UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

COORDENAÇÃO DE INFORMÁTICA

TECNOLOGIA EM ANALISE E DESENVOLVIMENTO DE SISTEMAS

MARCO AURÉLIO PEREIRA DA SILVA

NEWTON MACEDO NETO

WILLIAN RICARDO PEREIRA DA SILVA

SISTEMA WEB DE GERENCIAMENTO ACADÊMICO DA ESCOLA DE MÚSICA COVER'S.

TRABALHO DE CONCLUSÃO DE CURSO

PONTA GROSSA

2011

MARCO AURÉLIO PEREIRA DA SILVA

NEWTON MACEDO NETO

WILLIAN RICARDO PEREIRA DA SILVA

SISTEMA WEB DE GERENCIAMENTO ACADÊMICO DA ESCOLA DE MÚSICA COVER'S.

Trabalho de Conclusão de Curso apresentada como requisito parcial à obtenção do título de Tecnólogo em Analise e Desenvolvimento de Sistemas, da coordenação de informática, da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Rogério Ranthum

PONTA GROSSA

2011

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Ponta Grossa
Diretoria de Graduação e Educação Profissional
Coordenação de Informática
Tecnologia em Análise e Desenvolvimento de Sistemas

TERMO DE APROVAÇÃO

SISTEMA WEB DE GERENCIAMENTO ACADÊMICO DA ESCOLA DE MÚSICA COVER'S

por

MARCO AURÉLIO PEREIRA DA SILVA
NEWTON MACEDO NETO
WILLIAN RICARDO PEREIRA DA SILVA

Este(a) Trabalho de Conclusão de Curso foi apresentado(a) em 18 de Novembro de 2011 como requisito parcial para a obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas. O(a) candidato(a) foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Rogério Ranthum
Prof.(a) Orientador(a)

_____
Saulo Jorge Beltrão de Queiroz
Membro Titular

_____
Lourival A. de Góis
Membro titular

_____
Helyane Bronoski Borges
Responsável pelos Trabalhos
de Conclusão de Curso

_____
André Koscianski
Coordenador do Curso
UTFPR – Campus Ponta Grossa

Dedicamos este trabalho a todas as pessoas que prestaram seu apoio, seja de forma direta ou indiretamente, nos passando confiança e credibilidade para que pudéssemos concluir a realização desse trabalho de forma eficaz.

AGRADECIMENTOS

À Deus, que nos iluminou em todos os momentos, tanto nos fáceis como nos difíceis, proporcionando força de vontade e entusiasmo para a realização do trabalho.

Ao nosso Orientador, Prof. Rogério Ranthum, pelo incentivo, simpatia, auxilio durante as atividades e discussões sobre elaboração, esclarecimento de dúvidas e ajuda na busca de soluções ao enfrentar impasses no decorrer do trabalho.

Às nossas famílias e namoradas, que por muitas horas tiveram que se conformar e suportar a nossa ausência em diversas ocasiões, mas que mesmo assim compreenderam esses acontecimentos e nos apoiaram nos passando força e vontade para prosseguir por essa etapa.

Aos nossos professores, que no decorrer do curso tiveram importante contribuição, proporcionando um aprendizado de qualidade, que será de ótima utilidade no futuro, em nossas vidas profissionais.

Ao coordenador do curso de Tecnologia em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná – Campus Ponta Grossa, André Koscianski.

Aos colegas de classe, pelo apoio, auxílio em casos de dúvidas ou dificuldades para entender a assunto tratado em aula e também pelas horas de diversão, que nos ajudou a muitas vezes relaxar, proporcionando tranquilidade e conforto para a assimilação do conteúdo visto em sala.

O simples bater de asas de uma borboleta pode resultar em consequências gigantescas como um furacão (Edward Lorenz – O efeito borboleta e a descoberta do caos).

RESUMO

PEREIRA DA SILVA, Marco Aurélio; PEREIRA DA SILVA, Willian Ricardo; MACEDO, Newton. **SISTEMA WEB DE GERENCIAMENTO ACADÊMICO DA ESCOLA DE MÚSICA COVER'S**. 2011. 57 Pag. Trabalho De Conclusão De Curso (Graduação em Tecnologia e Análise e Desenvolvimento de Sistemas), Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2010.

O trabalho visa estudar e descrever tecnologias ainda não trabalhadas pelos acadêmicos em questão. Entre essas tecnologias, estão: JPA (API de Persistência do Java), Hibernate (Framework para persistência), ICE Faces, JSF e Ajax. A partir desse estudo, foi desenvolvido um Sistema Web para suprir as necessidades da Escola de Música Cover's. A escolha de um Sistema Web foi tomada pelo fato de que estes tipos de sistema proporcionam ao usuário um acesso mais fácil, sem que seja necessário um investimento de equipamentos para manter a aplicação. Serão demonstrados também os resultados do uso dessas tecnologias no Sistema em questão junto com os benefícios trazidos pelo uso das mesmas além de uma descrição do funcionamento do sistema e seus módulos. Com a possível implantação do sistema para uso da escola Cover's, foi montado uma lista de módulos que poderão ser desenvolvidos futuramente e anexados ao sistema.

Palavras-chave: Gerenciamento, Tecnologia, Aplicativos WEB.

ABSTRACT

SILVA, Marco Aurélio Pereira da; SILVA, Willian Ricardo Pereira da; NETO, Newton Macedo. SISTEMA WEB DE GERENCIAMENTO ACADÊMICO DA ESCOLA DE MÚSICA COVER'S. 2011. 57 Pag. Trabalho De Diplomação (Graduação em Tecnologia e Análise e Desenvolvimento de Sistemas) - Federal Technology University - Paraná. Ponta Grossa, 2011.

This Project studies and describes technologies that were not yet used by its academics authors. Between these technologies, are: JPA (Java Persistence API), Hibernate (Persistence Framework), ICE Faces, JSF and Ajax. Based on this research, A Web System was developed to supply the needs of Cover's – Musical School. The choice of a Web System was taken because this kind of system does not need high performance computer hardware, discarding its investment, besides it is easier to be manipulated by the users. Also will be shown the results of using these technologies in the system, as their benefits of their usage and a description of the system and its modules. With the possibility of implanting the system in the school, there was created a list of new modules that may be developed and add to the system in a near future.

Keywords: Management, Technology, WEB Application.

LISTA DE SIGLAS

Ajax	Asynchronous JavaScript and XML
API	Application Programming Interface
Crud	Create, Read, Update e Delete
CSS	Cascading Style Sheets
DOM	Document Object Model
ERP	Enterprise Resource Planning
HTTP	Hypertext Transfer Protocol
JDBC	Java Database Connectivity
JPA	Java Persistence API
JSF	Java Server Faces
MVC	Model, View e Control
SGBD	Sistema de Gerenciamento de Banco de Dados
SIG	Sistema de Informação Gerencial
SIO	Sistemas de Informação Operacional
SQL	Structured Query Language
URL	Uniform Resource Locator
XHTML	Extensible Hypertext Markup Language
XML	Extensible Markup Language

LISTA DE FIGURAS

LISTA DE CÓDIGOS FONTE

SUMÁRIO

1 INTRODUÇÃO

Há diversas maneiras de organizar uma empresa para que atenda às exigências do mercado. Funcionários qualificados e políticas corporativas consistentes contribuem para a estabilidade organizacional da empresa. Porém, o auxílio de um software robusto eficaz e seguro garantem a integridade dos dados e satisfaz as necessidades para as quais o mesmo foi desenvolvido (CALDWELL, 2008).

Sistemas desenvolvidos para funcionar em ambiente Web trazem benefícios aos usuários em comparação com sistemas que precisam ser instalados no computador. Entre elas está o fato de que o sistema não precisará ser instalado individualmente em cada computador que fará uso do mesmo.

O sistema fica hospedado em um servidor, que pode ser próprio ou de terceiros, porém este por sua vez precisa ter garantia de disponibilidade, já que se este estiver desligado ou sem acesso à internet, o sistema estará indisponível.

Dessa maneira, o sistema pode ser acessado via URL a partir de qualquer computador, diminuindo custos de hardware e possibilitando acesso sem precisar estar necessariamente na empresa.

Este projeto tem por objetivo fazer o estudo de tecnologias existentes no mercado que ainda não foram trabalhadas pelos acadêmicos autores do trabalho e aplicá-las no desenvolvimento de um sistema para facilitar o gerenciamento dos dados referentes a alunos e funcionários da Escola de Música Cover’s.

O sistema desenvolvido permitiu que os usuários do sistema visualizassem horário de aulas e outras rotinas referentes à área acadêmica da escola. Os funcionários podem matricular alunos, marcar horário de aulas e etc. Essas funcionalidades podem ser acessadas via web, sua atualização em tempo real permite que os alunos tomem conhecimento das alterações sem precisar telefonar ou comparecer à escola.

1.1. JUSTIFICATIVA

O motivo da realização deste trabalho é estudar e aplicar novas técnicas, ferramentas e tecnologias no desenvolvimento de um sistema visando atender às necessidades de uma empresa. O mesmo tem por função prover uma solução atrativa e eficiente, visando uma melhor organização de seus dados.

Um sistema informatizado elimina a necessidade de guardar informações em arquivos físicos e otimiza o controle e manipulação dos dados. Para o desenvolvimento do mesmo, faremos uso de novas tecnologias, como o *framework* ICE Faces, Ajax, JSF, Hibernate.

1.2. OBJETIVOS

1.2.1. Objetivo Geral

Efetuar a analise, desenvolver e implantar uma aplicação WEB utilizando novas tecnologias ágeis para o desenvolvimento do sistema.

1.2.2. Objetivos Específicos

- Analisar e entender o funcionamento do sistema de funcionamento atual da empresa.

- Realizar a extração dos requisitos para o sistema a partir dos dados coletados.

-Utilizar as tecnologias ICE Faces, JSF, Ajax e HIBERNATE para o desenvolvimento do sistema.

- Desenvolver uma aplicação WEB para a realização e o gerenciamento de rotinas acadêmicas realizadas.

1.3. PROBLEMA

Atualmente a escola de música Cover’s não gerencia seus dados através de um sistema informatizado, todos os tratamentos tais como: agendamento de aulas, cadastros e matrículas são feitos manualmente. Tais tarefas tornam-se mais trabalhosas e dispendiosas, além da dificuldade de armazenamento, na qual com o tempo e o material utilizado pode se deteriorar, e perder-se.

1.4. ORGANIZAÇÃO DO TRABALHO

Este trabalho está dividido em 5 capítulos. O capitulo 2 descreve uma revisão literária sobre sistemas de informação, o que são e como são classificados, e sistemas WEB e seus conceitos.

O capitulo 3 apresenta as tecnologias utilizadas no trabalho.

O capitulo 4 consiste na descrição da implementação das tecnologias estudas para a resolução do problema proposto.

Finalmente o capitulo 5 conclui o trabalho apresentando os resultados que foram obtidos a partir dos estudos inicias em relação aos objetivos iniciais, sugestões para trabalhos futuros e considerações finais.

2 CONCEITOS

Este capítulo tem como objetivo definir o que são sistemas de informações e sistemas WEB. A seção 2.1 relata sobre sistemas de informação e as suas classificações. A seção 2.2 descreve sobre sistemas WEB e explica as diferenças com sites WEB convencionais.

2.1. SISTEMAS DE INFORMAÇÃO

Atualmente a informática é parte importante da sociedade e cultura, estando presente em diversos ramos de atividade humana.

Todo sistema que manipula e gera informação pode ser considerado genericamente como um sistema de informação, e é difícil conceber um sistema que não gere alguma informação, independe do seu nível de aplicação, uso e tipo. As informações atualmente encontram-se em grande volume disponíveis em vários meios de comunicação, exigindo de todos o conhecimento para seleção e organização para a sua efetiva utilização. (REZENDE, 2002)

Os sistemas de informação são importantes para as empresas, pois podem contribuir para a solução de diversos problemas, porém é necessário o correto conhecimento das pessoas que o utilizam para que haja correto manuseio e organização.

