figura 22 – Cadastro de ordem de produção	40
Figura 23 – Edição de ordem de produção	41
Figura 24 – Cadastro de ingredientes para ordem de produção	41
Figura 25 – Menu movimentação	42
Figura 26 – Listagem de venda de produtos.....	43
Figura 27 – Cadastro de venda de produtos	43
Figura 28 – Edição de venda de produtos	44
Figura 29 – Adição de produto a uma venda	44

LISTAGENS DE CÓDIGO

Listagem 1 – Classe Application_Model_Auth.....	45
Listagem 2 – Estrutura do menu do sistema.....	46
Listagem 3 – application.ini	48
Listagem 4 – index.phtml	49
Listagem 5 – VinicolaController.php.....	50
Listagem 6 – Classe que representa a tabela vinícola	52
Listagem 7 – Classe Vinicola	53
Listagem 8 – Classe VinicolaMapper	54

LISTA DE ABREVIATURAS E SIGLAS

ACID	Atomicidade Consistência Isolamento Durabilidade
CNPJ	Cadastro Nacional de Pessoas Jurídicas
CSS	<i>Cascading Style Sheets</i>
GUI	<i>Graphical User Interface</i>
HTML	<i>HyperText Markup Language</i>
HTTP	<i>Hypertext Transfer Protocol</i>
IDE	<i>Integrated Development Environment</i>
IMAP	<i>Internet Message Application Protocol</i>
MVC	<i>Model-View-Controller</i>
PHP	<i>PHP Hypertext Preprocessor</i>
POP3	<i>Post Office Protocol</i>
SMTP	<i>Simple Mail Transfer Protocol</i>
SQL	<i>Structured Query Language</i>
URL	<i>Uniform Resource Locator</i>

SUMÁRIO

1 INTRODUÇÃO.....	8
1.1 CONSIDERAÇÕES INICIAIS	8
1.2 OBJETIVOS.....	9
1.2.1 Objetivo Geral	9
1.2.2 Objetivos Específicos	9
1.3 JUSTIFICATIVA	9
1.4 ORGANIZAÇÃO DO TEXTO	10
2 O MVC NO DESENVOLVIMENTO DE APLICAÇÕES WEB.....	11
2.1 CONTEXTO.....	11
2.2 ARQUITETURA DE SOFTWARE.....	12
2.2.1 Padrões e Estilos Arquiteturais	13
3 MATERIAIS E MÉTODO	17
3.1 MATERIAIS	17
3.1.1 PHP.....	17
3.1.2 EclipsePHP Helios.....	18
3.1.3 Zend Framework	18
3.1.4 MySQL	20
3.1.5 Apache HTTP Server.....	21
3.2 MÉTODO	22
4 SISTEMA DESENVOLVIDO.....	23
4.1 APRESENTAÇÃO DO SISTEMA	23
4.2 MODELAGEM DO SISTEMA	23
4.3 DESCRIÇÃO DO SISTEMA.....	31
4.4 IMPLEMENTAÇÃO DO SISTEMA	45
5 CONCLUSÃO.....	56
REFERÊNCIAS	57

1 INTRODUÇÃO

Este capítulo apresenta as considerações iniciais com uma visão geral do trabalho, os objetivos, a justificativa e a organização do texto.

1.1 CONSIDERAÇÕES INICIAIS

Uma vinícola tem como finalidade principal a fabricação de produtos derivados da uva. Em geral, o produto principal produzido é o vinho, mas também estão incluídos outros derivados como, por exemplo, suco, espumante e vinagre. A produção desses produtos, especialmente o vinho, pode ser realizada por uma fábrica de pequeno porte e o processo pode ser bastante artesanal, ou por uma indústria de grande porte com um chão de fábrica automatizado. Contudo, mesmo uma fábrica com produção artesanal e de baixa escala pode contar com o auxílio de um sistema de informação na realização das suas atividades.

Para facilitar o acesso das empresas de pequeno porte a sistemas informatizados para controle e gerenciamento de suas atividades é necessário que o sistema tenha requisitos de uso simples. Esses requisitos se referem tanto ao conhecimento necessário para usar o sistema, como os recursos de *hardware* e de *software* para a execução do sistema.

A Internet pode ser vista como uma possibilidade de facilitar o uso de sistemas de controle e de gestão por empresas de pequeno porte. Isso porque o aplicativo não precisa ser instalado na máquina cliente, é necessário apenas ter um computador com acesso à Internet e um navegador *web* instalado. Porém, ressalta-se que dependendo das tecnologias utilizadas pode ser necessário instalar *plugins*, por exemplo. Como ocorre com o *flash player* que é necessário para executar aplicações Internet ricas desenvolvidas em Flex.

Uma aplicação *web* é uma alternativa para uma empresa ou indústria de pequeno porte. Verificou-se, então, a oportunidade de implementar um sistema para *web* para gerenciamento de vinícolas. A tecnologia utilizada é PHP (PHP *Hypertext Preprocessor*), com o Zend Framework que implementa o padrão MVC (*Model-View-Controller*). O sistema desenvolvido está centrado no controle gerencial desses estabelecimentos.

1.2 OBJETIVOS

O objetivo geral apresenta o resultado principal da realização deste trabalho e os objetivos específicos complementam esse objetivo geral.

1.2.1 Objetivo Geral

Implementar um sistema *web* para gerenciamento de vinícolas.

1.2.2 Objetivos Específicos

- Prover uma maneira de auxiliar vinicultores no gerenciamento do processo de produção de uma vinícola.
- Auxiliar no controle de estoque e de matéria-prima envolvidos na produção e na venda dos produtos de uma vinícola.
- Exemplificar o uso do *framework* Zend na implementação do padrão arquitetural MVC.

1.3 JUSTIFICATIVA

A escolha por implementar um aplicativo de controle de vinícolas ocorreu pela oportunidade de desenvolver uma solução computacional para indústrias de produção de vinho de pequeno porte. Para que, assim, elas pudessem contar com um sistema para controle de sua produção.

Existem vários processos de uma vinícola que podem ser automatizados. Dentre esses estão: o controle das uvas utilizadas na produção de vinhos, sucos, espumantes e outros; o gerenciamento de estocagem desses produtos; o controle do tempo para que os vinhos estejam prontos para venda; o gerenciamento de estoque; o controle de compras e de fornecedores; e a venda propriamente dita.

Esses processos integrados em um sistema que auxilie no seu gerenciamento e controle podem trazer diversos benefícios ao gerente de uma vinícola. Isso porque o desvincularia de atividades rotineiras e ele poderia dedicar o tempo que utilizar para as tarefas

que podem ser automatizadas no melhoramento dos processos, no atendimento a clientes, na busca de novos clientes e demais atividades gerenciais e estratégicas.

A escolha pelo desenvolvimento de um sistema na plataforma *web* foi feita considerando os seguintes quesitos:

- a) Acesso de qualquer dispositivo que tenha um sistema navegador *web* instalado e possua acesso a internet;
- b) Não existe a necessidade de o aplicativo ser instalado em todas as máquinas que irão utilizá-lo;
- c) Manutenção facilitada do sistema, já que não são necessárias atualizações em todas as máquinas cliente;
- d) Multiplataforma, pela independência de computador e sistema operacional.

Para a implementação do sistema foi utilizado o *framework* Zend com a linguagem PHP. A escolha desse *framework* é justificada pela possibilidade de aplicação do padrão arquitetural MVC. E, assim, é uma oportunidade de exemplificar o uso desse padrão.

1.4 ORGANIZAÇÃO DO TEXTO

Este texto está organizado em capítulos, dos quais este é o primeiro e apresenta a ideia do sistema, incluindo os objetivos e a justificativa.

No Capítulo 2 está o referencial teórico sobre arquitetura de *software* e o modelo de arquitetura MVC. No Capítulo 3 está o método, que é a sequência geral de passos definidos para a realização do trabalho e os materiais utilizados. Os materiais se referem ao que foi utilizado para modelar e implementar o sistema. O Capítulo 4 apresenta a modelagem e o resultado da implementação. No Capítulo 5 está a conclusão com as considerações finais. Por fim estão as referências bibliográficas utilizadas para compor o texto dos Capítulos 3 e 4 e no entendimento dos conceitos relacionados a arquitetura de software e ao padrão arquitetural MVC.

2 O MVC NO DESENVOLVIMENTO DE APLICAÇÕES WEB

Este capítulo contém o referencial teórico do trabalho. Inicialmente é apresentado o contexto no qual padrões arquiteturais se enquadram. A seção seguinte se refere à arquitetura de *software*, com uma visão geral e conceitual do assunto em que padrões arquiteturais de projeto se inserem e em seguida é tratado especificamente sobre o padrão arquitetural MVC. Esse padrão é utilizado na implementação do sistema objeto deste trabalho.

2.1 CONTEXTO

Uma aplicação *web* é denominada, de maneira geral, como a que é executada utilizando um navegador *web* e é acessada por meio de recursos Internet. É nesse navegador que a interface de interação da aplicação é executada. Esse tipo de aplicação pode ser utilizado na rede Internet ou *intranet*. A ideia básica é de uma aplicação do tipo cliente/servidor em que há um servidor *web* e clientes que utilizam um navegador *web* para acessar a aplicação.

Em termos de serviços, os elementos são organizados da seguinte forma: de um lado está o cliente, navegador ou *browser web*, que solicita dados ao servidor, recebe as respostas, formata a informação e a apresenta ao usuário; do outro lado está o servidor que recebe as requisições, lê os dados armazenados em arquivos e os retorna para o cliente.

Em páginas *web* baseadas em HTML (*HyperText Markup Language*), o código desenvolvido com a linguagem de programação para implementar a lógica de negócio e a manipulação de dados pode ficar mesclado com a linguagem de marcação, entre *tags* HTML, que compõem a interface da aplicação. A junção das *tags* de marcação da linguagem de composição da interface, a página *web*, com o código que implementa a lógica de negócio dificulta a manutenção da aplicação.

Uma das maneiras de minimizar o problema do alto acoplamento entre a camada de interface, de lógica de negócio e de banco de dados é por meio da separação em camadas de implementação do sistema. O padrão de projeto (*design pattern*) MVC possibilita desvincular a manipulação de dados persistentes, da lógica de negócio e da lógica de visualização.

O padrão de projeto MVC é um padrão de projeto arquitetural. Esse tipo de padrão define os elementos, as relações e as regras a serem seguidas que já tiveram sua utilidade

avaliada em soluções de problemas passados (GERMOGLIO, 2010).

2.2 ARQUITETURA DE SOFTWARE

Em termos de histórico de conceito e referências sobre arquitetura de *software* destacam-se Perry e Wolf (1992) com um dos primeiros conceitos referenciados sobre arquitetura de *software* e que contém a essência de conceituações mais atuais; Garlan e Shaw (1994) como os autores mais referenciados nesse assunto; a IEEE 1471-2000 (2000) que normatiza definições e conceitos de arquitetura; e Bass, Clements e Kazman (2003) que é uma referência mais recente e bastante elucidativa desse assunto. O conceito de arquitetura relacionado a partes de sistema e a interação e organização dessas partes é utilizado por esses autores.

Para Perry e Wolf (1992), a arquitetura de *software* é um conjunto de elementos arquiteturais que possuem alguma organização. A definição desses autores é representada pela seguinte fórmula:

$$\text{Arquitetura} = \{\text{Elementos, Organização, Decisões}\}$$

Os elementos dessa fórmula representam o que está envolvido em uma representação de arquitetura. De acordo com essa fórmula, a arquitetura de software é um conjunto de elementos arquiteturais organizados de alguma maneira e essa organização está relacionada a decisões arquiteturais.

Para Germoglio (2010) os elementos e sua organização são definidos por meio de decisões que satisfazem objetivos e restrições do sistema. Os tipos de elementos arquiteturais são: elementos de processamento que usam ou transformam informação; elementos de dados que contêm a informação a ser usada e transformada; e elementos de conexão que ligam elementos entre si. A organização determina as relações entre os elementos. Essas relações possuem propriedades e restringem como os elementos devem interagir de forma a satisfazer os objetivos do sistema. Adicionalmente, essas relações devem ser ponderadas de modo a indicar sua importância no processo de decisão entre alternativas arquiteturais e de projeto.

Garlan e Shaw (1994) se referem à arquitetura de *software* como um nível de projeto que inclui questões que vão além dos algoritmos e das estruturas de dados da computação. É a projeção e a especificação da estrutura geral do sistema, incluindo a sua organização e a estrutura de controle global, protocolos de comunicação, sincronização e acesso a dados, atribuição de funcionalidade a elementos de projeto, distribuição física do sistema e a

possibilidade de seleção entre alternativas de projeto. E, também, decisões que impactarão no comportamento do sistema em termos de escala e de desempenho, dentre outros atributos de qualidade.