Diante deste conceito, reforça a importância da abordagem sociotécnica no desenvolvimento do software. Um sistema de informação pode ser definido como o processo de transformação de dados em informações que serão utilizados na estrutura decisória da empresa e que proporciona a sustentação administrativa visando a otimização dos resultados esperados, sendo necessária para o bom funcionamento do sistema de informação a analise de todos estes fatores para um bom resultado.

Segundo Rezende (2002), os sistemas de informação podem se enquadrar em diferentes classificações, porém não existe uma distinção rígida de sua classificação, mas é principalmente ligada a aos níveis hierárquicos dentro da organização, sendo:

- Sistemas de informação operacional (SIO) - Controla o processamento dos dados na organização para a tomada de decisão.

- Sistema de informação gerencial (SIG) - Trabalha com dados gerenciais de determinado setor para auxiliar na tomada de decisão da mesma.

- Sistema de informação estratégica - Trabalha com uma visão macro da organização, levando em conta o ambiente interno e externo da organização e transformando em informação estratégica para a empresa.

Os modelos de sistemas de informação são três:

- Modelo convencional de sistemas de informação:

É o modelo clássico de gerenciamento que engloba todos os três níveis de sistema de informação, cada um bem distinto entre si.

- Modelo dinâmico de sistemas de informação:

Este modelo se aplica a empresas de maior parte pois as tomadas de decisões são mais complexas e há necessidade englobar todos os setores da empresa derrubando os níveis de sistemas de informação.

- Modelo de sistemas de informação com tecnologia da informação:

As organizações tem a possibilidade de utilizar varias ferramentas em seus diferentes níveis hierárquicos, desde o inicio da produção até o produto final, neste conceito se aplicam os ERP (*Enterprise Resource Planning*), *data ware house*, SGBD, inteligência artificial, *data minning*, sistemas especialistas.

Estes recursos são empregados de maneiras a tornar os processos de tomada de decisão mais eficiente e a organização em geral.

As mudanças na forma de trabalho, gerenciamento e ate mesmo entretenimento proporcionado pela informática mostra que estamos vivendo um novo modelo de organização social: a cibercultura.

2.2. SISTEMAS WEB

Sistemas WEB são aplicações hospedadas em servidores *online* que podem ser acessadas de qualquer lugar a qualquer momento através da web. (REZENDE, 2002)

Trata-se de um conjunto de programas que são executados em um servidor HTTP. O desenvolvimento da tecnologia web está relacionado, entre outros fatores, à necessidade de simplificar a atualização e manutenção do código, mantendo-o em um mesmo local, podendo ser acessado por diferentes usuários (BARBOSA, 2007).

Em outras palavras, a utilização da web como ambiente de execução permite que esse tipo de aplicação disponibilize diversos recursos aos usuários. O acesso direto a documentos e informações, assim como a consulta e edição dos dados publicados em vários computadores que formam a internet são alguns exemplos de como o usuário pode interagir com o sistema.

A arquitetura básica de um sítio web pode ser observada na figura 1. O servidor web aguarda a requisição do cliente, processa o pedido e acessa o banco de dados retornando ao cliente os dados solicitados.

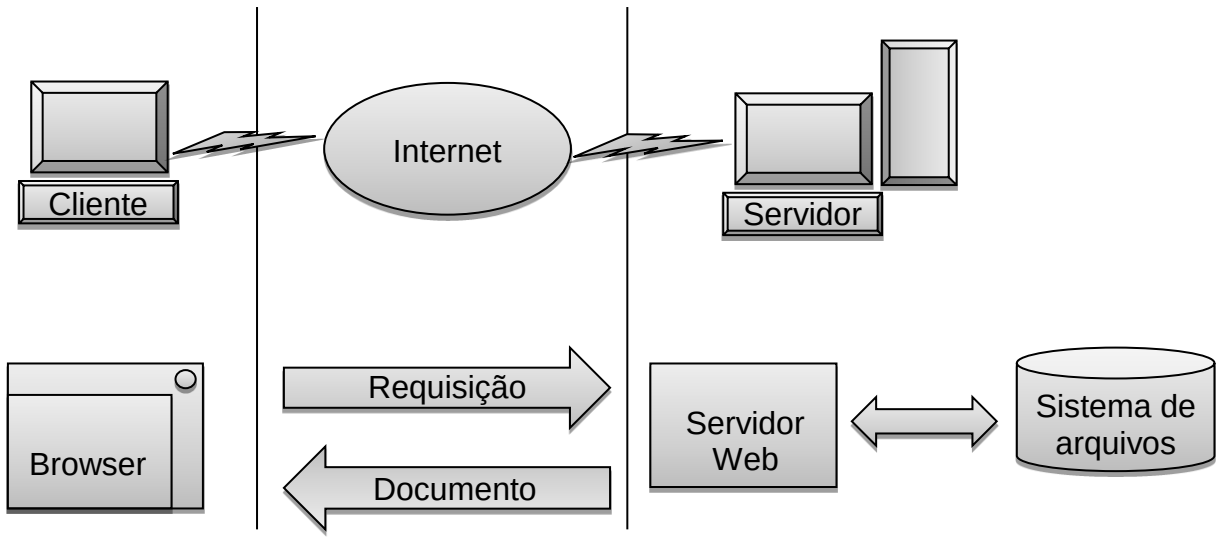


Figura 1 - Arquitetura básica de Site Web
Fonte: Adaptado de REZENDE (2002)

A aplicação Web difere de um sitio Web ao permitir que os usuários executem lógica de negócio com um navegador web. Com isso, pode-se dizer que a aplicação web permite a construção dinâmica de páginas que manipulam dados acessados na persistência através de um serviço Web. Enquanto os sítios web mostram como conteúdo um arquivo de documento pré-formatado, os aplicativos web constroem dinamicamente o conteúdo dependendo da interação do usuário com as páginas através do navegador (PIMENTA, 2002).

A figura 2 mostra a arquitetura básica de um sistema web dinâmico.

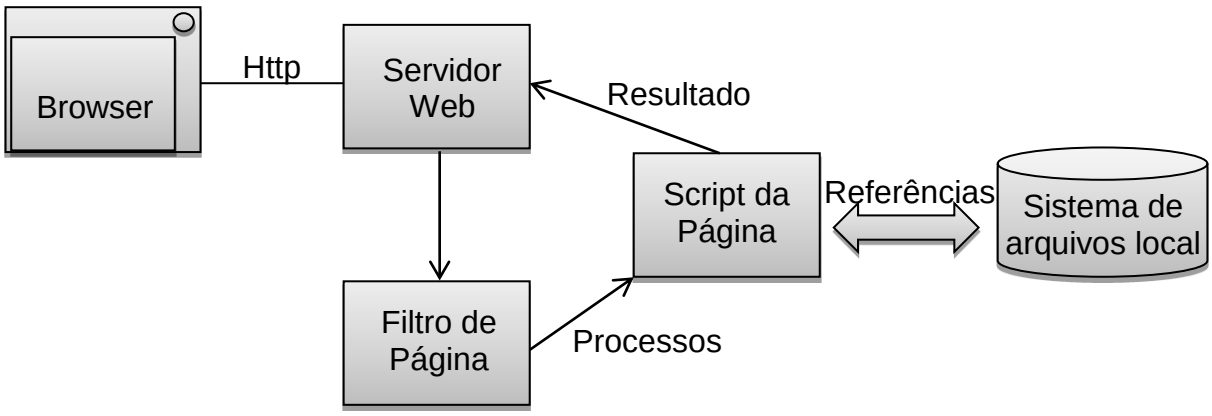


Figura 2 – Arquitetura básica de Site Web Dinâmico
Fonte: Adaptado de REZENDE (2002)

3 TECNOLOGIAS

Este capítulo demonstra as tecnologias estudadas e utilizadas para o desenvolvimento do software proposto. A seção 3.1 descreve a tecnologia HIBERNATE para o gerenciamento dos dados da aplicação proposta. A seção 3.2 relata sobre a tecnologia JPA que foi utilizada o conceito de entidade, que relaciona as classes da aplicação com as tabelas existentes no banco de dados. A seção 3.3 descreve a tecnologia Ajax utilizada na interface do software para apresentar os dados sem a necessidade de recarregar toda a pagina novamente. A seção 3.4 descreve o *framework* ICE Faces utilizadas para integrar o JSF com outra

tecnologias como o Ajax. A seção 3.5 descreve a tecnologia JSF utilizada nas paginas do aplicativo proposto.

3.1. HIBERNATE

Quase todos os aplicativos desenvolvidos atualmente dependem do gerenciamento de dados persistentes. Devido à familiaridade que os desenvolvedores têm com a linguagem SQL, tabelas e chaves estrangeiras, acabam por escolher a tecnologia JDBC e SQL (BAUER; KING, 2005, p.).

Porém, a tecnologia JDBC requer um considerável trabalho por parte do programador, pois necessita codificar manualmente a persistência para cada classe de domínio, afirmam Bauer e King (2005).

Segundo Gonçalves (2007), Hibernate é um projeto que procura solucionar completamente o problema de gerenciamento de dados persistentes em Java.

Hibernate é um *framework* que se relaciona com o banco de dados, onde esse relacionamento é conhecido como mapeamento objeto/relacional para Java, deixando o desenvolvedor livre para se concentrar em problemas da lógica de negócio. Sua simplicidade em configuração, dá ao desenvolvedor algumas regras para que sejam seguidas como padrões de desenvolvimento ao escrever sua lógica de negócios e suas classes persistentes. (GONÇALVES, 2007, p. 512).

O autor ainda afirma que “o Hibernate se integra suavemente ao seu sistema se comunicando com o banco de dados como se fosse diretamente feito por sua aplicação”. Como benefício, qualquer alteração que seja feita no banco de dados da aplicação não irá refletir em grandes mudanças no código do sistema. Assim, serão necessárias apenas algumas alterações nas configurações do Hibernate. (GONÇALVES, 2007, p. 512).

Antes de começarmos a nos aprofundar nas funcionalidades do Hibernate, devemos primeiramente entender as interfaces de programação. Segundo Bauer e

King (2005), essas interfaces (API – Interface de Programação de Aplicativos) é que serão usadas dentro da camada de persistência do aplicativo a ser desenvolvido.

“O objetivo principal do projeto da API é manter as interfaces entre os componentes do software tão estreitamente quanto possível.” (BAUER; KING, 2005, p. 50).

3.2. JPA

A API de persistência do Java faz todo o mapeamento de forma simples e transparente para o desenvolvedor, com isso pode se focar apenas na estrutura das entidades envolvidas no projeto. (SAMPAIO, 2011).

O JPA trabalha com o conceito de entidades, segundo Cleuton (2011, p. 207), “uma *Entity* é um objeto persistente de um banco de dados.” Ele ainda afirma que a entidade não equivale a um registro do banco, mas sim, que “ela pode ser composta por registros de tabelas diferentes.” (SAMPAIO, 2011, p. 207).

Segundo Cleuton Sampaio, para cada elemento que se deseja persistir em uma aplicação, uma classe entidade deve ser criada.

3.3. Ajax

Ajax (*Asynchronous JavaScript and XML*) não é uma tecnologia é sim uma técnica utilizada para programação WEB que faz uso de tecnologias JavaScript gerenciando a comunicação cliente servidor, e XML encapsulando a informação (LIMEIRA, 2006).

O Ajax surgiu para resolver o problema de atualização de paginas, a cada requisição toda a pagina era recarregada.

Com o Ajax pode-se trafegar apenas os dados que forem utilizados na pagina, deixando de maneira estática os outros dados da pagina.

A técnica Ajax faz uso das seguintes tecnologias:

- Apresentação baseada em padrões, que utilizam XHTML e CSS.
- Exibição e interação dinâmicas por meio de DOM *(DOCUMENT OBJECT MODEL*)
- Troca e manipulação de dados por meio do uso de XML e XLST.
- Recuperação assíncrona de dados com *XMLHttpRequest*.

A implementação baseada na classe *XMLHttpRequest* é a mais utilizada, faz o meio campo entre servidor e o navegador, bastante utilizada em serviços de *webmail* e paginas de noticias dinâmicas (LIMEIRA, 2006).

3.3.1. Vantagens

A grande vantagem do uso de Ajax para o desenvolvimento de um site é permitir alterações com menos trocas de dados entre cliente e servidor, pois permite a troca de trechos pequenos da pagina tornando a atualização de conteúdo muito rápida, dando a impressão para o usuário que a aplicação esta em seu próprio computador.

Menos uso de banda, como menos informação é trafegada , o uso de banda acaba sendo diminuído drasticamente (LIMEIRA, 2006).

3.3.2. Desvantagens

O maior problema se encontra no fato da grande quantidade de funcionalidades serem implementadas em Java script torna difícil a manutenção de funções que os usuários estão acostumados, como a opção voltar do navegador.

Outra desvantagem se da em redes com tempo de respostas muito grande, o utilizador acaba não percebendo que a pagina não esta mais respondendo e deve ser alertado visualmente (LIMEIRA, 2006).

Em uma pagina de estrutura tradicional o fluxo de informações o servidor envia para o navegador uma nova pagina.

A figura 3 demonstra a estrutura de uma aplicação WEB tradicional.

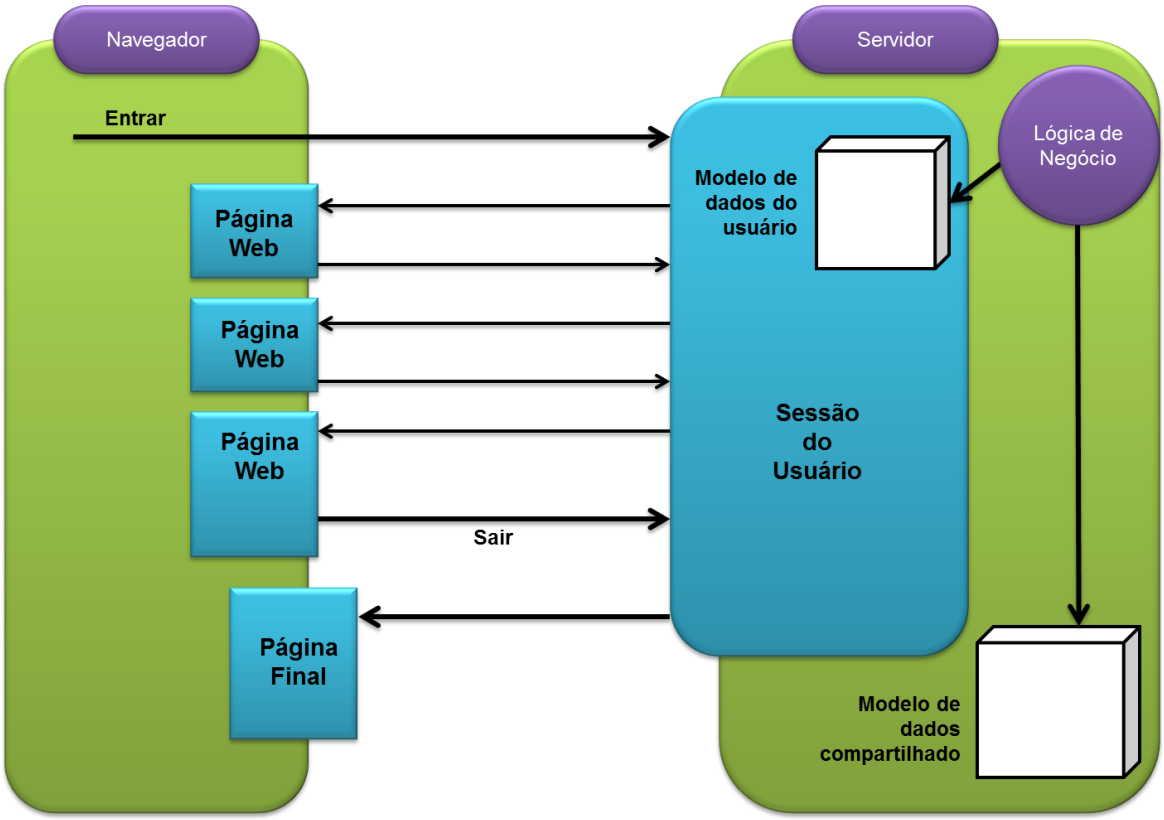


Figura 3 - Aplicação WEB tradicional
Fonte: Adaptado de SOUZA (2006)

Com o uso da técnica Ajax e as suas tecnologias envolvidas, as requisições e atualizações de tela tem um fluxo de informações diferente.

Utilizando o Ajax navegador interage com o servidor passando apenas as informações necessárias quando são necessárias.

A figura 4 demonstra uma aplicação WEB utilizando Ajax.

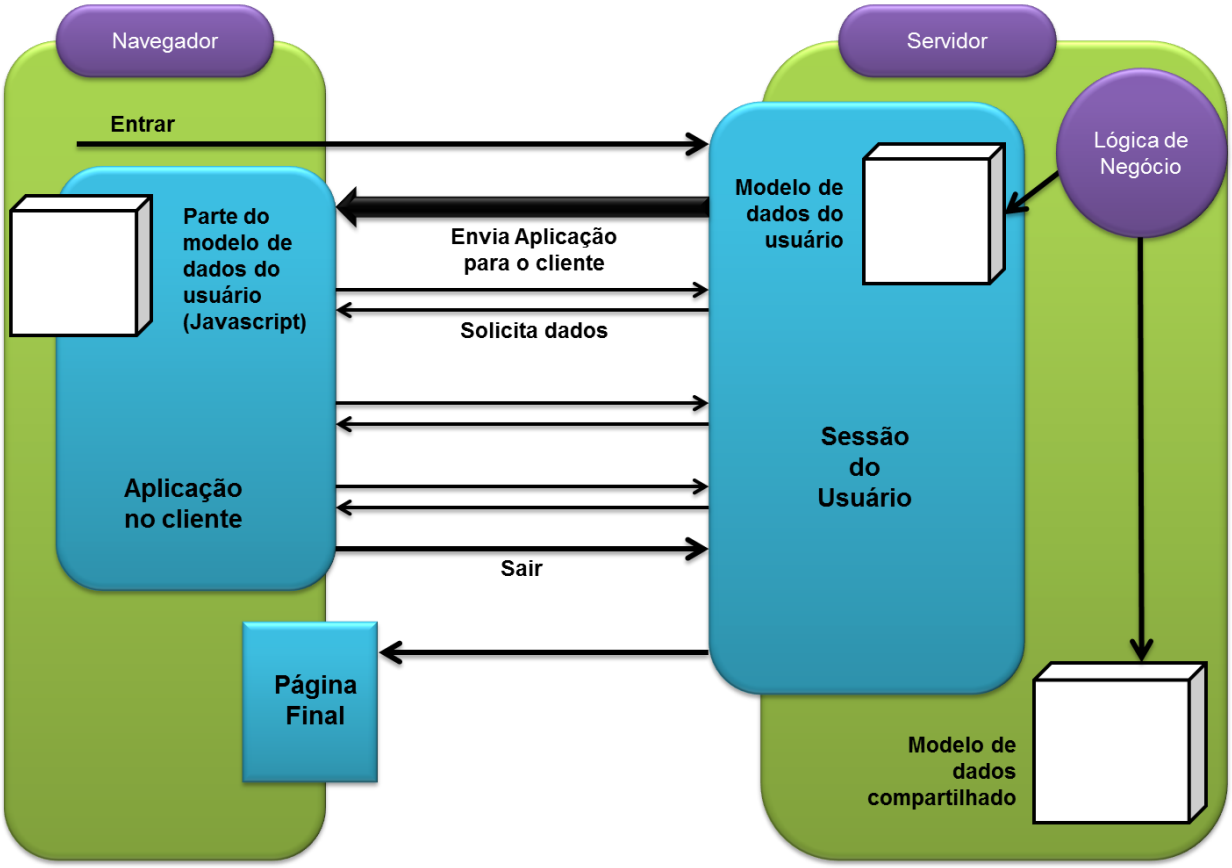


Figura 4 - Aplicação WEB com Ajax
Fonte: Adaptado de SOUZA (2006)

O modelo tradicional a interface interage diretamente com o servidor enviando e solicitando a pagina inteira, já no modelo Ajax, a interface interage com o mecanismo Ajax que intermedia as solicitações entre o cliente e servidor, apenas passando informações necessárias para ambos os lados, permitindo assim que a pagina carregada pelo usuário ainda possa ser manipulada pelo mesmo.

A figura 5 faz a comparação entre os modelos e os módulos envolvidos entre uma página utilizando o modelo clássico e outra utilizando o modelo com Ajax.

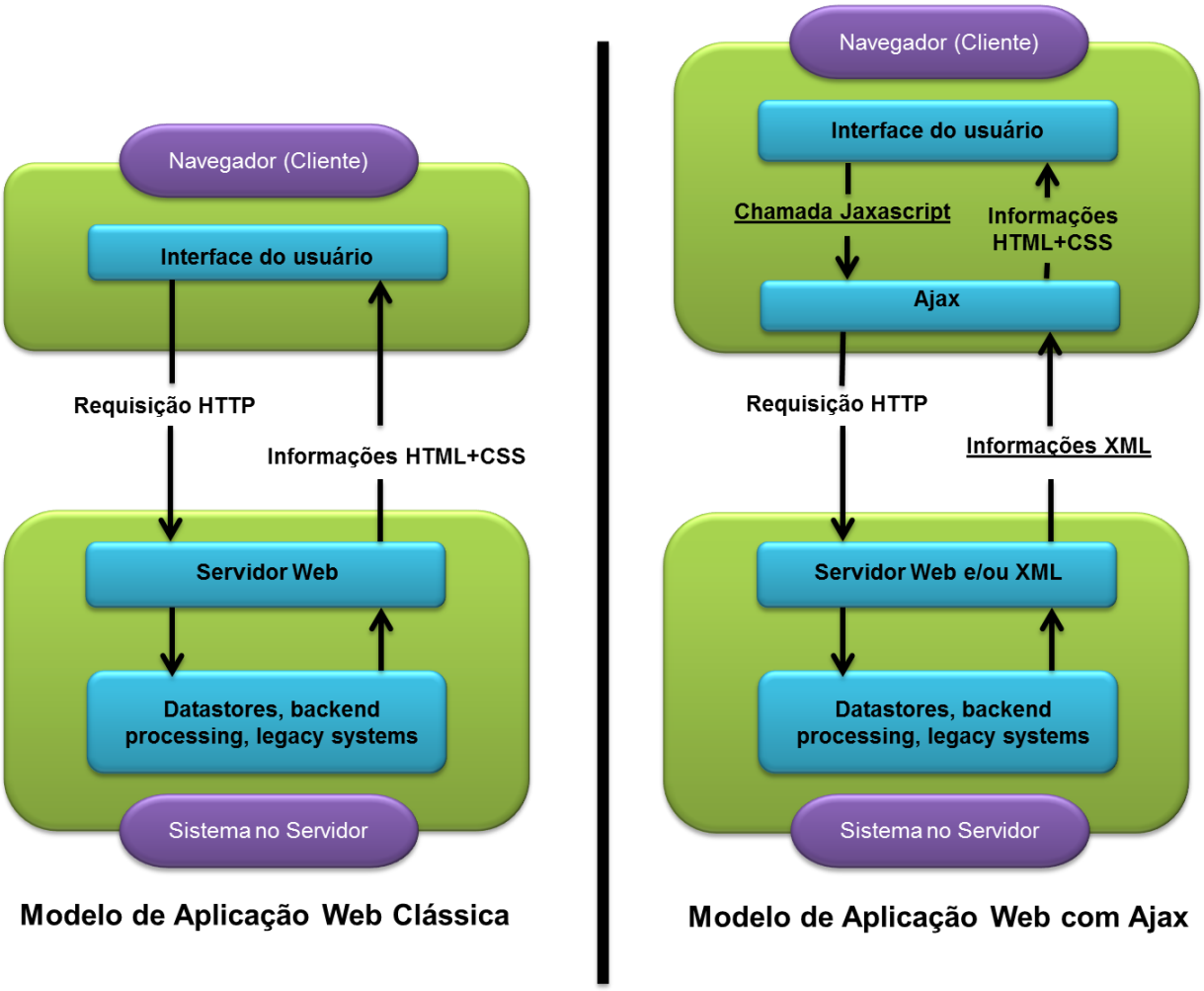


Figura 5 - Comparação entre aplicação WEB clássica e utilizando Ajax
Fonte: Adaptado de SOUZA (2006)