A IEEE 1471-2000 (2000) também tem como base para a definição de uma arquitetura elementos (no sentido de partes) e interação entre os mesmos, mas está centrada na forma de documentar a arquitetura, de organizar as suas representações em modelos de acordo com agrupamentos de interesses, que são as visões. Essa norma sugere um modelo conceitual para descrição arquitetural de sistemas intensivos em *software*. A IEEE 1471-2000 (2000) define arquitetura como a organização fundamental de um sistema incorporada em seus componentes, os relacionamentos desses componentes entre si e com o ambiente e os princípios que conduzem seu projeto e evolução.

Para Bass, Clements e Kazman (2003) a arquitetura de sistema computacional é a estrutura do sistema, que é composta de elementos de *software*, as propriedades externamente visíveis desses elementos e os relacionamentos entre eles.

Todas as definições apresentadas para arquitetura são, de certa forma, convergentes. Isso porque elas se referem à arquitetura como compreendendo a estrutura (elementos, componentes), as relações entre esses componentes e as decisões (ou princípios) que determinam como esses elementos estão relacionados.

De forma a resumir os conceitos apresentados e possibilitar as representações distintas da arquitetura, para este trabalho conceitua-se arquitetura de *software* como as diversas representações de um sistema. Essas representações são compostas por elementos relacionados entre si e cada representação visa atender interesses específicos.

2.2.1 Padrões e Estilos Arquiteturais

Um padrão pode ser considerado como uma experiência estruturada de projeto, que pode ser reusada para solucionar problemas recorrentes. Um padrão de projeto arquitetural define elementos, relações e regras a serem seguidas e que já tiveram sua utilidade avaliada na solução de problemas.

De acordo com Germoglio (2010) a principal diferença entre um padrão arquitetural, também denominado de estilo arquitetural, e um padrão de projeto, também denominado *design pattern*, é que o primeiro lida com problemas em nível arquitetural, tornando-se mais abrangente no *software*. Por outro lado, a aplicação de um padrão de projeto tem efeito mais

restrito na solução. É comum que os padrões de projeto estejam associados à implementação do sistema, ao uso de tecnologias. E os padrões arquiteturais estejam mais voltados para a solução do problema, em um escopo de mais alto nível. Contudo, é possível encontrar padrões inicialmente descritos como arquiteturais tendo efeito apenas local no projeto e vice-versa.

De acordo com McConnell (2004), os benefícios do uso de padrões em um projeto:

- a) Redução da complexidade da solução ao prover abstrações reusáveis: Um padrão arquitetural define elementos e relações entre esses elementos, diminuindo a quantidade de novos conceitos a serem utilizados na solução.
- b) Reuso: Como padrões arquiteturais são soluções de projeto para problemas recorrentes, é possível que a implementação (parcial ou total) do padrão possa ser reusada, facilitando o desenvolvimento.
- c) Geração de alternativas: Padrões arquiteturais distintos podem resolver o mesmo problema. Assim, é possível avaliar e escolher entre soluções possíveis, considerando os benefícios e as desvantagens de cada um deles.
- d) Facilidade de comunicação: Isso ocorre porque os padrões arquiteturais descrevem conceitos e elementos que estarão presentes no projeto.

Exemplos de padrões arquiteturais que foram popularizados por Buschmann et al. (1996): *layers* (camadas), *pipes e filters*, *Model-View-Controller*, *microkernel*. O padrão MVC é assunto da próxima subseção por ser o utilizado no desenvolvimento do sistema que é resultado da realização deste trabalho. Alguns dos padrões expostos em Buschmann et al. (1996) foram propostos por Garlan e Shaw (1994).

2.2.1.1 Padrão MVC

O *Model-View-Controller* é um padrão de projeto (*design pattern*), também conhecido como arquitetura, frequentemente utilizado para aplicações que necessitam manter múltiplas visões de um mesmo conjunto de dados. O MVC provê uma separação clara de objetos em três partes: modelo, visão e controle. Por causa dessa separação, múltiplas visões e controles podem interagir com um mesmo modelo. E novos tipos de visões e controles podem interagir com o modelo sem necessidade de realizar alterações no modelo definido no projeto.

O MVC é um conceito de desenvolvimento e projeto que visa organizar a separação de uma aplicação em três partes distintas (GAMMA et al., 2000; GONÇALVES et al., 2005; SWEAT, 2005).

- a) o modelo é o objeto de aplicação e está relacionado ao gerenciamento da visão. O modelo encapsula os dados do aplicativo e as regras de negócio; a visualização extrai dados do modelo e formata-os para apresentação
- b) A visão está relacionada a exibir os dados ou informações da aplicação.
- c) O controle coordena o modelo e a visão exibe a interface correta ou executa algum trabalho que a aplicação precisa completar. O controle direciona o fluxo do aplicativo e recebe entradas, traduzindo-as para o modelo e a visualização.

O MVC tem como principal objetivo separar dados ou lógica de negócios (*Model*) da interface do usuário (*View*) e o fluxo da aplicação (*Controller*), a ideia é permitir que uma mensagem da lógica de negócios possa ser acessada e visualizada por meio de várias interfaces. Na arquitetura MVC, a lógica de negócios (o modelo) não sabe quantas nem quais as interfaces com o usuário está exibindo o seu estado. Para a visão não importa a origem dos dados, mas ela precisa garantir que sua aparência reflita o estado do modelo, ou seja, sempre que os estados do modelo mudam, o modelo notifica as visões para que elas sejam atualizadas.

A Figura 1 exemplifica a relação entre essas três camadas.

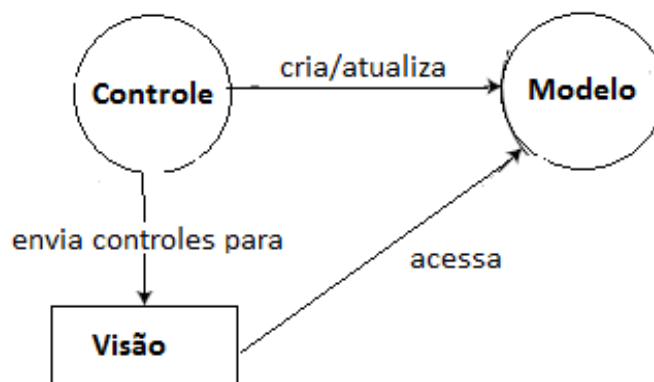


Figura 1 – Esquema básico do modelo MVC

Fonte: Adaptado de Pinto, Dionizio e Nunes (2010, p. 7).

Pela representação da Figura 1, a atuação do MVC inicia com um evento que é recebido pelo controle. O fluxo de dados representado na Figura 1 expressa que a partir de um evento iniciador, o controle altera a visão ou o modelo. A visão por sua vez obtém dados do modelo e o modelo atualiza a visão quando ocorrem mudanças nos dados.

Eventos geralmente fazem com que um controle altere um modelo, uma visão ou ambos. Sempre que um controle alterar dados ou propriedades de um modelo, todas as visões

dependentes são automaticamente atualizadas.

3 MATERIAIS E MÉTODO

Este capítulo contém os materiais e o método utilizados para a implementação do sistema. Os materiais se referem às ferramentas e as tecnologias, incluindo linguagem de programação, banco de dados, ambientes de desenvolvimento e a tecnologia *framework* Zend. O método se refere aos procedimentos utilizados no ciclo de vida do sistema, abrangendo a implantação e os testes do sistema.

3.1 MATERIAIS

As ferramentas e as tecnologias utilizadas para a implementação do sistema foram:

- a) PHP como a linguagem de programação;
- b) EclipsePHP Helios como IDE (*Integrated Development Environment*) de desenvolvimento;
- c) Zend Framework como *framework* de desenvolvimento PHP que provê a implementação do modelo MVC;
- d) MySQL para o banco de dados;
- e) Apache HTTP Server como servidor de aplicações para o PHP;
- f) Zend pdf, componente do próprio *framework* Zend, para a geração dos relatórios.

3.1.1 PHP

PHP é uma linguagem de programação dinâmica para *web* (PHP, 2011) que é processada no servidor, retornando para o cliente somente HTML. A linguagem PHP tem suporte para muitos dos bancos de dados existentes no mercado, o que torna simples a integração com aplicações que necessitem desta tecnologia. Essa linguagem também suporta outros protocolos como SMTP (*Simple Mail Transfer Protocol*), POP3 (*Post Office Protocol*) e IMAP (*Internet Message Application Protocol*).

3.1.2 EclipsePHP Helios

A ferramenta Eclipse é um ambiente integrado de desenvolvimento que foi desenvolvida utilizando a linguagem de programação Java e permite trabalhar com diversas linguagens de programação, como, Java, PHP e Ruby. Dentre as características da IDE Eclipse destacam-se:

- a) Disponibilização de funcionalidades como o auto-preenchimento de métodos, classes e variáveis, proporcionando maior velocidade no desenvolvimento.
- b) Integração de *plugins* externos, possibilitando maior quantidade de funcionalidades para desenvolvimento de projetos mais específicos.
- c) Utilização do conceito de perspectivas, que permite mudar de um ambiente de programação para um ambiente de depuração de forma facilitada, possibilitando a personalização de vários ambientes de trabalho dentro de um mesmo ambiente de desenvolvimento.
- d) Utilização de depuradores de código, facilitando a busca por erros no desenvolvimento de sistemas.

3.1.3 Zend Framework

O Zend Framework é um framework de desenvolvimento em PHP, que contém vários componentes que visam a melhor organização no desenvolvimento e também a reutilização de código para atividades básicas de um sistema (ZEND, 2011).

O Zend provê o padrão MVC. É um padrão de arquitetura que separa o GUI (*Graphical User Interface*) e *templates* (visão) da lógica de negócio (controle) e dos dados (modelo).

A flexibilidade quanto a utilização de bibliotecas e até mesmo outros *frameworks* é uma característica presente no Zend, ele permite que por exemplo, seja utilizado um *framework* de mapeamento objeto relacional na camada *model*, tal como o Doctrine ou o Propel.

O Zend é composto por vários componentes, sendo cada um responsável por uma atividade em específico, na Figura 2 estão listados os principais componentes desse *framework* organizados por categorias.

<p>Core:</p> <ul style="list-style-type: none"> Zend_Controller Zend_View Zend_Db Zend_Config Zend_Filter & Zend_Validate Zend_Registry Authentication and Access: Zend_Acl Zend_Auth Zend_Session <p>Internationalization:</p> <ul style="list-style-type: none"> Zend_Date Zend_Locale Zend_Measure <p>Http:</p> <ul style="list-style-type: none"> Zend_Http_Client Zend_Http_Server Zend Uri 	<p>Inter-application communication:</p> <ul style="list-style-type: none"> Zend_Json Zend_XmlRpc Zend_Soap Zend_Rest <p>Web Services:</p> <ul style="list-style-type: none"> Zend_Feed Zend_Gdata Zend_Service_Amazon Zend_Service_Flickr Zend_Service_Yahoo <p>Advanced:</p> <ul style="list-style-type: none"> Zend_Cache Zend_Search Zend_Pdf Zend_Mail/Zend_Mime <p>Misc!</p> <ul style="list-style-type: none"> Zend_Measure
--	---

Figura 2 – Principais componentes do Zend

Fonte: Allen e Lo (2007, p. 10).

- a) **Core** – na categoria Core encontram-se os componentes responsáveis por prover as funcionalidades necessárias para trabalhar com o modelo MVC. O componente `Zend_Controller` trata de prover os métodos e classes necessárias para a camada *controller*, assim como, o componente `Zend_View` provê todos os mecanismos necessários para se trabalhar com a camada *view*.
- b) **Authentication and Access** – os componentes desta categoria visam prover uma forma de autenticação e também de autorização de acesso a funcionalidades de um sistema.
- c) **Internationalization** – o Zend provê também componentes que proporcionam a internacionalização de um projeto, de uma maneira a auxiliar neste processo.
- d) **Http** – componentes que permitem o trabalho com requisições no protocolo HTTP (*Hypertext Transfer Protocol*), fazendo com que as requisições e respostas possam ser trabalhadas de uma forma mais organizada no sistema.
- e) **Inter-application communication** – os componentes `Zend_Json`, `Zend_XmlRpc`, `Zend_Soap` e `Zend_Rest` provem a comunicação com outras aplicações por meio do protocolo HTTP.
- f) **Web Services** – o Zend proporciona interação com vários serviços *webs*, tais como, o Yahoo pelo componente `Zend_Service_Yahoo` e o Flickr pelo `Zend_Service_Flickr`.

g) **Advanced** – Além de todos os componentes detalhados nos itens anteriores, o Zend dispõe de alguns componentes para tarefas menos comuns, como um gerador de arquivos pdf (Zend_Pdf) e um componente para envio de e-mail (Zend_Mail).