3.4. ICE FACES

O ICE Faces é um conjunto de componentes desenvolvidos pela ICE Soft, que tem como objetivo integrar as tecnologias JSF e Ajax de maneira nativa. Todos os componentes do ICE Faces são tecnologias JSF com suporte ao Ajax. (ICEFACES, 2011). A empresa desenvolvedora dessa tecnologia é a Rich Web Company e tem como principal objetivo “desenvolver soluções para tornar as aplicações Web mais ricas em funcionalidades”. (MARAFON, 2006)

Segundo Marafon, a empresa desenvolvedora do ICE Faces disponibilizou primeiramente o Framework em duas versões: ICE Faces Community Edition,

(versão gratuita) e *ICE Faces Enterprise Edition* (versão que poderia ser usada comercialmente).

Porém, em 2006, a empresa *Rich Web Company* optou por liberar o código do ICE faces e passou a cobrar apenas pelo suporte técnico, afirma Marafon.

As vantagens deste projeto se tornar open source já poderão ser observadas pelos usuários do framework Java Server Faces, tendo em vista que algumas ideias e alguns membros do ICE Faces irão participar do desenvolvimento da especificação JSF2.0, na qual se pretende fazer a integração com o Ajax. (MARAFON, 2006, p. 72).

A figura 6 descreve o fluxo de chamadas aos componentes do ICE Faces.

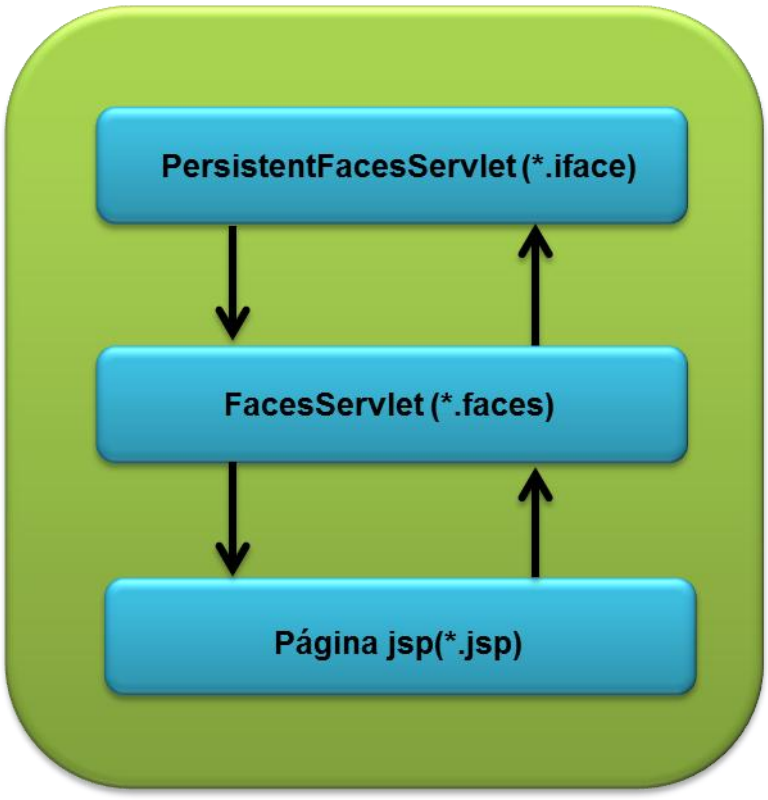


Figura 6 - Fluxo de chamadas do ICE Faces.
Fonte: Adaptado de ICEFACES (2011)

3.5. JSF

Java Server Faces é um *framework* fruto de um grande investimento da SUN MICROSYSTEM, e é atualmente muito utilizado na internet, porém uma de suas características é a utilização de requisições síncronas, exigindo que a página seja construída novamente a cada nova requisição, o uso de tecnologias com o Ajax eliminam esse problema. (Integração Java Server Faces e Ajax: Estudo de integração entre tecnologias JSF e Ajax. Revista Abstração, Santa Catarina: UFSC, mai. 2009. Edição 2.)

JSF é o *framework* oficial da especificação Java EE, e é desenhado para o desenvolvimento de aplicações WEB baseadas em componentes.

Essa tecnologia ainda tem como base o modelo de programação Cliente/Servidor (COULOURIS; DOLLIMORE; KINDBERG, 1994; apud MARAFON, 2006). Os servidores geralmente são as máquinas mais poderosas que atendem às requisições dos computadores clientes. Os servidores, devido à sua velocidade, também são responsáveis por processar os dados e fazer cálculos mais pesados que possam levar muito tempo se feitos nos computadores cliente, afirma Marafon.

Uma grande vantagem deste *framework* é a sua programação ser orientada a eventos, e possui uma grande conjunto de componentes prontos e padronizados.

Framework em linhas gerais é um conjunto de bibliotecas com praticas gerais encapsuladas que juntas criam uma forma mais simples de desenvolver alguma tarefa.

Uma característica deste *framework* é conhecida com ciclo de vida das requisições, o qual possui seis fases, sendo possível acrescentar novas etapas.

O JSF implementa o padrão de projeto MVC (*Model View Control*).

O controle é feito através de um servlet chamado Faces Servlet, por arquivos XML de configuração e por manipuladores, de ações e observadores de eventos, este servlet recebe as requisições dos usuários na WEB, redireciona para o modelo e manda uma resposta. As informações sobre mapeamentos de ações e regras de navegação estão nos arquivos de configuração. Os manipuladores de eventos são responsáveis pelo recebimento de dados da camada *VIEW*, acessar o

MODEL e enviar o resultado para o usuário através do Faces Servlet, (BAUER, 2007)

4 DESENVOLVIMENTO

Este capítulo apresenta como foram aplicadas as tecnologias estudadas e a arquitetura utilizada. A seção 4.1 descreve os módulos que foram desenvolvidos e as suas funções. A seção 4.2 descreve o modelo MVC utilizado para a construção do software. A seção 4.3 demonstra a aplicação do *framework* Hibernate no desenvolvimento da base de dados do aplicativo proposto.

4.1. DESCRIÇÃO DO SISTEMA

Este sistema possui recursos, que facilitam o manuseio de informações, de maneira que as mesmas sejam tratadas de forma mais fácil, rápida e segura, sendo assim, elas são armazenadas em um local de confiança, fazendo com que estejam disponíveis a qualquer horário, sem interrupções.

Atende o problema de arquivamento de dados, que antes era feito manualmente, e ocasionalmente transferidos para uma simples planilha, o que gerava tumulto e dificuldade de acesso.

O Sistema compreende módulos de gerenciamento referente à persistência de informações (módulos Professor e Secretaria) e módulos de pesquisa e agendamento (módulo Aluno).

4.1.1. Módulos de Gerenciamento:

Compreendem em possibilitar o registro e gerenciamento de informações referentes a professores responsáveis por lecionar, funcionários designados a ter acesso às essas alterações, alunos matriculados, cursos disponibilizados e seus respectivos valores, pagamentos de mensalidades, entre outras informações que

satisfazem a necessidade do contratante do sistema, como registro de aulas e horários.

4.1.2. Módulos de Consulta e Agendamento:

O Sistema permite que o aluno cadastrado, possua ter um controle de suas aulas, podendo verificar data e horário das mesmas e, conforme disponibilidade de novos horários, também pode solicitar a mudança de uma determinada aula para uma nova data e horário, desde que estes estejam disponíveis tanto pelo seu respectivo professor e por uma sala de aula vaga no horário estipulado. O próprio sistema se encarrega de fazer essa verificação de disponibilidade, garantindo as opções para escolha feitas pelo requerente.

4.2. ARQUITETURA E IMPLEMENTAÇÃO

A figura apresenta a arquitetura do aplicativo, apresentando o modelo MVC (*MODEL VIEW CONTROLLER*), as classes de dados, fluxo de objetos, usuários e o processo de desenvolvimento. O MVC garante a separação de tarefas facilitando a manutenção do código.

A figura 7 descreve a arquitetura MVC utilizada para o desenvolvimento do projeto.

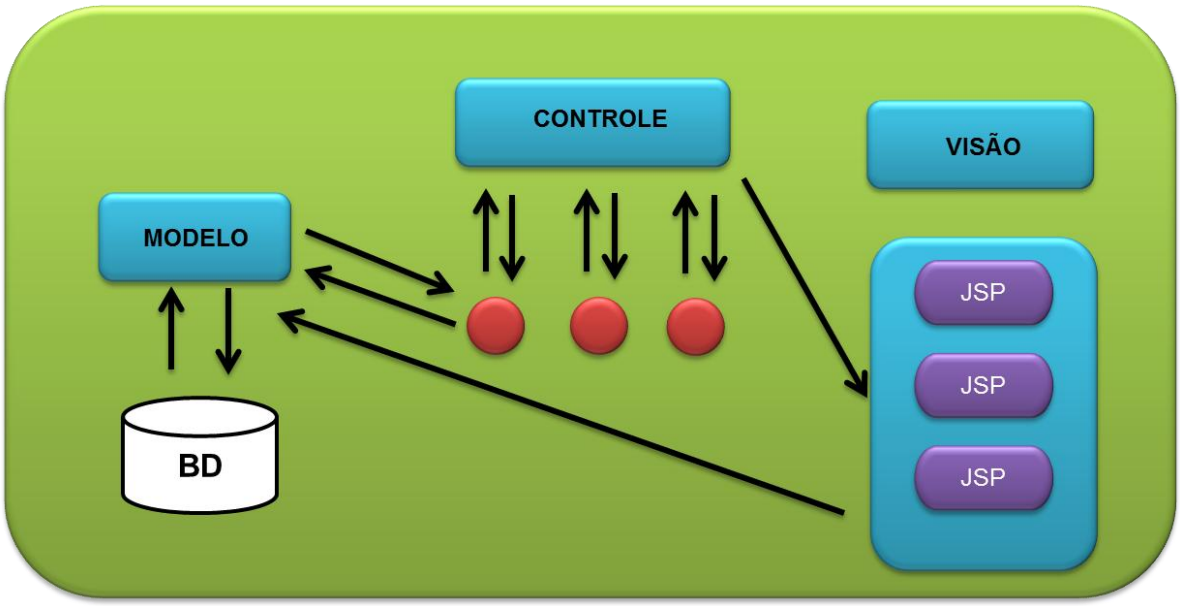


Figura 7 - Arquitetura MVC
Fonte: Adaptado de BAUER (2007)

A camada VISÃO, responsável pela apresentação das telas foi utilizado JSP (*Java Server Pages*) utilizando alguns recursos e metodologias como o Ajax e o *framework ICE Faces.*

4.3. PERSISTÊNCIA DE DADOS

Como descrito anteriormente, o Hibernate facilita o desenvolvimento da aplicação, já que este, por sua vez, se encarrega de gerenciar a ligação entre a aplicação e a persistência de dados através do mecanismo de JPA.

O uso desse framework na aplicação desenvolvida possibilita descrever os benefícios que o mesmo trouxe para os desenvolvedores, sejam eles no quesito tempo, complexidade e número de linhas de código, e, por fim, praticidade de gerenciamento dos dados da persistência.

O *framework* torna o desenvolvimento mais ágil se comparado à tecnologia JDBC. O Hibernate mapeia a base de dados de forma que esta possa ser manipulada como se fizesse parte diretamente da aplicação. Ele diminui o número de instruções SQL que habitualmente seriam necessárias e simplifica a escrita do

código a ser desenvolvido. Em testes realizados, verificou-se que o desempenho da aplicação não apresentou melhorias com o uso do Hibernate comparado a outras ferramentas e tecnologias para persistência, como por exemplo, JDBC.

As classes são mapeadas e relacionadas com as tabelas. A partir de então se começa a trabalhar com o conceito de JPA, ou seja, o conceito de entidades. Essas entidades, basicamente, são classes Java. O que as difere de classes Java comuns, são as anotações que o Hibernate gera automaticamente para cada atributo da classe. Através dessas anotações, o Hibernate faz a comunicação entre a aplicação e o gerenciador de banco de dados relacionando os atributos com as colunas da tabela mapeada.

Por trás desse mapeamento, o Hibernate gerencia a persistência desses dados. Ao invés de criar códigos SQL para cada tipo de requisição no banco de dados, o Hibernate relaciona cada tipo de objeto com sua respectiva tabela na base. Com isso, o objeto é persistido sem a necessidade de referenciar manualmente cada um de seus atributos com as colunas das tabelas do banco.

Algumas consultas básicas também podem ser feitas através das anotações criadas pelo Hibernate. Para isso, *query's* padronizadas são disponibilizadas dentro das classes de objetos, permitindo que extensos comandos SQL que seriam usados para fazer as consultas sejam traduzidos em pequenas *String's*. No momento da consulta, essas *String's* são passadas por parâmetros para determinados métodos internos das classes do Hibernate. Esses métodos, por sua vez, se encarregam de trazer automaticamente o resultado do comando SQL relacionado com a *String* enviada. Dessa maneira, as consultas podem ser feitas de forma simplificada sem precisar concatenar *String's*.

As tabelas relacionadas entre si, com chave estrangeira, também são reconhecidas pelo Hibernate. Ele trata essas ligações de forma interessante. Ao criar o banco de dados, definimos as chaves primárias das tabelas, esses campos geralmente recebem valores do tipo inteiro. Consequentemente, as colunas das outras tabelas que recebem a chave estrangeira também terão este campo com um valor do tipo inteiro.

No banco de dados, as tabelas ficam relacionadas desta forma convencional, porém, quando criamos as classes modelo da aplicação de forma automática,

através do Hibernate, ele cria as classes com os mesmos nomes e atributos do mesmo tipo das tabelas do banco. No entanto, quando há um campo com chave estrangeira que referencia outra tabela mapeada pelo Hibernate, na classe Java não é criado o atributo do mesmo tipo que o campo contido na tabela. Para facilitar a referência das chaves estrangeiras, o campo criado na classe gerada pelo Hibernate armazena um objeto do tipo da classe que representa a tabela referenciada.

Como exemplo, podemos utilizar o registro de uma matrícula acadêmica. A classe Matricula teria como chave estrangeira o código do aluno. Na classe Java que representa a Matricula, teremos os atributos referentes aos campos da tabela do banco de dados e a chave estrangeira será um objeto do tipo Aluno com todos os seus atributos. Ao realizar uma matrícula, será selecionado o aluno que deve ter sido previamente cadastrado e preenchido o restante dos atributos. Ao persistir este objeto do tipo matrícula, o Hibernate se encarrega de gravá-lo no banco de dados de forma que apenas o código do objeto aluno seja inserido como chave estrangeira na tabela Matricula.

No lugar dos códigos SQL que determinariam o tipo de procedimento a ser usado (no caso, inserção), são usados métodos pré-definidos nas classes internas do Hibernate. Esses métodos, graças ao relacionamento feito entre as tabelas e as entidades, permitem que seja passado como parâmetro o objeto a ser gravado no banco, sem que seja necessário fazer referência aos seus atributos individualmente.

O Hibernate se encarrega de fazer automaticamente a persistência desses dados sem que demais códigos sejam escritos, gerando maior praticidade e agilidade para a realização da tarefa estipulada.

O Código Fonte 1 mostra a classe utilizada para a conexão com banco de dados utilizando a técnica JDBC.

```
6   public class Conexao {
7
8       Connection conn = null;
9
10      public Conexao() {
11      }
12
13      public Connection conectar() throws Exception, ConnectException {
14          try {
15              Class.forName("org.firebirdsql.jdbc.FBDriver");
16              conn = DriverManager.getConnection("jdbc:firebirdsql:localhost/3050:"
17                      + "D:/BASE.FDB", "SYSDBA", "masterkey");
18          } catch (ClassNotFoundException e) {
19              System.out.println("Classe do firebird não encontrada."
20                      +"\n\nMenssagem Técnica:\n " + e.getMessage());
21          } catch (Exception e) {
22              throw new Exception("Erro desconhecido na de Conexão com a base."
23                      +"\n\nMenssagem Técnica:\n " + e.getMessage());
24          }
25          return conn;
26      }
27
28      public void desconectar() throws SQLException {
29          try {
30              conn.close();
31          } catch (SQLException e) {
32              throw new SQLException("ERRO: " + e.getMessage());
33          }
34      }
35
36      public Connection getConn() {
37          return conn;
38      }
39  }
```

Código Fonte 1 - Classe para conexão com banco de dados utilizando JDBC
Fonte: Autoria Própria

Como podemos ver, nas linhas 15, 16 e 17 do código, são definidos dados para conexão com o banco, como o driver, local da base de dados, usuário e senha para acesso.

O Código Fonte 2 demonstra o método inserirAluno de um cadastro utilizando JDBC.

Método inserirAluno - JDBC

```
18      public void inserirAluno(String nome, String endereco, String email) {
19          try {
20              Conexao cx = new Conexao();
21              cx.conectar();
22
23              if (cx.getConn() != null) {
24                  PreparedStatement ps = cx.getConn().prepareStatement("insert "
25                          + "into ALUNO(nome, endereco, email) values(?,?,?)");
26                  ps.setString(1, nome);
27                  ps.setString(2, endereco);
28                  ps.setString(3, email);
29                  ps.executeUpdate();
30
31              } else {
32                  System.out.println("erro");
33              }
34              cx.desconectar();
35          } catch (SQLException e) {
36              System.out.println("Erro na Consulta: " + e.toString());
37          } catch (Exception e) {
38              System.out.println("Driver não Encontrado: " + e.toString());
39          } finally {
40          }
41      }
```

Código Fonte 2 - Exemplo de inserção utilizando JDBC
Fonte: Autoria Própria

O método de inserção recebe os valores de cada campo da tabela individualmente (linha 18), cria *statement* e atribui o valor de cada campo, e os passa através de uma SQL gerada para executar a tarefa (linhas 24 - 29).

O Código Fonte 3 mostra a aplicação de um cadastro utilizando Hibernate.

Método cadastrarAlunoPers - Hibernate

```
15  public boolean cadastrarAlunoPers(Aluno aluno) throws Exception {
16      try {
17          EntityManagerFactory emf = Persistence.createEntityManagerFactory(
18                  "SistemaCoversPers");
19          EntityManager em = emf.createEntityManager();
20          em.getTransaction().begin();
21          em.persist(aluno);
22          em.getTransaction().commit();
23          em.close();
24          emf.close();
25          return true;
26      } catch (Exception e) {
27          throw new Exception(e.getMessage());
28      }
29  }
```

Código Fonte 3 - Exemplo inserção utilizando Hibernate
Fonte: Autoria Própria

O método ficou mais simples. O objeto é passado por parâmetro como um todo, sem precisar especificar seus atributos individualmente. O mesmo ocorre no comando para gravar o objeto no banco de dados (linha 21).

O Código Fonte 4 demonstra o método atualizarAluno utilizando JDBC.

Método atualizarAluno - JDBC

```
75      public void atualizarAluno(int codAluno, String nomeAluno,
76              String enderecoAluno, String emailAluno) throws Exception {
77          try {
78              Conexao cx = new Conexao();
79              Statement statement = null;
80              ResultSet rs;
81
82              cx.conectar();
83              System.out.println(cx.getConn());
84              if (cx.getConn() != null) {
85                  PreparedStatement ps = cx.getConn().prepareStatement("update"
86                          +"ALUNO set NOME = ?, ENDERECO = ?, EMAIL = ?"
87                          +"where COD = " + codAluno);
88
89                  ps.setString(1, nomeAluno);
90                  ps.setString(2, enderecoAluno);
91                  ps.setString(3, emailAluno);
92                  ps.executeUpdate();
93                 } else {
94                  System.out.println("erro");
95              }
96              cx.desconectar();
97
98          } catch (SQLException e) {
99              System.out.println("Erro na Consulta:" + e.toString());
00          } catch (Exception e) {
01              System.out.println("erro" + e.toString());
02          }
03      }
```

Código Fonte 4 - Exemplo atualização utilizando JDBC
Fonte: Autoria Própria

O uso de JDBC para atualização dos dados necessita que um valor para comparação de existência do item na tabela seja passado por parâmetro. Pra melhor funcionamento e não resultar em múltiplos valores, o atributo a ser comparado tem que ser de uma coluna do tipo *unique*, fazendo com que os dados atualizados sejam do item correto. Os valores novamente são atribuídos e criado *statement* junto com um código SQL (linhas 85 – 92).

O Código Fonte 5 mostra a aplicação de uma atualização de dados utilizando Hibernate.

Método alterarAlunoPers - Hibernate

```
31  public boolean alterarAlunoPers(Aluno aluno) throws Exception {
32      try {
33          Aluno aux = new Aluno();
34          EntityManagerFactory emf = Persistence.createEntityManagerFactory(
35                  "SistemaCoversPers");
36          EntityManager em = emf.createEntityManager();
37          em.getTransaction().begin();
38          aux = em.find(Aluno.class, aluno.getCod())
39          aux = aluno;
40          em.merge(aux);
41          em.getTransaction().commit();
42          em.close();
43          emf.close();
44          return true;
45      } catch (Exception e) {
```

Código Fonte 5 - Exemplo de atualização utilizando Hibernate
Fonte: Autoria Própria

Neste exemplo mostra-se que para alteração de dados, também é passado o objeto como um todo e o hibernate se encarrega de alterar o item correto de acordo com o ID do objeto. (linhas 38 - 40).

Foi desenvolvido um módulo para manipulação de dados do sistema utilizando JDBC especialmente para fazer essa comparação com relação ao uso do Hibernate.

A partir dessas imagens podemos facilmente reparar os primeiros benefícios que ele proporciona.

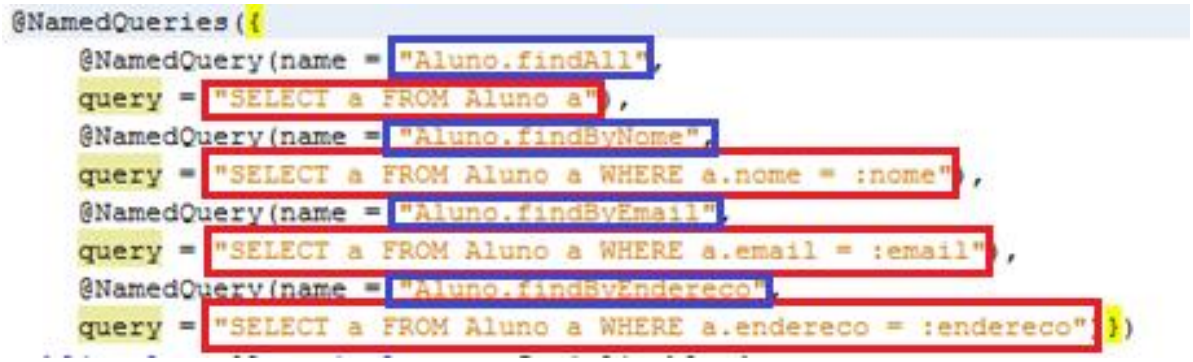
Percebemos que o número de linhas de código diminuiu consideravelmente. Usando JDBC, foi necessário duas classes e aproximadamente 170 linhas de código juntando ambas, enquanto pelo Hibernate, usamos apenas uma classe para um Crud completo e o mesmo resultou em quase metade do número de linhas de código usando JDBC. Apesar dessas mudanças, o desempenho da aplicação não foi otimizado.