3.1.4 MySQL

O MySQL é um servidor e gerenciador de banco de dados relacional, que utiliza a linguagem SQL (MYSQL, 2011). Ele é de licença dupla (*software* livre e paga), projetado inicialmente para trabalhar com aplicações de pequeno e médio portes, mas atualmente atende também aplicações de grande porte. A seguir, algumas das principais características incorporadas na versão 5 do MySQL (MILANI, 2007):

a) **Visões** - são consultas pré-programadas ao banco de dados que permitem vincular duas ou mais tabelas e retornar uma única tabela como resultado. Além disso, podem ser utilizadas para filtrar informações, exibindo somente os dados específicos de uma determinada categoria de uma ou mais colunas da tabela. Com o uso de visões, operações frequentes com uniões de tabelas podem ser centralizadas. É possível também utilizá-las para controle de acesso, permitindo que determinados usuários acessem dados de uma visão, mas não às tabelas utilizadas por esta, restringindo acesso a informações.

b) **Cursores** - cursores possibilitam a navegação em conjuntos de resultados. De forma simples, é possível navegar pelos registros de uma tabela a partir de estruturas de repetição, permitindo realizar operações necessárias e transações à parte para cada linha da tabela.

c) **Information Schema** - tabelas responsáveis apenas pela organização dos recursos do banco de dados, conhecidos como dicionário de dados, ou metadados. Desta forma, é possível realizar consultas sobre a estrutura do banco de dados por meio dessas tabelas.

d) **Transações distribuídas XA** – esse conceito refere-se a uma espécie de extensão da ACID (Atomicidade, Consistência, Isolamento, Durabilidade) que fornece a possibilidade de gerenciamento dessas transações realizadas com a união de múltiplos bancos de dados (transações distribuídas) para a execução de uma mesma transação. Por exemplo, em determinadas situações pode surgir a necessidade de integração de duas bases de dados distintas, mas que em algum momento necessitem uma da outra

para realizar uma operação.

e) **Integridade referencial** - os relacionamentos entre tabelas distintas agora são gerenciados pelo banco de dados nos momentos de inclusão, alteração e exclusão.

Esse recurso visa manter confiáveis as relações existentes no banco de dados.

f) **Clusterização** - a clusterização é baseada na integração e sincronismo de dois ou mais servidores para dividirem a demanda de execuções entre si. Além da sincronização de um *cluster*, é possível especificar um balanceador de cargas. Desta forma, esse recurso não gerenciará apenas o redirecionamento de servidores secundários no caso de o primário sair do ar, mas sim balanceará as consultas recebidas pelo *cluster* e irá distribuí-las pelos servidores de acordo com sua disponibilidade.

3.1.5 Apache HTTP Server

O Apache HTTP Server é um servidor *web* compatível com o protocolo HTTP/1.1. Dentre as muitas funcionalidades disponibilizadas pelo servidor *web* Apache, estão a possibilidade de autenticar usuários, customização de mensagens de erro e trabalhar com módulos para atividades em particular.

Para que o servidor Apache esteja apto a trabalhar com a linguagem PHP, é necessária a presença do módulo do PHP para o Apache, denominado, *libapache2-mod-php5*. Com esse módulo o servidor Apache trabalha como um servidor de aplicações PHP, disponibilizando todas as funcionalidades competentes a um servidor de aplicações.

3.2 MÉTODO

O método consiste nas atividades que foram realizadas para o desenvolvimento deste trabalho. As principais etapas baseadas no modelo sequencial linear (PRESSMAN, 2005) foram: definição dos requisitos, análise e projeto, implementação e testes. O modelo sequencial linear foi escolhido porque o sistema modelado é simples e um dos acadêmicos autores deste trabalho conhece, ainda que superficialmente, os processos envolvidos em uma

vinícola. A seguir são definidas as etapas do modelo sequencial linear adotado:

a) Definição dos requisitos

A definição dos requisitos foi realizada como trabalho de estágio. Eles foram revisados e não foram necessárias complementações significativas.

b) Análise e projeto

Nesta fase os diagramas de classes e de entidade e relacionamento do banco de dados foram revisados. Esses modelos também foram definidos como trabalho de estágio.

c) Implementação

Na fase de implementação, inicialmente um cadastro foi implementado, incluindo as operações de inclusão, exclusão, consulta e alteração. A implementação desse cadastro permitiu entender o funcionamento básico do Zend Framework e do uso do modelo MVC. Em seguida o restante do sistema foi implementado, iniciando das funcionalidades mais simples. A opção pela implementação a partir das funcionalidades mais simples do sistema decorre da não familiaridade dos autores deste trabalho com o Zend. Assim, era necessário aprender a tecnologia à medida que o sistema era implementado.

d) Testes

Os testes realizados foram informais, definidos e executados pelos autores deste trabalho. Esses testes visaram a identificação de defeitos no código e verificar se os requisitos definidos para o sistema estavam sendo atendidos.

4 SISTEMA DESENVOLVIDO

Este capítulo apresenta o sistema para gerenciamento de vinícolas implementado como resultado do desenvolvimento deste trabalho. Da modelagem estão os casos de uso e diagramas de classes e de entidade e relacionamento do banco de dados. Esses são colocados para prover o entendimento dos requisitos do sistema e representam um resumo da modelagem realizada como trabalho de estágio.

4.1 APRESENTAÇÃO DO SISTEMA

O sistema para gerenciamento de vinícolas tem como finalidade o controle das principais atividades realizadas em uma vinícola. Com o uso do sistema os vinicultores poderão realizar o controle de entradas de estoque da uva *in natura*, bem como dos demais ingredientes para a obtenção dos produtos. Nesse sistema, haverá a possibilidade de inserção da ordem de produção dos diferentes tipos de vinhos e sucos provenientes da uva, bem como o controle de estocagem e posterior venda dos produtos finalizados.

4.2 MODELAGEM DO SISTEMA

O Quadro 1 resume os requisitos definidos para o sistema. Esses requisitos estão sob a forma de casos de uso. Foram definidos dois atores para o sistema. O ator **Funcionário** que opera na maioria dos casos de uso que contém atividades de cadastro, alteração, consulta e remoção. E o ator **Administrador** que herda todos os casos de uso do ator funcionário e ainda as funcionalidades de criação de novos usuários, vinícolas e vinicultores, que são funções exclusivas do administrador.

Quadro 1 – Casos de uso do sistema

Identificação	Objetivo	Ator
---------------	----------	------

		Adminis- trador	Funcio- nário
Administrar Ingredientes	Permitir que atores adicionem, consultem, alterem ou removam ingredientes (exemplo: açúcar, água, uva, etc).	X	
Administrar Produtos	Permitir que atores adicionem, consultem, alterem ou removam produtos (exemplo: vinho, suco, espumante, etc).	X	X
Administrar Clientes	Permitir que atores adicionem, consultem, alterem ou removam clientes.	X	X
Administrar Fornecedores	Permitir que atores adicionem, consultem, alterem ou removam fornecedores.	X	X
Emitir Ordens de Produção	O software deve permitir que atores emitam ordens de produção, onde serão definidos os ingredientes e suas respectivas quantidades utilizadas para posterior criação de um produto.	X	X
Administrar Tipos	Permitir que atores adicionem, consultem, alterem ou removam tipos de ingredientes e tipos de produtos.	X	X
Controlar saída de Produtos	Permitir o controle da saída de produtos de uma vinícola para um cliente, ou seja, a venda de um ou mais produtos para um cliente.	X	X
Controlar entrada de Ingredientes	Permitir o controle de entrada de ingredientes para uma vinícola, ou seja, a compra de um ou mais ingredientes de um fornecedor ou a entrada de ingrediente provenientes da produção da vinícola.	X	X
Administrar Vinícolas	Permitir que atores adicionem, consultem, alterem ou removam vinícolas.	X	
Administrar Vinicultores	Permitir que atores adicionem, consultem, alterem ou removam vinicultores.	X	
Administrar usuários e suas permissões	Permitir que os atores adicionem, consultem, alterem ou removam usuários, e também definam suas restrições de acesso.		

O diagrama de classes foi dividido em três partes para possibilitar uma explicação mais detalhada de suas classes e seus relacionamentos e facilitar a visualização dos mesmos.

Na Figura 3 estão representadas as classes relacionadas à manipulação de cadastros básicos do sistema. A classe **Vinicola** representa a instituição vinícola em si, ela está relacionada através de associação simples com as classes **Fornecedor**, **Cliente**, **Vinicultor** e **Usuario**. As classes **Cidade**, **Estado** e **Pais** representam em conjunto a localização das classes relacionadas a elas.

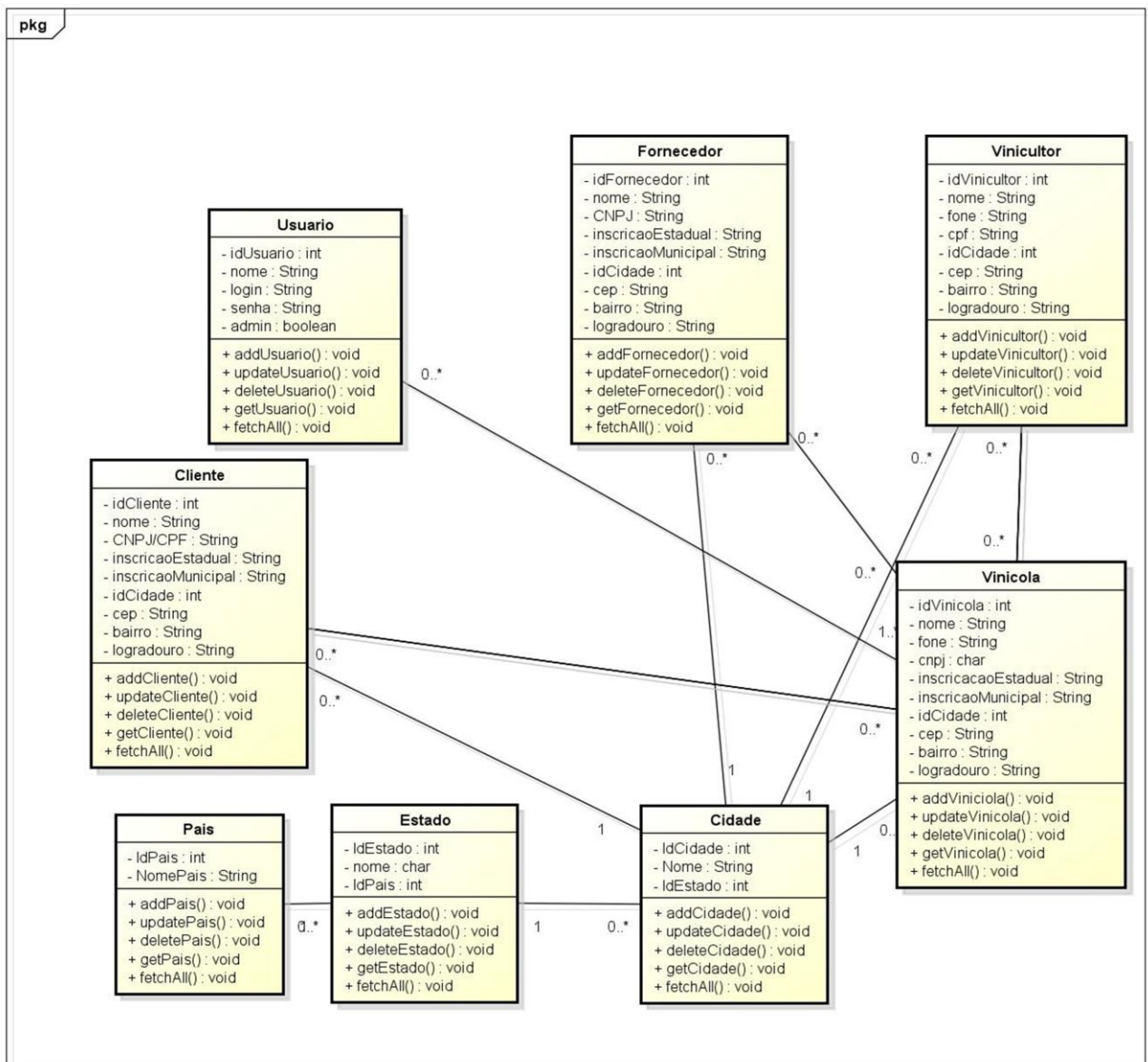


Figura 3 – Diagrama de classes (parte 1)

A Figura 4 apresenta a segunda parte do diagramas de classe. Nessa figura é apresentada a estrutura referente ao processo de produção e estoque. A classe **Vinicola** possui um relacionamento do tipo composição em relação as classes **IngredienteVinicola** e **ProdutoVinicola**. Fazendo, assim, com que uma **Vinicola** seja composta por vários produtos e ingredientes, sendo que quando uma determinada vinícola for excluída, os produtos (**ProdutoVinicola**) e os ingredientes (**IngredienteVinicola**) relacionados a ela, também serão excluídos.

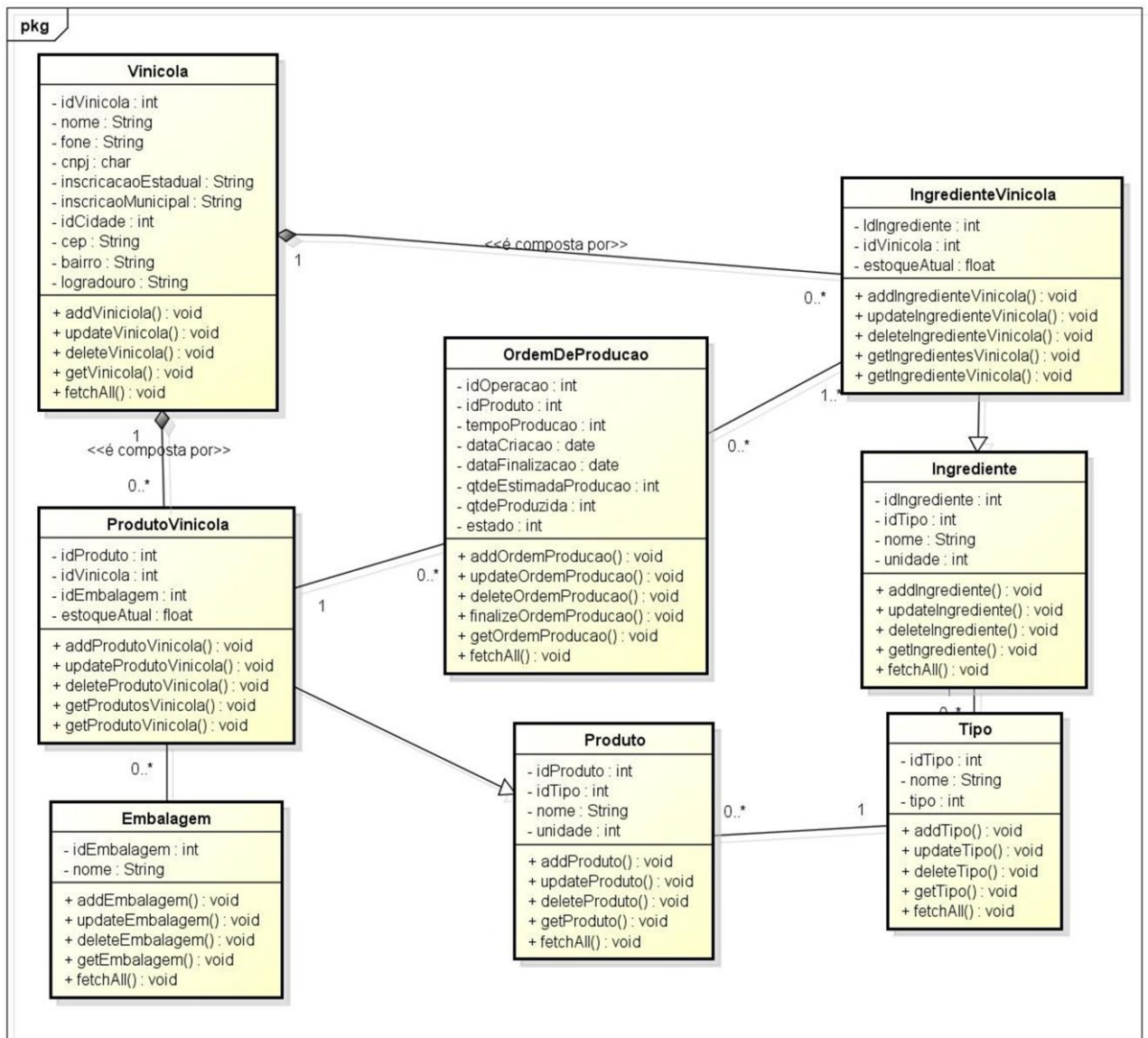
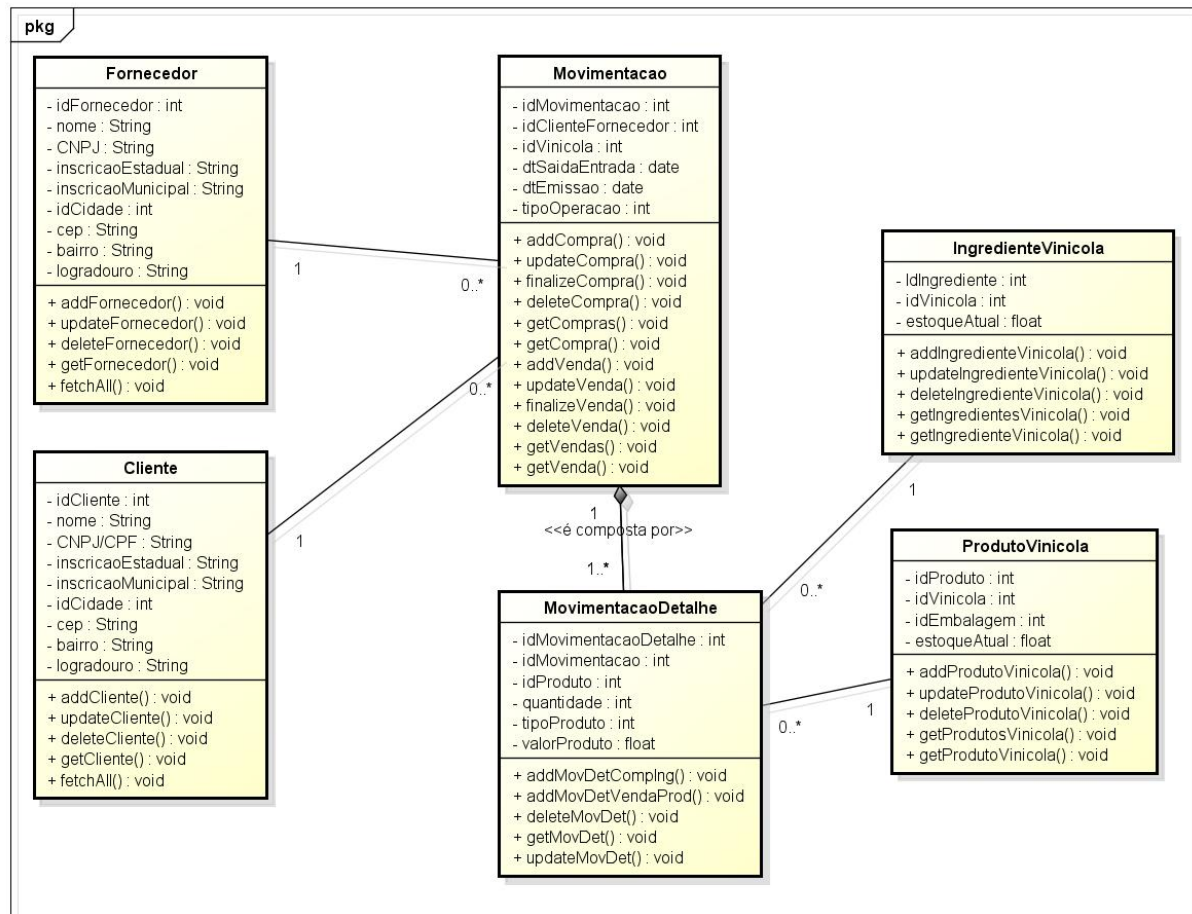


Figura 4 – Diagrama de classes (parte 2)

As classes **IngredienteVinicola** e **ProdutoVinicola** possuem um relacionamento de herança com as classes **Ingrediente** e **Produto** respectivamente, permitindo que as classes **IngredienteVinicola** e **ProdutoVinicola** possam utilizar-se dos métodos e atributos existentes nas classes **Ingrediente** e **Produto**.

Na última parte do diagrama de classes, que pode ser vista na Figura 5, está disposta a estrutura das classes em relação a movimentação de entrada e saída de produtos e ingredientes. As classes **Fornecedor** e **Cliente** estão relacionadas por uma associação simples com a classe **Movimentacao**. A classe **Movimentacao** por meio de relacionamento de composição com a classe **MovimentacaoDetalhe**, indica que uma movimentação de entrada ou saída será

composta de uma ou várias movimentações detalhe.



powered by astah

Figura 5 – Diagrama de classes (parte 3)

O diagrama de entidade relacionamento foi dividido em três partes para possibilitar uma explicação mais detalhada dos relacionamentos envolvendo as entidades. E, ainda, para que a visualização fosse facilitada, em decorrência do número de entidades envolvidas.

Na Figura 6 está a primeira parte do diagrama de entidade relacionamento. A entidade **vinicola** é a principal entidade do sistema, praticamente todas as entidades tem algum relacionamento com a mesma.

Nesse diagrama há quatro relacionamentos de muitos para muitos envolvendo a entidade **vinicola**. A entidade **fornecedor** tem uma relação de muitos para muitos com a entidade **vinicola** representada pela entidade associativa **fornecedor_vinicola**, indicando que um **fornecedor** poderá participar de várias **vinicolas** e uma **vinicola** poderá ter vários fornecedores. O mesmo processo acontece com a entidade **cliente**, a qual possui a entidade associativa **cliente_vinicola** e com a entidade **vinicultor** e a entidade associativa **vinicultor_vinicola**.

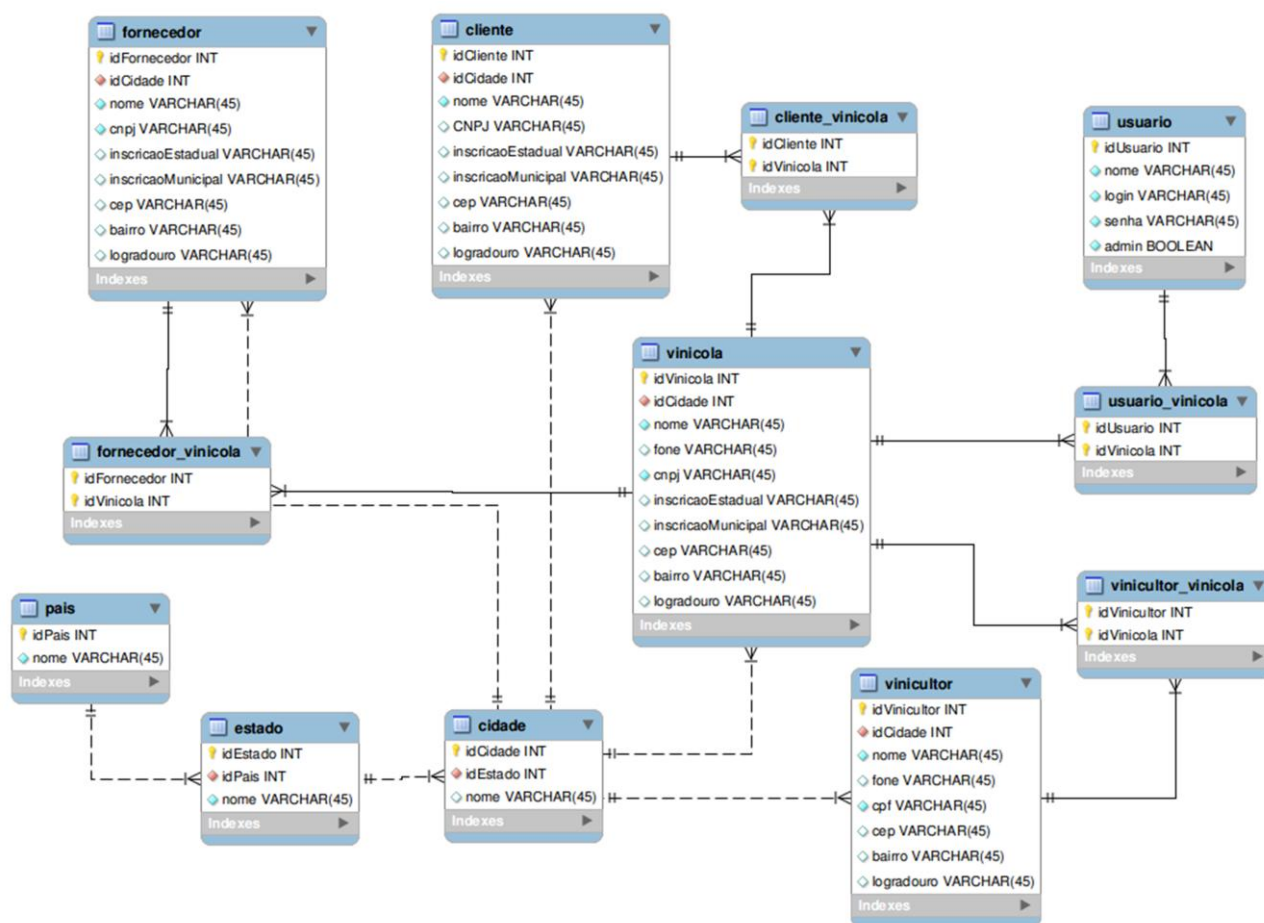


Figura 6 – Diagrama entidade relacionamento (parte 1)

A entidade **usuario** representa os usuários do sistema e também possui um relacionamento de muitos para muitos para com a entidade **vinicola**, representada pela entidade associativa **usuario_vinicola**.