O tratamento de dados entre as duas opções varia, podemos observar que pelo Hibernate, o código ficou mais “limpo” e simples, de fácil entendimento, dispensando a manipulação individual de cada tipo de dado para tratar sua persistência, já que utilizando Hibernate, a persistência trata o objeto como um todo, se encarregando do trabalho “braçal”, diferente do JDBC em que temos que construir toda a parte do código dessa persistência.

O que torna mais simples ainda, é o fato da diferença entre as opções do Crud pelo Hibernate em nível de código serem mínimas. Basicamente as invocações são as mesmas, apenas mudando a opção (salvar, atualizar, deletar ou selecionar um objeto).

A etapa de seleção também é muito interessante, ao invés de criar a SQL de busca, o próprio Hibernate gerou no mapeamento da base de dados com as classes da aplicação códigos nomeados para facilitar a busca, trazendo o objeto como um todo e ordenando de acordo com essa busca nomeada.

O Código Fonte 6 demonstra o código SQL que foi gerado utilizando o HIBERNATE.



Código Fonte 6 - Código SQL gerado e nomeado pelo Hibernate
Fonte: Autoria Própria

Destacado em azul, encontramos a *String* que representa o nome da *query* que poderá ser usada para realizar a busca. Em vermelho, temos o comando SQL referente à *query* nomeada.

4.4. DIAGRAMA DE MODELO ENTIDADE RELACIONAMENTO

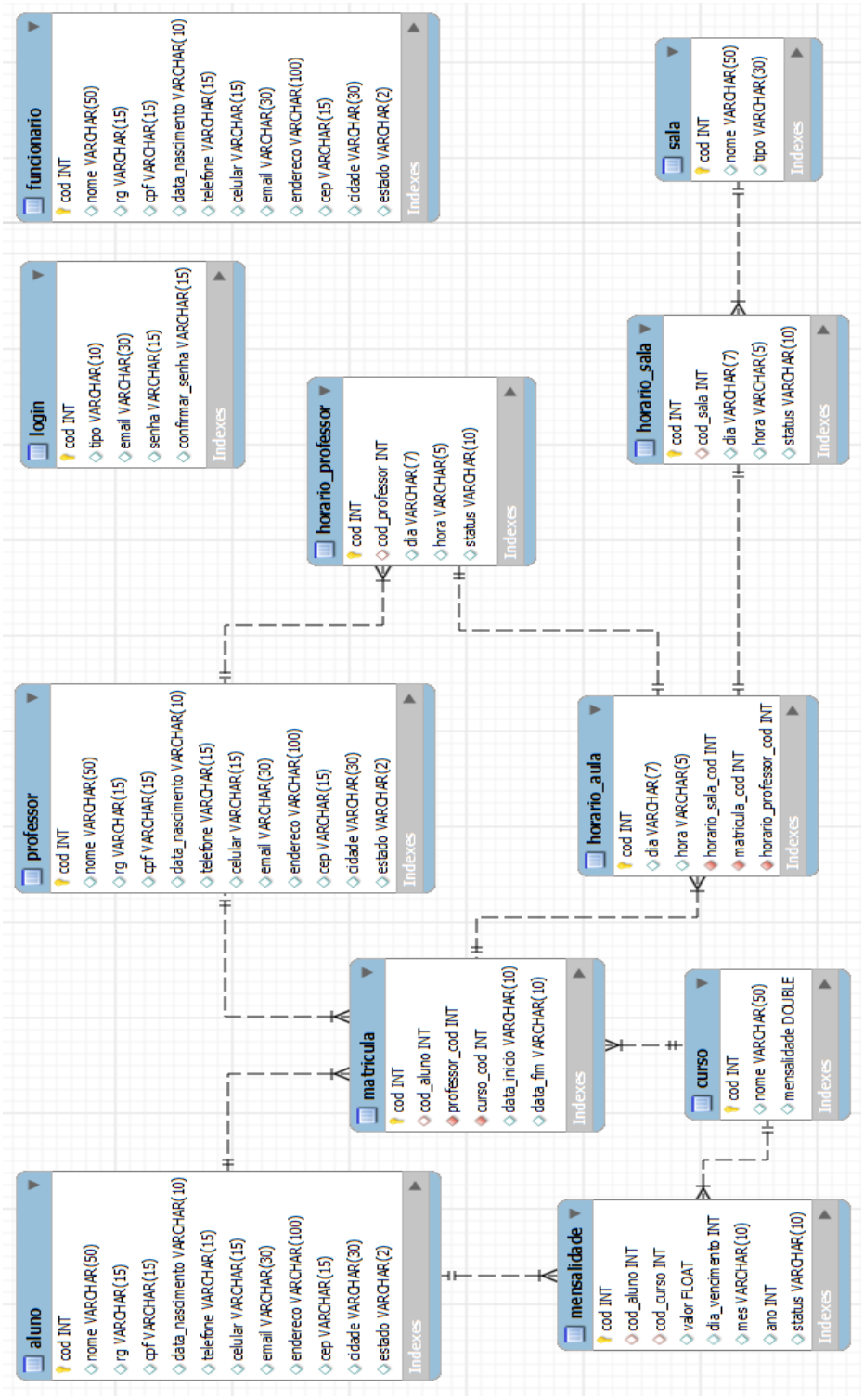


Figura 8 - Diagrama Modelo Entidade Relacionamento
Fonte: Autoria Própria

4.4.1. Pontos Críticos

4.4.1.1. Tabela login

Os dados dessa tabela serão usados unicamente para questão de sessões de usuário e segurança do sistema. Por esse motivo a mesma não está associada à nenhuma outra tabela, pois seria desnecessário. O campo "e-mail" representa o nome de usuário para acesso ao sistema.

Foi definido dessa maneira para facilitar o acesso por parte do usuário, dispensando a criação de um valor que possa futuramente ser esquecido.

4.4.1.2. Tabela matrícula

Primeiramente foi criada uma tabela chamada "Turma" a qual teria os campos "professor_cod" e "curso_cod", chaves estrangeiras das tabelas "professor" e "curso", respectivamente. Também teria o campo "cod_matricula", chave da tabela "matricula", que no caso possuía apenas o campo "cod_aluno" de chave estrangeira da tabela "aluno".

Porém, como existe apenas um aluno por turma, foi decidido colocar todos esses campos na tabela "matrícula" e eliminar a tabela "turma".

Esta decisão não fere a 1ª forma normal, pois nenhum dos campos da tabela "matrícula" são multivalorados.

Para cada matrícula, haverá apenas um aluno, um professor e um curso. Caso um aluno faça outro curso, será uma nova matrícula, ou seja, as matrículas não são por aluno, mas sim pelo conjunto de aluno, professor e curso.

4.5. DIAGRAMA DE CASO DE USO

A figura 9 mostra detalhes do funcionamento do sistema, terminando as tarefas disponíveis e quem serão os responsáveis de realizar as mesmas.

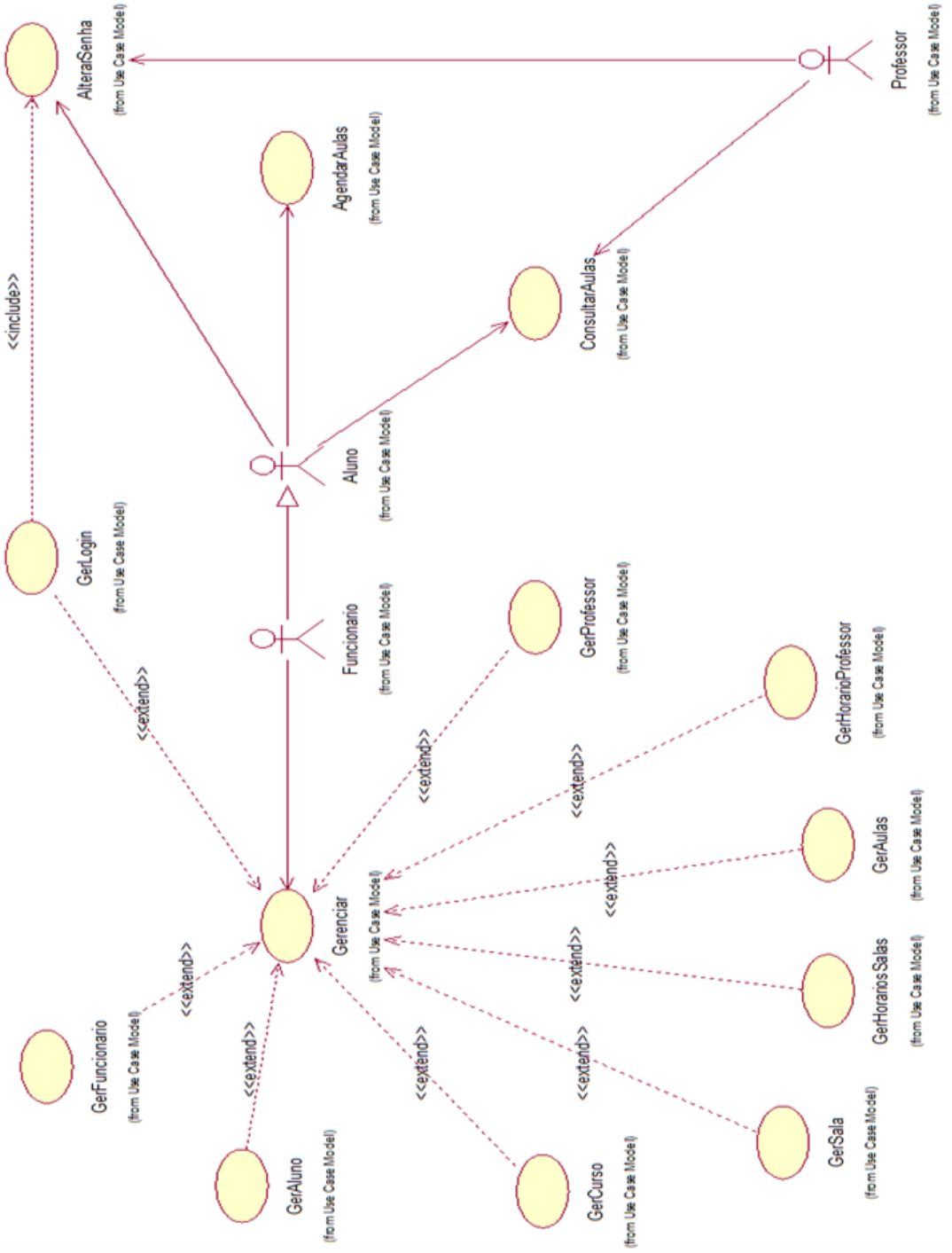


Figura 9 - Diagrama de Caso de Uso
Fonte: Autoria Própria

O Aluno terá direito a realizar tarefas mais básicas comparadas às tarefas de professor e funcionário, pois se trata apenas de objetivos ligados diretamente a ele e não aos demais envolvidos no uso do sistema. Entre essas tarefas estão:

- Consulta de Horários de Aulas: é onde o aluno poderá monitorar as datas e horários das aulas, além de também ter informações de qual o professor que lecionará e em que sala a aula será realizada;
- Agendamento de Aulas: o aluno poderá se achar conveniente mudar a data e horário de suas aulas, desde que seja para um dia e horário que seu professor esteja com horário disponível, o mesmo ocorre com relação à sala de aula, que deve estar disponível para que a aula seja agendada;
- Alterar Senha: o aluno pode também modificar sua senha de acesso ao sistema, porém não poderá fazer modificações no nome de usuário, pois este se trata do e-mail cadastrado desse aluno.