As entidades **fornecedor**, **cliente**, **vinicola** e **vinicultor** estão relacionadas a entidade **cidade** por meio do atributo **id_cidade**, representando a cidade para cada entidade. A entidade **cidade** relaciona-se com a entidade **estado**, que relaciona-se com a entidade **pais**. Essas associações representam a localização física para as entidades relacionadas com a entidade **cidade**.

A segunda parte do diagrama de entidade relacionamento está apresentada na Figura 7. Essa segunda parte se propõe a explicar as entidades relacionadas ao processo de produção e estoque do sistema.

A entidade **vinicola** está relacionada com as entidades **produto_vinicola** e **ingrediente_vinicola**, ambas as entidades possuem o atributo **id_vinicola** e **estoqueAtual**. Esses atributos são responsáveis por identificar a quantidade de produtos e ingredientes em

estoque para cada vinícola em particular.

As entidades **produto** e **ingrediente** representam respectivamente, os produtos e ingredientes no sistema, ambos relacionam-se com a entidade **tipo**, responsável por definir tipos, tanto para produtos, quanto para ingredientes.

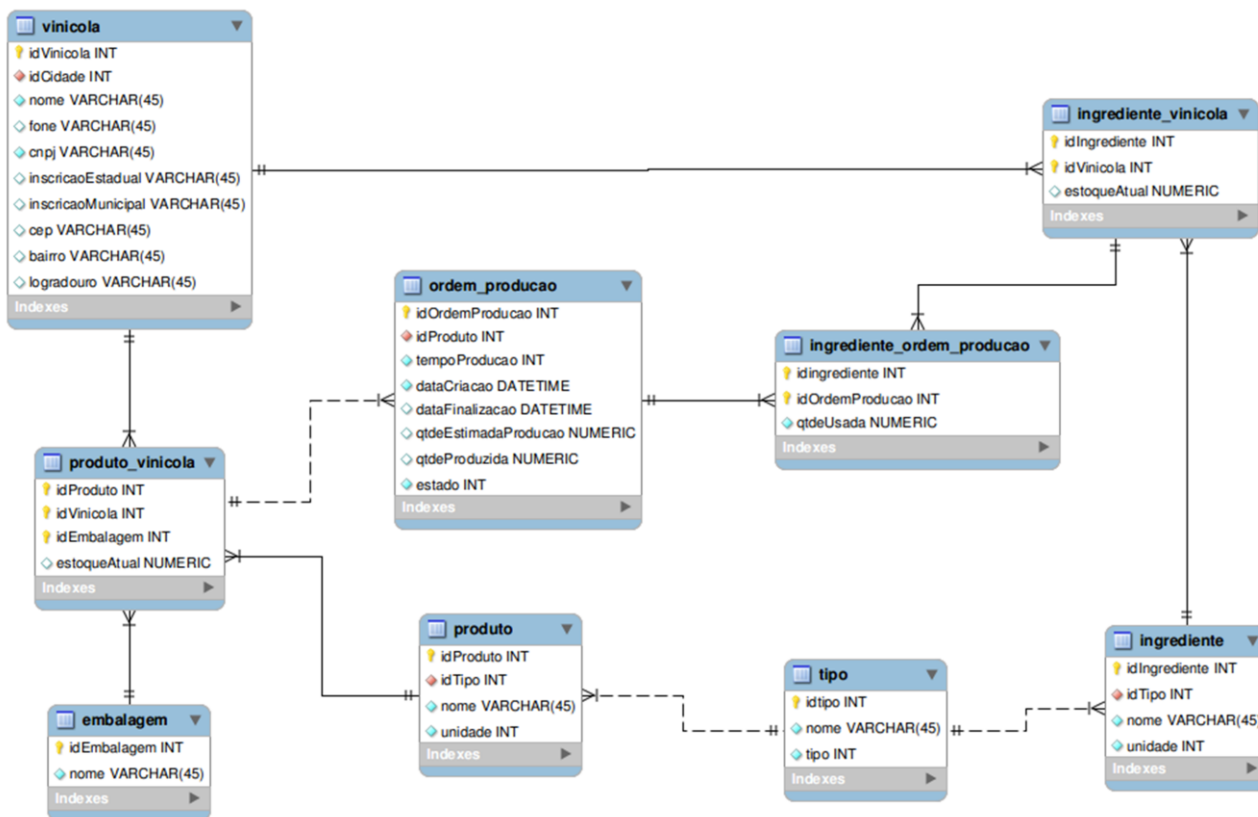


Figura 7 – Diagrama entidade relacionamento (parte 2)

A entidade **ordem_producao** representa uma ordem de produção para um produto, por meio do uso de um ou vários ingredientes e suas respectivas quantidades. Ela está relacionada a entidade **ingrediente_ordem_producao**, que representa quais ingredientes serão usados e em qual quantidade, para posterior geração de um produto, representado aqui pela entidade **produto_vinicola**. Como um produto pode ser embalado em diferentes recipientes, há a ligação do atributo **id_embalagem** com a entidade **embalagem**.

A terceira e última parte do diagrama de entidade relacionamento trata das movimentações de entrada e saída de produtos e ingredientes e pode ser vista na Figura 8. A entidade **movimentacao** representa uma entrada ou saída de um ou vários produtos, o atributo **tipoOperacao** é responsável por indicar se uma operação será de entrada ou saída, e assim definindo se ela será relacionada a entidade **fornecedor** ou a entidade **cliente**.

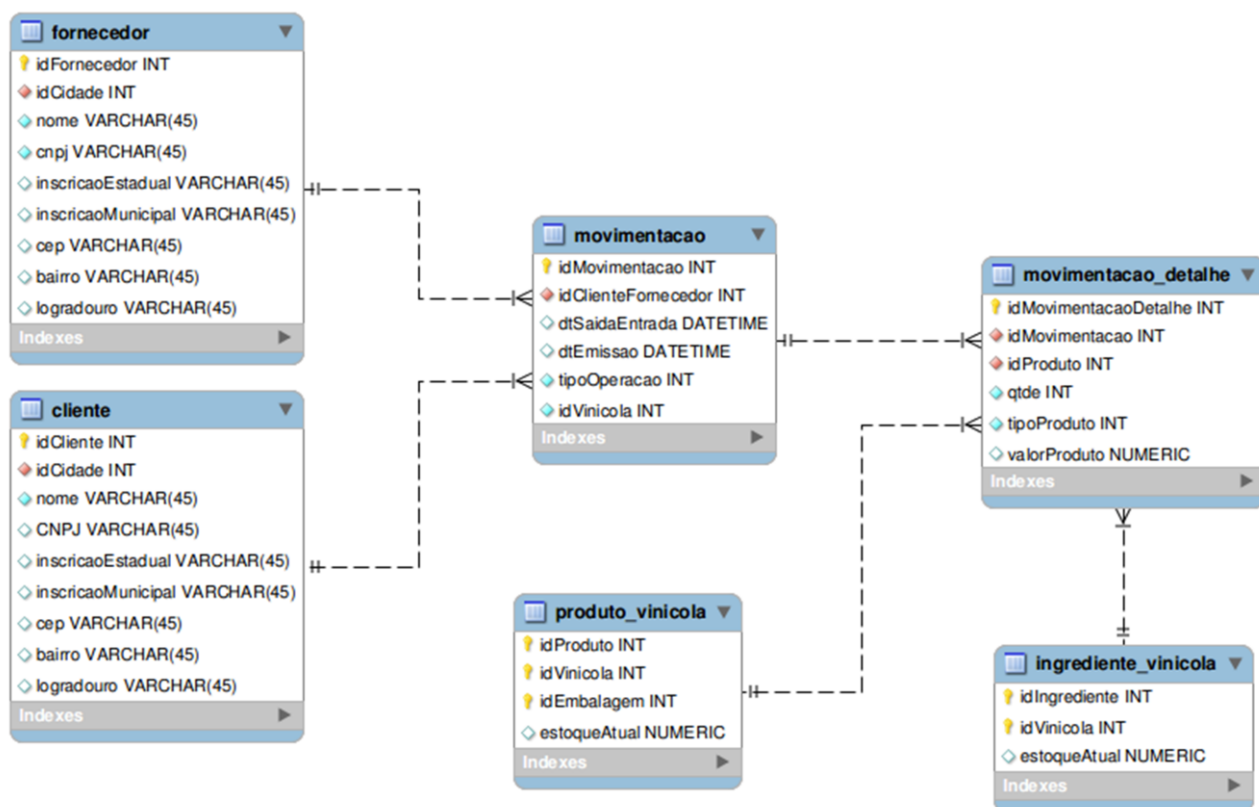


Figura 8 – Diagrama entidade relacionamento (parte 3)

A entidade **movimentacao_detalhe** é responsável por representar as informações de saída/entrada de um produto ou ingrediente em específico, o atributo **tipoProduto** é responsável por definir se os detalhes de uma movimentação pertencem à entidade **produto_vinicola** ou a entidade **ingrediente_vinicola**.

4.3 DESCRIÇÃO DO SISTEMA

Na Figura 9 está a tela de *login* ao sistema, responsável pela validação de acesso ao sistema por meio do *login*, da respectiva senha e da vinícola vinculada.

A imagem mostra uma tela de login centralizada em um fundo escuro. O formulário é composto por quatro elementos principais: um campo de texto rotulado 'Login:', um campo de texto rotulado 'Senha:', um menu suspenso rotulado 'Vinícola:' com a opção 'Vinícola São Felipino' selecionada, e um botão azul rotulado 'Entrar'.

Figura 9 – Tela de login

Como pode ser visto na Figura 9, além das opções de login e senha, o usuário deverá informar à qual vinícola o mesmo deseja acesso. Com isso, o sistema somente será aberto caso tanto o *login* quanto a senha estejam corretas para a respectiva vinícola. Se o usuário não tem acesso à vinícola informada, uma mensagem “Usuário sem acesso à vinícola selecionada” é apresentada.

Após ser realizada a validação de acesso e todos os dados estarem de acordo com o seu cadastro no banco de dados, o sistema será aberto com a tela inicial (Figura 10).

Sistema de gerenciamento de vinícolas Usuário logado: Rafael Albani

[Efetuar logout](#)

Cadastros | **Estoque** | **Produção** | **Movimentação**

Gerenciar Vinícolas

Nome

[Cadastrar nova vinícola](#)

Nome	CNPJ	Fone	Cidade
São Leopoldo	23432423423	234324	Pato Branco
Santa Fé	243432432		Santo André

TOTAL DE ITENS: 2

Figura 10 – Tela inicial do sistema

Na parte superior da tela da Figura 10 está a opção de “Logout” do sistema, função essa que permite ao usuário desconectar-se do sistema. Nesse mesmo local é apresentado o nome do usuário que está acessando o sistema. Ainda no topo da tela, podem ser vistos os menus de acesso às funções do sistema. Na parte central da tela é mostrada a barra de pesquisa para os itens apresentados em tela.

Nessa mesma figura observa-se a listagem de todas as vinícolas cadastradas no sistema. Cada linha da listagem representa uma vinícola cadastrada, com algumas informações tais como nome e telefone. Ao lado de cada linha listada, estão disponíveis a opção de “Editar” um cadastro e “Excluir”.

Na parte superior da listagem de vinícolas (Figura 10), encontra-se a opção de “Cadastrar nova vinícola”. Essa opção abre uma nova tela com as informações necessárias para o cadastramento de uma nova vinícola (Figura 11).

Ainda nessa mesma tela (Figura 10), está a opção de paginação. Essa função permite a

navegação pelas páginas de cadastro em caso de o resultado ser superior a uma tela (que é composta por cinco itens). É importante ressaltar que tanto a opção de pesquisa, quanto a opção de cadastramento, edição e exclusão são aplicadas a todos os cadastros do sistema.

Ao clicar na opção de “Cadastrar nova vinícola” será aberta a tela apresentada na Figura 11.



O formulário, intitulado "Gerenciar Vinícolas", contém o seguinte conteúdo:

- Título: **Gerenciar Vinícolas**
- Subtítulo: **Adicionar nova vinicola**
- Campos de entrada:
 - Nome
 - CNPJ
 - Inscrição Estadual
 - Inscrição Municipal
 - Cidade: Francisco Beltrão (menu suspenso)
 - Bairro
 - Logradouro
 - Cep
 - Fone
- Botão: Adicionar

Figura 11 – Cadastro de uma nova vinícola

O cadastro de vinícolas contém como informações obrigatórias os campos nome e CNPJ (Cadastro Nacional de Pessoas Jurídicas). Ao ser deixado um ou os dois desses campos em branco, será mostrado um alerta na tela e a inserção não será concluída com êxito, conforme Figura 12.

Gerenciar Vinícolas

Adicionar nova vinícola

Nome

• Por favor preencha o campo acima

CNPJ

• Por favor preencha o campo acima

Inscrição Estadual

Inscrição Municipal

Cidade

Bairro

Logradouro

Cep

Fone

Figura 12 – Cadastro de uma vinícola (2)

A opção “Editar” (Figura 13), cria uma nova tela com os dados da vinícola correspondente, possibilitando a alteração dos dados da mesma.

Gerenciar Vinícolas

Editar vinícola

Nome

CNPJ

Inscrição Estadual

Inscrição Municipal

Cidade

Bairro

Logradouro

Cep

Fone

Figura 13 – Edição de uma vinícola

Se selecionada a opção “Excluir”, constante na Figura 10, será mostrada na tela uma mensagem de confirmação de exclusão conforme a Figura 14.

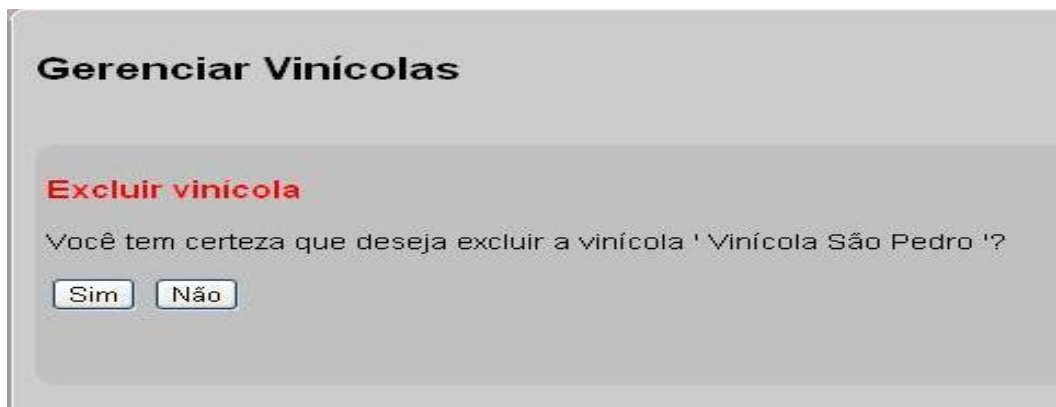


Figura 14 – Exclusão de uma vinícola

Caso o usuário clique no botão “Sim”, todos os dados referentes à vinícola selecionada serão excluídos do banco de dados. O sistema trará novamente a listagem geral de vinícolas cadastradas, já sem a vinícola que foi excluída.

Uma particularidade dos cadastros de cliente, fornecedor, usuário e vinicultor é a referência à qual vinícola o cliente está cadastrado (Figura 15). Com isso, o sistema permitirá apenas a movimentação de produtos e ingredientes aos cadastros vinculados a mesma.

The screenshot shows a web interface titled "Gerenciar Cliente". Below the title, there is a section titled "Adicionar novo Cliente" in red text. The form contains the following fields and values:

Nome	Teste
CNPJ	12.876.876/0011-77
Inscrição Estadual	
Inscrição Municipal	Isenta
Cidade	Francisco Beltrão
Bairro	Bairro
Logradouro	Log
Cep	85580-000
Fone	423432
Vinícolas em que será utilizado:	<input checked="" type="checkbox"/> Vinícola São Felipino <input checked="" type="checkbox"/> Vinícola São Pedro

At the bottom right of the form, there is a purple button labeled "Adicionar".

Figura 15 – Cadastro de clientes

A Figura 16 apresenta todo o menu de Cadastros.



Figura 16 – Menu de cadastros

Por meio do menu “Cadastros” podem ser cadastrados, editados, excluídos e pesquisados cada um dos itens do sistema. A seguir uma breve explicação sobre cada item:

- a) **Vinícola** - Nesse item de cadastro estão os dados da(s) vinícola(s) a ser(em) usada(s) no sistema. Desta forma, podem ser definidos posteriormente relatórios específicos de cada empresa, como estoque atual, lista de clientes e outros.
- b) **Fornecedor** - Nesse item de cadastro serão definidos todos os dados dos fornecedores. Nesses cadastros, deve ser definido a quais vinícolas o referido cadastro pertence, ou seja, com quais empresas o fornecedor possui vínculo.
- c) **Cliente** - Nesse item de cadastro serão definidos todos os dados dos clientes. Também devem ser definidas a quais vinícolas tal cadastro pertence, ou seja, a quais empresas o cliente possui vínculo.
- d) **Ingrediente** - Nesse item de cadastro serão definidos todos os cadastros dos ingredientes usados para fabricação do produto industrializado. O cadastramento nessa tela é feito para todas as vinícolas, não sendo inserido o estoque; dado esse que será definido posteriormente para cada vinícola em específico.
- e) **Produto** - Nesse item de cadastro serão definidos todos os cadastros dos produtos para posterior produção. O cadastramento nessa tela é feito para todas as vinícolas, não sendo inserido o estoque; dado esse que será definido posteriormente para cada vinícola em específico.
- f) **Vinicultor** - Nesse item de cadastro serão definidos todos os dados do vinicultor. Deve ser indicado a quais vinícolas tal cadastro pertence.

g) **Tipo** - Nessa tela serão realizados os tipos de ingredientes e produtos (vinho tinto, vinho branco, suco e outros). Esse cadastro é válido para todas as vinícolas.

h) **Usuário** - Nesse item de cadastro serão cadastrados os usuários que terão acesso ao sistema. Nessa mesma tela, poderá ser definido se o usuário possui o perfil de administrador. No caso de cadastros sem o preenchimento do perfil administrador, algumas funções do sistema serão bloqueadas para o mesmo (cadastro de vinícolas, usuários e vinicultor). Também no cadastramento do mesmo, será solicitado quais vinícolas o usuário terá acesso. Conforme Figura 17.

A imagem mostra a interface de usuário para gerenciar usuários. O título principal é "Gerenciar Usuários". Abaixo dele, há um subtítulo "Editar usuário" em vermelho. O formulário contém os seguintes campos e opções:

- Nome: Jaison
- Login: jaison
- Senha: (campo vazio)
- Administrador:
- Acesso as vinícolas: Vinícola São Felipino, Vinícola São Pedro

Um botão "Salvar modificações" está localizado na parte inferior direita do formulário.

Figura 17 – Cadastro de usuário

i) **Embalagem** - Nessa tela serão cadastradas todas as embalagens a serem usadas posteriormente para o engarrafamento dos produtos finalizados. Esse cadastro é válido para todas as vinícolas. Na aba "Estoque" estão as opções de produto e ingrediente (Figura 18).



Figura 18 – Menu de estoque

A aba "Estoque" (Figura 18) é destinada para inserção dos dados de ingredientes e produtos relacionados a vinícola acessada. Por meio desse recurso tem-se o detalhamento

exato do estoque de cada vinícola. Ao clicar na opção de ingrediente, o sistema trará a descrição e o estoque atual para a vinícola acessada. Uma alteração feita nesse controle e que não está presente nas demais telas do sistema é referente a edição do estoque. O sistema apenas permitirá a alteração do estoque, ficando a descrição do ingrediente sem a opção de alteração (Figura 19).

A interface apresenta o título "Gerenciar Estoque de Ingrediente" em negrito. Abaixo dele, o texto "Editar estoque de ingrediente" está em vermelho. O formulário contém dois campos de entrada: "Ingrediente" com o valor "Branca" e "Estoque atual" com o valor "1.300,000". Um botão azul com o texto "Editar estoque de ingrediente" está posicionado à direita dos campos.

Figura 19 – Edição de estoque de ingrediente

No cadastramento do estoque de produtos essa alteração também está presente. Na tela mostrada na Figura 20, pode ser visto o bloqueio tanto para o campo de descrição do produto quanto para a embalagem usada para o mesmo.

A interface apresenta o título "Gerenciar Estoque de Produto" em negrito. Abaixo dele, o texto "Editar estoque de produto" está em vermelho. O formulário contém três campos de entrada: "Produto" com o valor "Bordô", "Embalagem" com o valor "Garrafa PET" e "Estoque atual" com o valor "1.220,000". Um botão azul com o texto "Editar estoque de produto" está posicionado à direita dos campos.

Figura 20 – Edição de estoque de produto

A aba de “Produção” traz apenas um recurso que é responsável pelo cadastramento e gerenciamento das ordens de produção conforme Figura 21.

Cadastros | Estoque | **Produção** | Movimentação

Ordem de produção

Gerenciar Ordem de produção

[Cadastrar Ordem de Produção](#)

N° Ordem de Produção	Data de criação	Tempo de Produção	Estado	
6	12/03/2011	21 dias	aberta	Editar Excluir

« Primeira | « Anterior | 1 | Próxima » | Última »

TOTAL DE ITENS: 1

Figura 21 – Gerenciamento de ordens de produção

Estando na tela de ordem de produção, o sistema possibilita a inserção de uma nova ordem de produção por meio do *link* “Cadastrar Ordem de Produção”. Ao clicar nesse *link*, o sistema migra para uma nova tela conforme Figura 22.

Gerenciar Ordem de Produção

Adicionar nova Ordem de Produção

Ordem N°:

Produto:

Data de criação: Duração(em dias): Estado:

Figura 22 – Cadastro de ordem de produção

Na tela de cadastro de ordem de produção, estão disponíveis os campos: “Ordem N°.” que possui incremento automático gerado pelo sistema para o controle interno no banco de dados; uma lista de opções com todos os produtos cadastrados no sistema que possibilita ao usuário definir qual o produto a ser fabricado; “Data de criação”, “Duração (em dias)” e “Estado”. Por meio do campo de “Estado” será feito o controle da ordem de produção.

A ordem de produção tem três opções: aberta, em andamento e concluída. Em “aberta” todos os campos são liberados para edição. Nessa fase, são inseridos todos os dados da ordem

e também especificados os ingredientes necessários para o processo de fabricação. O estado “em andamento” bloqueará todas as edições da tela com exceção do campo estado. Nesse estado, por exemplo, a uva, o açúcar e demais ingredientes estão compondo o vinho. No estado “concluída” é feita a atualização do estoque do produto na aba de “Estoque”. Ao clicar no botão “Adicionar ordem de produção”, o sistema automaticamente migrará para a tela de editar (função essa que pode ser acessada pela tela principal do sistema) conforme Figura 23.

Gerenciar Ordem de Produção

Editar Ordem de Produção

Ordem N°:

Produto:

Data de criação: Duração(em dias): Estado:

Ingrediente	Quantidade	
Sarcosi	120.000	Editar Excluir
Bordô	120.000	Editar Excluir

[Adicionar ingrediente](#)

Figura 23 – Edição de ordem de produção

Na tela da Figura 23 é feito o detalhamento em relação aos ingredientes que compõem as ordens de produção. Enquanto o estado da ordem de produção estiver como “aberta”, o usuário poderá incluir e editar informações. Na tela exemplificada, poderá ser visto o bloqueio dos campos, não podendo os dados serem editados. No momento em que o usuário clicar na opção de “Adicionar Ingrediente”, o sistema abrirá a tela da Figura 24 na qual poderão ser definidos os dados específicos do mesmo.

Gerenciar Ingredientes Ordem de Produção

Adicionar ingrediente

Ingrediente: Isabel ▼

Quantidade:

Adicionar ingrediente

Figura 24 – Cadastro de ingredientes para ordem de produção

Na tela da Figura 24 são mostrados todos os ingredientes que possuem estoque positivo cadastrado e que ainda não fazem parte da ordem selecionada; juntamente com o campo de “Ingrediente”. O usuário poderá inserir a quantidade a ser usada na ordem e posteriormente salvar a inclusão clicando no botão “Adicionar ingrediente”. O *link* para exclusão da ordem de produção (Figura 21) poderá ser usado somente quando o estado da ordem de produção estiver como “aberta”.

A aba de “Movimentação” dispõe de duas funções como pode ser visto na Figura 25. Em movimentação poderão ser feitas as entradas de estoque de ingredientes e as vendas de produtos fabricados.

Sistema de gerenciamento de vinícolas

Cadastros Estoque Produção **Movimentação**

Compra - Ingredientes
Venda - Produtos

Gerenciar Compras de ingredientes

[Cadastrar Compra de ingredientes](#)

N° Compra	Fornecedor	Data de emissão	
1	Açúcar Alto Alegre	25/06/2011	Editar Excluir

« Primeira | « Anterior | 1 | Próxima » | Última »

TOTAL DE ITENS: 1

Figura 25 – Menu movimentação

Ao Clicar na opção de “Venda - Produtos” será aberta a tela da Figura 26 na qual poderão ser feitas as vendas de produtos fabricados.

Gerenciar Venda de produtos

[Cadastrar Venda de produtos](#)

Venda N°	Cliente	Data de emissão	
2	Seu Luís	25/06/2011	Editar Excluir

« Primeira | « Anterior | 1 | Próxima » | Última »

TOTAL DE ITENS: 1

Figura 26 – Listagem de venda de produtos

Na tela apresentada na Figura 26, poderão ser cadastradas as vendas por meio do *link* “Cadastrar Venda de produtos”. Ao clicar nesse *link* será apresentada a tela da Figura 27.