O Professor, mesmo fazendo parte da equipe de trabalho para qual o sistema foi desenvolvido, possui também acesso limitado, justamente para atender apenas o que está relacionado ao mesmo. O seu acesso compreende os mesmos acessos do Aluno, com exceção do agendamento de aulas, que não faz parte das tarefas determinadas ao professor. Este por sua vez pode:

- Consulta de Horários de Aulas: obter uma relação das aulas que este professor tem no decorrer do tempo estabelecido pela escola, em acordo foi estipulado cronograma de aulas semanais;
- Alterar Senha: mesmo critério do aluno, o professor que também tem um nome de usuário e senha no sistema para poder realizar acesso, pode então fazer alteração dessa senha quando julgar necessário.

Para finalizar a lista de usuários do sistema, resta o Funcionário. Este sim terá acesso superior e completo, diferenciando dos usuários Aluno e Professor. É o funcionário que será o responsável pelo gerenciamento de todas as informações do sistema.

As atividades que o Funcionário fará, que também Aluno e Professor fazem, diferem pelo fato de este por sua vez poder realizar as tarefas de todos os Alunos e funcionários, que podem apenas realizar as tarefas relacionadas a eles próprios. Casos de Agendamento de aulas, consulta de aulas e alterações de senha. Este

pode alterar uma senha de qualquer usuário, sem ser necessário colocar a senha antiga – um critério de segurança caso a senha antiga seja esquecida;

Demais atividades de gerenciamento de registros: De alunos, professores, cursos, salas, aulas e horários estão todos relacionados ao Funcionário, responsável por estas tarefas.

4.6. DESENVOLVIMENTO E APLICAÇÃO DAS TECNOLOGIAS.

A linguagem escolhida para o desenvolvimento foi a linguagem Java, pois é uma linguagem que permite uma enorme quantidade de tecnologias integradas a ela, portabilidade, possibilidade de integração dos componentes em um mesmo ambiente e padronização do desenvolvimento (SCHERER, 2010).

As ferramentas usadas para o desenvolvimento do projeto do Sistema Web de Gerenciamento da Escola de Música Cover's são ferramentas *open source*, ou seja, de códigos abertos e disponibilizados gratuitamente.

A principal ferramenta de desenvolvimento usada no projeto foi a IDE NetBeans nas versões 6.7 e 6.9. Os frameworks que auxiliaram no desenvolvimento foram o ICE faces Run-Time 1.8.2, Java Server Faces 1.1.02 e Hibernate 3.2.5.

De acordo com o funcionamento da escola, o Sistema foi desenvolvido para melhor se adaptar à rotina administrativa da mesma. Considerando as diversas atividades feitas na escola, pode-se citar entre as mais relevantes, a matrícula do aluno e o agendamento de aula. Essas tarefas, além de serem umas das mais importantes, também eram realizadas diariamente.

A seguir será descrito a atividade de matrícula de um aluno, tarefa realizada apenas pela secretaria da escola.

Com o uso da tecnologia ICE Faces, composta por componentes que integram JSF e Ajax, os dados injetados na aplicação podem ser manipulados e exibidos dinamicamente, como se o usuário estivesse interagindo com um Sistema local.

O componente *DataTable* é um dos componentes presentes nas bibliotecas do ICE faces e pode ser utilizado através da tag <ice:dataTable>. Muito similar ao componente de tabela de dados do JSF, a tabela do ICE faces tem o diferencial de já trabalhar interiormente com a tecnologia Ajax. (MARAFON, 2006)

A tabela de agendamento de aulas pode ser usada como exemplo para demonstrar o dinamismo que se ganha ao trabalhar com ICE faces.

Ao acessar a tela de horários do aluno, apenas o fragmento central de conteúdo da página é atualizado. O cabeçalho da página é mantido, contendo inclusive o nome do aluno logado, até que a sessão seja finalizada. A tabela é automaticamente preenchida com os dados referentes aos horários de aulas do aluno que está logado no sistema demonstrada na figura 10.

Figura 10 - Tela de visualização e edição de horário de Aluno
Fonte: Autoria Própria

A tabela do ICE Faces deve ser preenchida com um vetor de objetos, onde cada posição do vetor corresponde a uma coluna da tabela. Portanto, foi criado um

objeto do tipo *Object* composto por dois atributos, um String que contém o horário e um objeto do tipo HorarioAula (o qual é composto por informações como dia e horário da aula, matrícula, curso e horário do professor). Considerando que o horário de expediente da escola é de 14 horas diárias, o vetor de objetos é instanciado com 14 posições como demonstra o Código Fonte 7.

```
public class ObjetoTabela {

    private String hora;
    private HorarioAula domingo;
    private HorarioAula segunda;
    private HorarioAula terca;
    private HorarioAula quarta;
    private HorarioAula quinta;
    private HorarioAula sexta;
    private HorarioAula sabado;
```

Código Fonte 7 - Estrutura do objeto instanciado para preencher a tabela.
Fonte: Autoria Própria

Cada célula da tabela corresponde a uma variável do tipo HorarioAula. Assim que se obtém a lista de aulas de determinado aluno, é verificado o horário e o dia da aula e atribuído no vetor na posição respectiva ao horário e dia reservados no vetor.

A cada requisição feita pelos componentes do ICE faces, é enviada uma requisição ao servidor solicitando apenas os dados referentes àquele componente. Ao clicar em algum horário listado na tabela, os componentes de texto no painel acima são renderizados assumindo os valores referentes ao objeto que preenche a tabela demonstrado no Código Fonte 8.

```
public void selecionarHorario(RowSelectorEvent rse) {
    this.objTab = vetObjTab[rse.getRow()];
    this.horarioAula = new HorarioAula();
    if (objTab.getDomingo() != null) {
        this.horarioAula = objTab.getDomingo();
    } else if (objTab.getSegunda() != null){
        this.horarioAula = objTab.getSegunda();
    } else if (objTab.getTerca() != null){
        this.horarioAula = objTab.getTerca();
    } else if (objTab.getQuarta() != null){
        this.horarioAula = objTab.getQuarta();
    } else if (objTab.getQuinta() != null){
        this.horarioAula = objTab.getQuinta();
    } else if (objTab.getSexta() != null){
        this.horarioAula = objTab.getSexta();
    } else if (objTab.getSabado() != null){
        this.horarioAula = objTab.getSabado();
    }
    this.op = 0;
    this.setDesativarCampos(true);
    this.setHorarioAlunoBoolean(false);
}
```

Código Fonte 8 - Ação do clique em valores da tabela
Fonte: Autoria Própria

Basta preencher o objeto vetObjTab (referenciado no atributo "*value*" do formulário) com o objeto selecionado que os campos do formulário automaticamente assumem os valores desse objeto como mostra Código Fonte 9.

```
<ui:define name="content">

    <div id="h1">
        <ice:form >
            <ice:outputLabel value="Horários de Aula"/>
            <ice:panelGrid  columns="4" styleClass="painelGrid">
                <ice:outputLabel value="Curso: "/>
                <ice:outputText style="text-align: left;" value="#{beanHorarioAluno.horarioAula.matricula.curso.nome}"/>
                <ice:outputLabel value="Professor: "/>
                <ice:outputText style="text-align: left;" value="#{beanHorarioAluno.horarioAula.matricula.professor.nome}"/>
                <ice:outputLabel value="Dia:"/>
                <ice:selectOneMenu value="#{beanHorarioAluno.horarioAula.dia}"
                                   disabled="#{beanHorarioAluno.desativarCampos}">
                    <f:selectItems value="#{bean1.listaDiaSemana}"/>
                </ice:selectOneMenu>
                <ice:outputLabel value="Hora:"/>
                <ice:selectOneMenu value="#{beanHorarioAluno.horarioAula.hora}"
                                   disabled="#{beanHorarioAluno.desativarCampos}">
                    <f:selectItems value="#{bean1.listaHora}"/>
                </ice:selectOneMenu>
            </ice:panelGrid>
```

Código Fonte 9 - Preenchimento dos campos de edição – HorarioAluno.xhtml
Fonte: Autoria Própria

Na tela de cadastro dos horários dos professores fica bem explícita a forma dinâmica que o ICE faces faz as requisições. Na Figura 11 são mostrado todos os professores e seus horários

Figura 11 - Tela de Cadastro de Horários dos Professores
Fonte: Autoria Própria

O componente SelectOneMenu, que pode ser usado pela tag <ice:selectOneMenu> (semelhante ao *ComboBox*), traz uma lista de professores cadastrados no sistema. Num primeiro momento, este componente tem a função de selecionar o professor ao cadastrar um novo horário de professor. Porém, nesta tela ele também pode ser usado como filtro. Ao selecionar o professor no componente, a tabela é renderizada trazendo apenas os horários de aula do professor selecionado como demonstra a figura 12.

Figura 12 - Componente SelectOneMenu renderizando a tabela com o valor selecionado
Fonte: Autoria Própria

A mudança do valor no componente *selectOneMenu* aciona o método selecionarProfessor, e como parâmetro é passado um objeto do tipo *ValueChangeEvent*, o qual traz atributos como o valor antigo do componente (antes de sofrer a ação do usuário) e o novo valor atribuído a ele demonstrado no Código Fonte 10.

```
public void selecionarProfessor(ValueChangeEvent vcl) throws Exception {
    try {
        if (this.op == 0) {
            Bean1 b1 = new Bean1();
            this.horarioProfessor = new HorarioProfessor();
            Professor[] vetor = new Professor[b1.getVetorProfessores().length];
            vetor = b1.getVetorProfessores();
            int codProfSel = Integer.parseInt(String.valueOf(vcl.getNewValue()));
            HorarioProfessorRN hrn = new HorarioProfessorRN();
            if (codProfSel != 0) {
                for (int i = 0; i < vetor.length; i++) {
                    if (codProfSel == vetor[i].getCod()) {
                        this.horarioProfessor.setProfessor(vetor[i]);
                        break;
                    }
                }
                this.vetorHorarioProfessor = hrn.listarHorarioProfessorRN(this.horarioProfessor.getProfessor());
                this.codProfessor = this.horarioProfessor.getProfessor().getCod();
            } else {
                this.vetorHorarioProfessor = hrn.listarHorarioProfessorRN();
            }
        }
    } catch (Exception e) {
        throw new Exception(e.getMessage());
    }
}
```

Código Fonte 10 - Método acionado na mudança de valor do componente SelectOneMenu
Fonte: Autoria Própria

Na tela de *Login*, o usuário entra com seu *login* (e-mail) e sua senha cadastrados previamente no sistema. A autenticação do usuário é feita através do método validarSessao() chamado no *action* do botão “Entrar”. Na persistência, é verificado se existe o *login* associado com a senha informada passados por parâmetro. Se existir, o *login* é retornado para o *managed Bean*. Após isso, é verificado o tipo do usuário e uma busca é realizada na tabela dos usuários. Ao ser encontrado, o usuário é inserido na sessão e é retornado a String contendo o tipo do usuário. Nas regras de navegação do arquivo *faces-config* o direcionamento para a Home Page com as funcionalidades permitidas ao tipo do usuário. Caso o usuário não seja encontrado, a mensagem de erro é exibida informando que o usuário ou a senha estão incorretos.