Gerenciar Venda de Produtos

Adicionar nova venda

Venda Nº:

Cliente:

Figura 27 – Cadastro de venda de produtos

Na parte superior da tela de gerenciar venda de produtos (Figura 26), o usuário poderá definir qual cliente está vinculado à respectiva venda. Logo após, ao clicar em adicionar, o sistema será direcionado para a função “editar” (função que pode ser acessada posteriormente pela tela principal de movimentação). Na tela de edição poderão ser definidos os produtos que pertencem ao cadastramento feito conforme Figura 28.

Gerenciar Venda de Produtos

Editar Venda de produtos

Venda Nº:

Cliente:

Produto	Quantidade	Valor
---------	------------	-------

[Adicionar produto à venda](#)

Figura 28 – Edição de venda de produtos

O *link* usado para a inclusão dos produtos é o “Adicionar produto à venda”. Ao Clicar nesse local, será aberta uma nova tela com os dados de “Produto”, “Embalagem”, “Quantidade” e “Valor”, conforme mostra a Figura 29.

Gerenciar Produtos

Adicionar novo produto

Produto: Suave

Embalagem: Garrafa PET

Quantidade:

Valor:

Adicionar

Figura 29 – Adição de produto a uma venda

Os dados de produto e embalagem serão trazidos em tela baseados nos cadastros feitos anteriormente na aba de Cadastros. O campo “Quantidade” será validado com os dados de estoque salvos na aba de “Estoque”.

4.4 IMPLEMENTAÇÃO DO SISTEMA

Nesta seção é apresentado o processo de autenticação de usuário, criação do menu do sistema e a estrutura MVC proporcionada pelo Zend Framework utilizada no desenvolvimento deste trabalho. A última tem o objetivo de exemplificar a divisão entre as camadas: *model*, *view* e *controller*, assim como a interação entre elas.

O processo de autenticação do sistema, ou seja, a validação de um usuário com seu usuário e senha no sistema, é feito pela classe **Application_Model_Auth** disponível na Listagem 1.

```

1 <?php
2
3 class Application_Model_Auth
4 {
5
6     public static function login($login, $senha, $idVinicola)
7     {
8         $dbAdapter = Zend_Db_Table::getDefaultAdapter();
9
10        //Inicia o adaptador Zend_Auth para banco de dados
11        $authAdapter = new Zend_Auth_Adapter_DbTable($dbAdapter);
12        $authAdapter->setTableName('usuario')
13                    ->setIdentityColumn('login')
14                    ->setCredentialColumn('senha');
15
16        //Define os dados para processar o login
17        $authAdapter->setIdentity($login)
18                    ->setCredential($senha);
19
20        //Efetua o login
21        $auth = Zend_Auth::getInstance();
22        $result = $auth->authenticate($authAdapter);
23

```

Listagem 1 – Classe Application_Model_Auth

A classe **Application_Model_Auth** contém apenas o método chamado **login**, o qual recebe os parâmetros correspondentes ao usuário, senha e a vinícola ao qual gostaria de se conectar. Por meio do componente **Zend_Auth**, componente do Zend Framework responsável por auxiliar na tarefa de autenticação de um usuário, o processo torna-se bem simples, como explicado pelos comentários disponíveis no código na Listagem 1.

A Listagem 2 contém o arquivo **navigation.xml**, nele está definida a estrutura de menu do sistema. O nó **estoque** representa um dos itens principais do menu, a descrição do mesmo é feita no nó **label** e o caminho chamado ao clicar no item de menu é representado pelo símbolo # no nó **uri**, fazendo com que quando clicado mantenha-se na mesma tela.

```
navigation.xml
50         <label>Embalagem</label>
51         <controller>embalagem</controller>
52         <action>index</action>
53     </embalagem>
54 </pages>
55 </home>
56 <estoque>
57     <label>Estoque</label>
58     <uri>#</uri>
59
60     <pages>
61         <ingrediente>
62             <label>Ingrediente</label>
63             <controller>ingrediente-vinicola</controller>
64             <action>index</action>
65         </ingrediente>
66         <produto>
67             <label>Produto</label>
68             <controller>produto-vinicola</controller>
69             <action>index</action>
70         </produto>
71     </pages>
72 </estoque>
73 <producao>
```

Listagem 2 – Estrutura do menu do sistema

O nó **pages** indica os sub itens do menu principal **estoque**, sendo eles representados pelos nós **ingrediente** e **produto**. Dentro de cada um dos sub itens existem três nós.

1. **Label**: contém a descrição do menu;
2. **Controller**: contém o *controller* que será requisitado no momento do clique no item;
3. **Action**: contém a ação que será utilizada pelo *controller* requisitado.

A Figura 30 apresenta a estrutura de um projeto criado a partir do Zend Framework. A pasta **application** contém os arquivos de desenvolvimento do sistema, tais como, arquivos de configuração da aplicação, classes, páginas HTML e a estrutura MVC provida pelo Zend.

A pasta **docs** é reservada para o armazenamento de arquivos referentes a documentação do sistema. A pasta **libray** contém o Zend Framework, é a pasta destinada às bibliotecas externas ao sistema.

Na pasta **public**, encontra-se o arquivo **index.php**, que é responsável pela inicialização do sistema. Esta é a única pasta a qual o servidor de aplicações tem acesso, sendo necessário colocar arquivos de CSS (*Cascading Style Sheets*), javascript e imagens dentro dela.

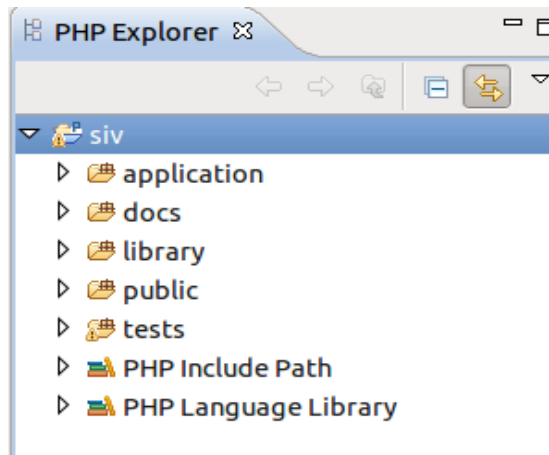


Figura 30 – Estrutura de uma aplicação utilizando Zend

A pasta **tests** fornece uma área para testes, sendo possível executar testes no sistema, sem interferência nos arquivos de desenvolvimento, disponíveis na pasta **application**.

A Figura 31 apresenta a estrutura da pasta **application** em mais detalhes, nela observa-se a estrutura MVC disponibilizada pelo Zend.

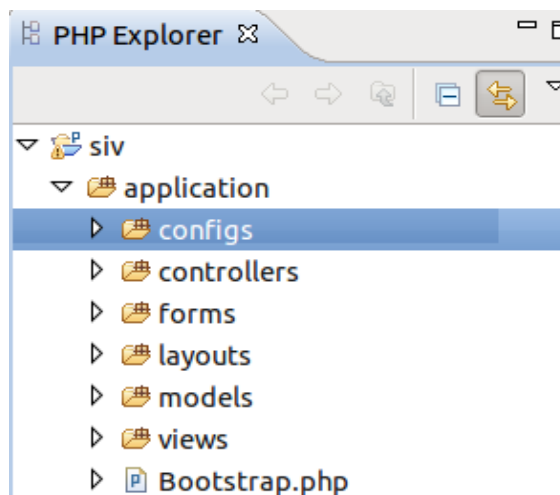


Figura 31 – Estrutura MVC disponibilizada pelo Zend

A pasta **configs** dispõe do arquivo **application.ini**, o qual pode ser visto em detalhes na Listagem 3.


```

1 [production]
2 phpSettings.display_startup_errors = 0
3 phpSettings.display_errors = 0
4 phpSettings.date.timezone = "America/Sao_Paulo"
5 includePaths.library = APPLICATION_PATH "../library"
6 bootstrap.path = APPLICATION_PATH "/Bootstrap.php"
7 bootstrap.class = "Bootstrap"
8 appnamespace = "Application"
9 resources.frontController.controllerDirectory = APPLICATION_PATH "/controllers"
10 resources.frontController.params.displayExceptions = 0
11 resources.db.adapter = PDO_MYSQL
12 resources.db.params.host = localhost
13 resources.db.params.username = root
14 resources.db.params.password = within
15 resources.db.params.dbname = siv
16
17 resources.layout.layoutPath = APPLICATION_PATH "/layouts/scripts/"
18 resources.view.doctype = "XHTML1_STRICT"
19
20 [staging : production]
21
22 [testing : production]
23 phpSettings.display_startup_errors = 1
24 phpSettings.display_errors = 1
25
26 [development : production]
27 phpSettings.display_startup_errors = 1
28 phpSettings.display_errors = 1
29 resources.frontController.params.displayExceptions = 1
30

```

Listagem 3 – application.ini

O arquivo **application.ini** contém as configurações para acesso ao banco de dados, nesse caso o MySQL. Esse arquivo também inclui configurações de caminhos para alguns diretórios do sistema, tais como, o diretório **library**, o qual contém as bibliotecas externas utilizadas pelo sistema.

As pastas **controllers**, **models** e **views**, são as representações da estrutura MVC provida pelo Zend, cada uma representando uma camada do padrão MVC.

Na Listagem 4 é possível observar a estrutura do arquivo **index.phtml** pertencente a camada *view* do sistema. Ele é responsável por criar a listagem das vinícolas cadastradas no sistema.

```

1 <?php
2 $this->title = "Vinícolas";
3 $this->headTitle($this->title);
4 ?>
5 <p>
6     <a
7         href="<?php echo $this->url(array('controller'=>'vinicola',
8             'action'=>'add'));?>">Cadastrar nova vinicola</a>
9 </p>
10 <table>
11     <tr>
12         <th>Nome</th>
13         <th>CNPJ</th>
14         <th>Fone</th>
15         <th>&nbsp;</th>
16     </tr>
17     <?php foreach($this->vinicolas as $vinicola) : ?>
18     <tr>
19         <td><?php echo $this->escape($vinicola->getNome());?></td>
20         <td><?php echo $this->escape($vinicola->getCnpj());?></td>
21         <td><?php echo $this->escape($vinicola->getFone());?></td>
22         <td><a
23             href="<?php echo $this->url(array('controller'=>'vinicola',
24                 'action'=>'edit', 'idVinicola'=>$vinicola->getIdVinicola()))?>">Editar</a> <a
25                 href="<?php echo $this->url(array('controller'=>'vinicola',
26                     'action'=>'delete', 'idVinicola'=>$vinicola->getIdVinicola()))?>">Excluir</a>
27         </td>
28     </tr>
29 <?php endforeach; ?>
30 </table>

```

Listagem 4 – index.phtml

Pode-se observar que a construção das URLs (*Uniform Resource Locator*), tem como parâmetros as variáveis **controller** e **action**, as duas representam respectivamente qual controlador será requisitado e qual ação será executada, nesta listagem estão as opções de “Cadastrar uma nova vinícola”, “Editar” e “Excluir” uma vinícola, todos apontando para o controlador vinícola e para as ações “add”, “edit” e “delete”, todas essas ações fazem referência aos métodos pertencentes a classe **VinicolaController** (Listagem 5).

No arquivo **index.phtml** (Listagem 4) ocorre o acesso a todos os métodos disponibilizados pela classe **Zend_View**, responsável por prover métodos para a camada *view* do sistema, possibilitando que sejam manipulados dados enviados pelo *controller* após recuperação dos mesmos na camada *model*.

A Listagem 5 apresenta a classe **VinicolaController**, responsável por receber todas as requisições feitas para o *controller* vinicola e executar uma determinada ação, e se necessário redirecionar para uma página na camada *view*. Todas as requisições do sistema seguem o mesmo padrão, para a manutenção de informações de um cliente por exemplo, foi criada a classe **ClienteController**, contendo basicamente as mesmas ações da classe **VinicolaController**, direcionadas para os propósitos da manutenção de clientes.

```

1  <?php
2
3  class VinicolaController extends Zend_Controller_Action
4  {
5
6      public function init(){}
10
11     public function indexAction(){}
16
17     public function addAction(){}
41
42     public function editAction() {}
72
73     public function deleteAction() {}
90
91
92 }
93

```

Listagem 5 – VinicolaController.php

Na classe **VinicolaController** pode-se observar que todos os métodos terminam com a palavra *action*. Isso identifica que cada método será responsável por lidar com uma ação em particular requisitada por um usuário através da interface, por exemplo, o método **indexAction()** será invocado quando for carregada a listagem de vinícolas (Figura 15), já o método **addAction()**, será invocado apenas quando clicado no link “Cadastrar nova vinícola” (Figura 15).

Os métodos para tratar uma requisição são disponibilizados pela extensão da classe **Zend_Controller_Action**. Essa extensão disponibiliza métodos de acesso as informações enviadas em uma requisição, redirecionamento de páginas e também métodos de acesso a camada *model* e *view*, para manipulação dos dados e disponibilização deles na interface do sistema.

A Figura 32 apresenta a estrutura da camada *model* do sistema, que é dividida em algumas pastas e classes responsáveis pelo mapeamento dos dados entre o modelo orientado a objetos e o modelo relacional utilizado pelo banco de dados.

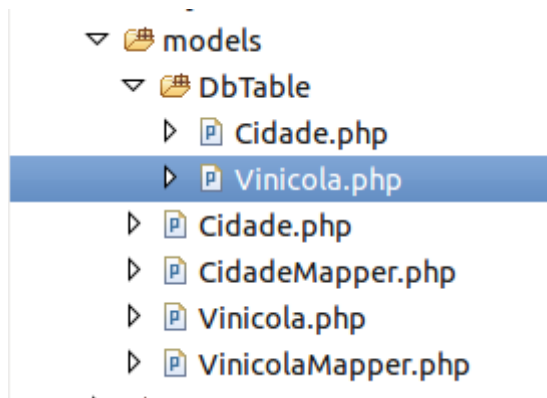


Figura 32 – Estrutura da camada model

As classes disponibilizadas na pasta **DbTable** são responsáveis por prover alguns métodos para manipulação de dados e cada uma delas representa uma tabela do banco de dados.

O Zend Framework não disponibiliza uma forma de mapeamento objeto relacional nativa, por esse motivo, na camada *models* foi utilizado o *design pattern Data Mapper*, que consiste em mapear uma classe e seus atributos. As classes **Cidade.php** e **Vinicola.php** disponíveis na pasta **models**, representam o modelo relacional. As classes **CidadeMapper.php** e **VinicolaMapper.php** são as responsáveis pelo mapeamento objeto relacional, permitindo que se possa trabalhar com objetos normalmente no sistema, ao invés de um *array* de informações resgatados de um banco de dados relacional. O *design pattern Data Mapper* foi utilizado para todas as classes que trabalham com alguma operação de manipulação direta com o banco de dados.

A Listagem 6 contém a classe **Application_Model_DbTable_Vinicola**. Essa classe dispõe de dois atributos, os atributos **\$_name** e **\$_primary**. O primeiro contém o nome da tabela a qual essa classe faz referência e o segundo define a chave primária da tabela, por padrão ele vem definido com o valor “id”.

```

1  <?php
2
3  class Application_Model_DbTable_Vinicola extends Zend_Db_Table_Abstract
4  {
5
6      protected $_name = 'vinicola';
7      protected $_primary = 'idVinicola';
8
9  }
10
11

```

Listagem 6 – Classe que representa a tabela vinícola

A classe **Application_Model_DbTable_Vinicola** estende da classe **Zend_Db_Table_Abstract**, a qual disponibiliza vários métodos para manipulação de dados de forma mais simples, por exemplo, pode-se utilizar o método **fetchAll()** para recuperar todos os registros da tabela Vinicola.

Na Listagem 7 está a classe **Application_Model_Vinicola**, nela são encontrados os atributos referentes a vinícola, assim como os métodos *getters* e *setters*.

```
1 <?php
2
3 class Application_Model_Vinicola {
4
5     protected $idVinicola;
6     protected $idCidade;
7     protected $nome;
8     protected $fone;
9     protected $cnpj;
10    protected $inscricaoEstadual;
11    protected $inscricaoMunicipal;
12    protected $cep;
13    protected $bairro;
14    protected $logradouro;
15
16    public function __construct(array $options = null) {}
21
22    public function __set($name, $value) {}
29
30    public function __get($name) {}
37
38    public function setOptions(array $options) {}
48
49    public function objectToArray () {
53
55        * Getters
57
58    public function getIdVinicola() {}
61
62    public function getIdCidade() {}
65
66    public function getNome() {}
69
```

Listagem 7 – Classe Vinicola

A classe **Vinicola** possibilita trabalhar com as informações de uma vinícola em forma de objeto, tanto na camada *view*, quanto na camada *controller*, permitindo a utilização do modelo orientado a objetos para manipulação de dados no sistema.

A Listagem 8 apresenta a classe **Application_Model_VinicolaMapper** que é responsável por fazer o mapeamento do modelo de objetos, representado neste caso pela classe **Application_Model_Vinicola** (Listagem 5) e para o modelo relacional, representado pela classe **Application_Model_DbTable_Vinicola** (Listagem 4).

```

1  <?php
2
3  class Application_Model_VinicolaMapper
4  {
5      protected $_dbTable;
6
7      public function setDbTable($dbTable) {}
17
18     public function getDbTable() {}
24
25     public function addVinicola(Application_Model_Vinicola $vinicola) {
26
27         $data = array(
28             'idCidade' => $vinicola->getIdCidade(),
29             'nome' => $vinicola->getNome(),
30             'fone' => $vinicola->getFone(),
31             'cnpj' => $vinicola->getCnpj(),
32             'inscricaoEstadual' => $vinicola->getInscricaoEstadual(),
33             'inscricaoMunicipal' => $vinicola->getInscricaoMunicipal(),
34             'cep' => $vinicola->getCep(),
35             'bairro' => $vinicola->getBairro(),
36             'logradouro' => $vinicola->getLogradouro();
37
38         $this->getDbTable()->insert($data);
39
40     }
41

```

Listagem 8 – Classe VinicolaMapper

O método **addvinicola** recebe um objeto da classe **Application_Model_Vinicola** (Listagem 8), em que os valores dos atributos do objeto são passados a um *array*, no método **\$this->getDbTable()->insert(\$data)** a primeira parte da instrução (**\$this->getDbTable**) recupera um objeto da classe **Application_Model_DbTable_Vinicola** (Listagem 6). Essa mesma classe por meio do método **insert(\$data)**, executa a operação de inserção dos dados contidos no *array* **\$data** no banco de dados.

Na Listagem 9 pode ser observado por meio do método **addAction()**, um exemplo de interação das três camadas do modelo MVC.

```

1 |<?php
2
3 | class VinicolaController extends Zend_Controller_Action
4 | {
5
6 |     public function init()
10
11 |     public function indexAction()
16
17 |     public function addAction()
18 |     {
19 |         $form = new Application_Form_Vinicola();
20 |         $form->submit->setLabel('Adicionar');
21
22 |         // recuperando cidades
23 |         $cidade = new Application_Model_CidadeMapper();
24 |         $form->idCidade->addMultiOptions($cidade->getCidades());
25 |         $form->idCidade->addValidator('inArray', false, array(array_keys($cidade->getCidades())));
26
27 |         if($this->getRequest()->isPost() {
28 |             if($form->isValid($this->getRequest()->getPost()) {
29 |                 $vinicola = new Application_Model_Vinicola($form->getValues());
30 |                 $mapper = new Application_Model_VinicolaMapper();
31 |                 $mapper->addVinicola($vinicola);
32
33 |                 $this->_helper->redirector('index');
34 |             } else {
35 |                 $form->populate($form->getValues());
36 |             }
37 |         }
38
39 |         $this->view->form = $form;
40 |     }

```

Listagem 9 – Exemplo de action

Quando um usuário escolhe a opção “Cadastrar uma nova vinícola” (Figura 10), pela definição do valor dos dois atributos **controller** e **action**, como **vinicola** e **add** respectivamente (Listagem 4), o método **addAction()** será executado.

Nas duas primeiras linhas do método **addAction()**, será instanciado um novo objeto da classe **Application_Form_Vinicola**. Essa classe é responsável por criar o formulário para cadastro de uma vinícola. Nesse processo pode-se encontrar a interação da camada *controller* com a camada *view*, por meio da criação de um formulário de dados que será apresentado para o usuário final

O terceiro comando instancia um novo objeto da classe **CidadeMapper**. O próximo comando atribui um vetor de cidades resgatadas do banco pelo método **getCidades()**, para um componente de lista do **\$form**. Aqui pode ser observada a busca das cidades no banco de dados por meio da camada *model* e a atribuição das cidades resgatadas para um elemento do formulário que será apresentado ao usuário final, pertencente a camada *view*. Isso mostra claramente o papel da camada *controller* no modelo MVC.

5 CONCLUSÃO

O objetivo deste trabalho de conclusão de curso é a implementação de um sistema de gerenciamento de vinícolas utilizando o *framework* de desenvolvimento Zend. Esse *framework* utiliza a linguagem PHP e por padrão o modelo MVC como estrutura de projeto.

Devido a existência de poucas vinícolas na região sudoeste do Paraná e ao fato dos integrantes não disponibilizarem de tempo para deslocarem-se até outras regiões para o levantamento de requisitos, a etapa de modelagem teve como base apenas o conhecimento prévio dos integrantes do projeto. Um dos autores deste trabalho conhece como ocorrem esses processos, ainda que não seja de forma explicitamente técnica.

O sistema desenvolvido atende os objetivos propostos. Os cadastros pretendidos estão implementados e o controle de ordem de produção é realizado. O sistema é simples, mas abrange as principais funcionalidades de uma vinícola e visa auxiliar o gerente no processo de tomada de decisões e realizar o controle gerencial da fábrica e da venda ao consumidor.

Um ponto a ser destacado no desenvolvimento deste trabalho é a curva de aprendizado longa quanto ao Zend Framework e também quanto a arquitetura MVC, ambos utilizados no desenvolvimento deste trabalho. Por isso foi necessário iniciar a implementação do sistema pelas funcionalidades mais simples. Assim, a preocupação centrou-se no aprendizado da tecnologia ao invés de em regras de negócio.

Após uma etapa de estudos sobre o Zend Framework e o padrão de desenvolvimento MVC, os quais possuem uma vasta documentação na internet, em sua grande maioria em inglês, o trabalho fluiu de maneira mais natural. Isso ocorreu devido às vantagens disponibilizadas por ambos, que possibilitaram uma melhor divisão do projeto e facilidades quanto ao desenvolvimento do projeto.

A utilização de ferramentas para modelagem e de *frameworks* para o desenvolvimento de sistemas, no caso deste trabalho, o Zend Framework, no início exige um bom estudo e demanda um certo tempo para adaptação aos padrões adotados pelo *framework*.

No entanto verificou-se, que os benefícios do uso de *frameworks*, em especial o Zend, por ter sido o utilizado, são grandes, já que existe a possibilidade de uma melhor organização das camadas do projeto e também de reutilização de componentes. Esses componentes, em sua grande maioria, já disponibilizados pelo Zend Framework.

REFERÊNCIAS

ALLEN, R., LO, N. **Zend framework in action**. MEAP Editon, 2007.

BASS, L, CLEMENTS, P, KAZMAN, R. **Software architecture in practice**, 2a ed. Addison-Wesley Professional, 2003.

BUSCHMAN, F. et al., **Pattern-oriented software architecture**, vol. 1: A System of Patterns. John Wiley & Sons, 1996.

IEEE. INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS. **IEEE Std 1471-2000: recommended practice for architecture description of software-intensive system**, IEEE Computer Society, 2000, 29 p.

GAMMA, et al. **Padrões de projeto: soluções reutilizáveis de software orientado a objetos**. Porto Alegre: Bookman, 2000.

GARLAN, D., SHAW, M. **An introduction to software architecture**. SEI Relatório Técnico CMU/SEI-94-TR-21, 1994.

GERMOGLIO, G. **Arquitetura de software**, Texas: Rice University, 2010.

GONÇALVES, R. F., GAVA, V. L., PESSÔA, M. S. P. SPINOLA, M. M. **Uma proposta de processo de produção de aplicações web**. Revista Produção, vol. 15, n. 3, p. 376-389, Set./Dez. 2005.

MCCONNELL, S. **Code complete**. 2A ed., Microsoft Press, 2004.

MILANI, A. **MySQL – guia do programador**. São Paulo: Novatec, 2007.

MYSQL. **MySQL**. Disponível em: <<http://www.mysql.com>>. Acesso em: 29 mar. 2011.

PERRY, D. E., WOLF, A. L. *Foundations for the study of software architecture*. **SIGSOFT Software Engineering Notes**, vol. 17, nun 4, p. 40–52, out. 1992.

PINTO, G. C; DIONIZIO, I. N.; NUNES, L. D. V. **Pacote de desenvolvimento ASP.NET MVC**. VII Simpósio de Excelência em Gestão e Tecnologia (SEGeT), P. 1-11, 2010.

PHP. **Linguagem PHP**. Disponível em: <<http://www.php.net>>. Acesso em: 05 mar. 2011.

PRESSMAN, R. **Engenharia de software**, 2005. Rio de Janeiro: McGrawHill.

SWEAT, J. E. **PHP architect's: guide to PHP design patterns**. Marco Tabini and Associates. Canada, 2005.

ZEND. **Zend framework**. Disponível em <<http://framework.zend.com/manual/en/>>. Acesso em: 21 mar. 2011.