Para garantir a segurança do sistema, impedindo acessos de usuários às páginas não permitidas, foi criada uma classe que implementa a classe *PhaseListener*. Essa classe monitora o que acontece antes e depois de cada fase do ciclo de vida do JSF. Neste caso, a fase monitorada foi a primeira fase do ciclo de vida JSF: *Restore View*.

Segundo Conceição (2008), no JSF, é na *View* que todos os componentes de uma página são representados e a principal função desta fase é dar início ao

ciclo de vida da requisição. Se for a primeira vez que o usuário visita a página, uma nova árvore de componentes é formada. Se a página já tiver sido visitada anteriormente, a árvore já existente é recuperada. (ANDRADE; FARIA, 2010).

Para isso, o *listener* da aplicação deve ser incluído no faces-config.xml.

```
<lifecycle>
    <phase-listener>ListenerLogin.CheckLogin</phase-listener>
</lifecycle>
```

Código Fonte 11 - Declaração do Listener Login
Fonte: Autoria Própria

O teste de validação do usuário é feito assim que a fase *RestoreView* é realizada, ou seja, o código de validação é inserido dentro do método afterPhase.

Primeiramente é verificado se a página que deu origem à requisição foi a página de *login* e se o usuário contido na sessão é nulo.

```
boolean isLoginPage = (currentPage.lastIndexOf("index.xhtml") > -1);
HttpSession session = (HttpSession) facesContext.getExternalContext().getSession(true);
Object usuario = session.getAttribute("usuario");
```

Código Fonte 12 - Verificação se a página atual é a página de login.
Fonte: Autoria Própria

Se o usuário não estiver logado e tentar acessar por URL uma página que não seja a página de *login*, é automaticamente redirecionado para a página de *login*. Porém, se o usuário estiver autenticado e tentar acessar a página de *login*, é automaticamente redirecionado para a homepage referente ao seu tipo de usuário.

```
if (usuario == null) {
    if (!isLoginPage) {
        //redireciona para a página de login
        NavigationHandler nh = facesContext.getApplication().getNavigationHandler();
        nh.handleNavigation(facesContext, null, "deslogado");
    }
} else if (isLoginPage) {
    // Se o usuário logado tentar acessar a página de login ele é
    // redirecionado para a página inicial
    if (usuario.getClass() == aluno.getClass()) {
        NavigationHandler nh = facesContext.getApplication().getNavigationHandler();
        nh.handleNavigation(facesContext, null, "Aluno");
    } else if (usuario.getClass() == professor.getClass()) {
        NavigationHandler nh = facesContext.getApplication().getNavigationHandler();
        nh.handleNavigation(facesContext, null, "Professor");
    } else if (usuario.getClass() == funcionario.getClass()) {
        NavigationHandler nh = facesContext.getApplication().getNavigationHandler();
        nh.handleNavigation(facesContext, null, "Secretaria");
    }
} else {
    session.setAttribute("usuario", usuario);
}
```

Código Fonte 13 - Validação do usuário através do PhaseListener
Fonte: Autoria Própria

Além disso, se o usuário autenticado tentar acessar uma página que não é permitida ao seu tipo de *login*. O usuário também é redirecionado para a página de *login*. Essa verificação é feita no método construtor do *Bean* referente ao cabeçalho da página de cada um dos três tipos de *login*, como pode ser visto no Código Fonte 14.

```
public BeanTemplateSecretaria() throws Throwable {
    session = (HttpSession) FacesContext.getCurrentInstance().getExternalContext().getSession(false);
    if (session.getAttribute("usuario").getClass() == funcionario.getClass()) {
        funcionario = (Funcionario) session.getAttribute("usuario");
    } else {
        BeanSessao bean = new BeanSessao();
        bean.logout();
        HttpServletResponse res = (HttpServletResponse) FacesContext.getCurrentInstance().getExternalContext().getResponse();
        res.sendRedirect("../index.iface");
    }
}
```

Código Fonte 14 - Verificação do tipo do usuário logado.
Fonte: Autoria Própria

As regras de navegação são declaradas no arquivo faces-config.xml.

```
<navigation-rule>
    <navigation-case>
        <from-outcome>deslogado</from-outcome>
        <to-view-id>/index.iface</to-view-id>
        <redirect/>
    </navigation-case>
</navigation-rule>
<navigation-rule>
    <from-view-id>/index.xhtml</from-view-id>
    <navigation-case>
        <from-outcome>Aluno</from-outcome>
        <to-view-id>/paginasAlunos/0HomeAlunos.iface</to-view-id>
        <redirect/>
    </navigation-case>
</navigation-rule>
<navigation-rule>
    <from-view-id>/index.xhtml</from-view-id>
    <navigation-case>
        <from-outcome>Professor</from-outcome>
        <to-view-id>/paginasProfessores/0HomeProfessor.iface</to-view-id>
        <redirect/>
    </navigation-case>
</navigation-rule>
<navigation-rule>
    <from-view-id>/index.xhtml</from-view-id>
    <navigation-case>
        <from-outcome>Secretaria</from-outcome>
        <to-view-id>/paginasSecretaria/0HomeSecretaria.iface</to-view-id>
        <redirect/>
    </navigation-case>
</navigation-rule>
```

Código Fonte 15 - Regras de navegação no arquivo faces-config.xml
Fonte: Autoria Própria

5 CONCLUSÃO

O principal objetivo deste trabalho é a pesquisa e o desenvolvimento de um sistema de gerenciamento Web fazendo uso de novas tecnologias proporcionando flexibilidade e estabilidade ao usuário.

A escolha da tecnologia a ser utilizada para a criação de um software afeta todo o seu desenvolvimento. No estudo realizado com o objetivo de minimizar os transtornos que, cotidianamente a empresa enfrenta em relação ao controle de dados dos alunos, puderam ser estudadas e aplicadas novas tecnologias disponíveis no mercado.

A utilização de tecnologias como *ICE faces*, *Hibernate*, *Java Persistence API*, *Java Server Faces* e Ajax, que dinamizam a interação do usuário com o sistema, permitiram que o desempenho da aplicação fosse satisfatório superando as expectativas dos usuários. A integração entre essas tecnologias contribuiu para que a implementação do projeto fosse feita de maneira ágil em todas as etapas do desenvolvimento.

O uso da linguagem Java proporcionou a integração com as demais tecnologias e, através da IDE NetBeans, puderam ser utilizados *frameworks* que auxiliaram durante todo o processo de implementação. Embora se tratando de um projeto de uma aplicação para ambiente Web, todas as funcionalidades requeridas pela Escola de Música puderam ser atendidas e dinamizadas para o melhor funcionamento da mesma.

Algumas pesquisas realizadas na área de sistemas web serviram como embasamento e foram comparadas entre si. Desta comparação, puderam-se conhecer as vantagens e limitações de ambas as tecnologias usadas no desenvolvimento da aplicação.

5.1. TRABALHOS FUTUROS

O desafio para os trabalhos futuros ficam por suprir os requisitos que por hora não puderam ser supridos ou implementados no software como, por exemplo, o modulo de controle financeiro, impressão de boletos.

Outro módulo é o de disponibilização de material digital, através do qual os professores poderão fazer upload de seu material teórico, e multimídia, para que seus alunos possam obtê-los sem precisar ir até a escola.

Pela sua implementação ser baseada em tecnologias quem permitem a fácil manutenção, a aplicação de melhorias e novos módulos torna-se menos dispendiosa e trabalhosa.

REFERÊNCIAS

ANDRADE, Bruno Pereira de; FARIA, Rodrigo Luis de. Sistema Colaborativo para Requisição de Software Acadêmico. Trabalho de Conclusão de Curso. Graduação. Sistemas de Informação. Universidade do Vale do Sapucaí. Pouso Alegre. 2010.

BARBOSA, Glenn William Rodrigues. **Estudo comparativo entre tecnologias para desenvolvimento Web.** 2007. Dissertação de Mestrado (Especialização em Desenvolvimento de Sistemas para a Web) – Departamento de Informática, Universidade Estadual de Maringá. Maringá, 2007.

BAUER, Christian; KING, Gavin. **Hibernate em ação**. Rio de Janeiro: Ciência Moderna, 2005.

CALDWELL, French. **Como organizar sua empresa para a conformidade.** Disponível em **<**http://info.abril.com.br/corporate/gartner/como-organizar-sua-empresa-para-a-conformidade.shtml>. Acesso em: 03 mai. 2011.

CONCEIÇÃO, Rodrigo Menezes da. Java Server Faces (JSF): Um estudo comparativo entre bibliotecas de componentes. 2008. Trabalho de Conclusão de Curso (Curso Superior de Bacharelado em Sistemas de Informação) – Universidade Tiradentes. Aracaju, 2008.

COSTA, Carlos, **Desenvolvimento para web**. 1ª edição: Lusocrédito, 2007.

ETAG SOLUÇÕES EM TECNOLOGIA. **Acorde** – Sistema de Gestão para Escolas de Música. Disponível em <http://etaginformatica.com.br/2010/08/acorde-sistema-administrativo-escolas-musica/>. Acesso em 03 mai. 2011.

GOMES, Nelma da S. **Qualidade de software** – Uma necessidade. Disponível em <http://www.fazenda.gov.br/ucp/pnafe/cst/arquivos/Qualidade_de_Soft.pdf>. Acesso em 04 mai. 2011.

GONÇALVES, Edson. **Desenvolvendo aplicações web com JSP, Servlets, Java Server faces, Hibernate, EJB 3 Persistence e Ajax**. Rio de Janeiro, RJ: Ciência Moderna, 2007.

ICEFACES.ORG **Icefaces**. Disponível em: http://www.icefaces.org/main/home//. Acesso em: 18 de agosto, 2011.

Integração Java Server Faces e Ajax: Estudo de integração entre tecnologias JSF e Ajax. **Revista Abstração,** Santa Catarina: UFSC, mai. 2009. Edição 2.

LIMEIRA, Jose Luiz S. **Utilização de Ajax no desenvolvimento de sistemas Web.** 2006. Trabalho de Conclusão de Curso (Especialização em Desenvolvimento de Sistemas para a Web) – Departamento de Informática, Universidade Federal do Rio Grande do Sul, 2006

MARAFON, Diego L. **Integração Java Server Faces e Ajax. Estudo da integração entre as tecnologias JSF e Ajax**. Trabalho de Conclusão de Curso. Florianópolis, 2006. Universidade Federal de Santa Catarina. Departamento de Informática, e Estatística. Curso de Ciências da Computação

PIMENTA, Marcelo S. WINCKLER, Marco. **Avaliação de Usabilidade de Sites Web.** 2002. Dissertação de Mestrado (Especialização em Desenvolvimento de Sistemas para a Web) – Departamento de Informática, Universidade Estadual de Maringá. Maringá, 2007

REZENDE, D. A. **Engenharia de software e sistemas de informação**. 2ª edição Rio de Janeiro: Brasport, 2002. 358 p.

SAMPAIO, Cleuton. Java Enterprise Edition 6: **Desenvolvendo Aplicações Corporativas**. 1. ed. Rio de Janeiro, RJ: Brasport, 2011. 280 p.

SCHERER, Richardson Wilson. **Desenvolvimento de uma Arquitetura de Serviços Web Para Redes de Sensores Sem Fio**. 2010. Trabalho de Conclusão de Curso (Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas) – Coordenação de Informática, Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2010.

SOMMERVILLE, I. **Engenharia de Software**. 6. ed. São Paulo: Pearson Addson-Wesley, 2005.

SOMMERVILLE, I. **Engenharia de Software**. 8. ed. São Paulo: Pearson Addson-Wesley, 2007.

SOUSA, M. **Unindo Java Server Faces a Ajax: melhorando o processo de desenvolvimento Web:** Disponível em: http://www.devmedia.com.br/visualizaConponente.aspx?comp=3199&site=6 . Acesso em: 26 de agosto, 2011.

WELLING, Luke. **PHP e MYSQL Desenvolvimento Web**. 2ª edição: Campus, 2003.