



Universidade Norte do Paraná

SISTEMA DE ENSINO PRESENCIAL CONECTADO
ESPECIALIZAÇÃO EM TECNOLOGIAS PARA APLICAÇÕES WEB

FÁBIO BRITO PINTO

SISTEMA WEB DE RESERVA DE QUARTOS:

Utilizando Frameworks Java EE

FÁBIO BRITO PINTO

SISTEMA WEB DE RESERVA DE QUARTOS:
Utilizando Frameworks Java EE

Trabalho de Conclusão de Curso apresentado à
Universidade Norte do Paraná - UNOPAR, como
requisito parcial para a obtenção do título de Especialista
em Tecnologias para Aplicações Web.

Orientador: Prof. .

São José dos Campos - SP
2014

Dedico este trabalho a minha esposa Cristiane,
minha filha Isabela, meu filho Erick e a minha
mãe Orminda.

AGRADECIMENTOS

Agradeço primeiramente, a Deus por me dar forças saúde e foco para percorrer essa jornada.

A minha companheira Cristiane e meus filhos Erick e Isabela pela compreensão e apoio

A minha mãe Orminda por apoio e incentivo de sempre estar lutando pelos meus sonhos.

“O futuro pertence àqueles que acreditam na
beleza de seus sonhos” (Eleanor Roosevelt)

BRITO PINTO, Fábio Brito. **Sistema WEB para Reserva de quartos**: Utilizando Frameworks Java EE. 2014. Número total de folhas 93. Trabalho de Conclusão de Curso Graduação em Especialização em Tecnologia para Aplicações WEB – Centro de Ciências Empresariais e Sociais Aplicadas, Universidade Norte do Paraná, São José dos Campos, 2014.

RESUMO

Este Trabalho demonstra as tecnologias da plataforma Java WEB, seus preceitos, fundamentos e características, bem como a elaboração de um Projeto de Reserva para hotéis como projeto de obtenção da Especialização. O projeto aborda técnicas para a construção de uma aplicação com os principais frameworks da atualidade, serão apresentados métodos de cadastro, consultas, alterações e exclusão de dados.

Palavras-chave: Java. Tomcat. JSF 2. MVC. Bootstrap. Primefaces. JavaBeans. ORM. Hibernate. Spring. Ireports

BRITO PINTO, Fábio. **Web System for Reservation**: using Java EE Frameworks. 2014. Total number of leaves 93. Working End of Course Undergraduate Specialization in Technology for Web Applications - Centre for Applied Social and Business Sciences, University of Northern Paraná, São José dos Campos, 2014.

ABSTRACT

This work demonstrates the Java web platform technologies, its precepts, fundamentals and characteristics, as well as the preparation of a Draft Reservation for hotels as a project for obtaining Specialization. The project covers techniques for building an application with main frameworks of the present, methods of registration, queries, changes and deletion of data.

Key-words: Java. Tomcat. JSF 2. MVC. Bootstrap. Primefaces. JavaBeans. ORM. Hibernate. Spring. Ireports

LISTA DE FIGURAS

Figura 1 - Logotipo Oficial do Eclipse	18
Figura 2 - Logo do Tomcat	20
Figura 3 - Ciclo de Vida do Servlet.....	22
Figura 4 - Logo JavaServer Faces	22
Figura 5 - Diagrama do Fluxo da Arquitetura MVC	25
Figura 6 - Exemplo de Organização de um Template	28
Figura 7 - Exemplo de Hierarquia da pasta resources	29
Figura 8 - Logo Primefaces	29
Figura 9 - Logo MySql	31
Figura 10 - Relação dos atributos do Modelo Relacional com o Modelo Orientado ao Objeto.....	33
Figura 11 - Logo Hibernate	33
Figura 12 - Arquitetura simplificada do Hibernate	34
Figura 13 - Controle de Acesso Spring Security.....	38
Figura 14 - Fluxo de Execução de um Relatório	39
Figura 15 - Protótipo da Tela de Cadastro de Apartamento.....	44
Figura 16 - Diagrama de Caso de Uso.....	45
Figura 17 - Diagrama de Classe.....	46
Figura 18 - Tela de Login acessada via Menu.....	47
Figura 19 - Tela de Login via URL.....	48
Figura 20 - Tela de Erro de Login.....	48
Figura 21 - Tela de cadastro e Edição de Apartamento	49
Figura 22 - Tela de Cadastro de Hospede	50
Figura 23 - A tela de Listagem dos Apartamentos cadastrados	51
Figura 24 - Tela de Listagem de Hospede	51
Figura 25 - Tela com Informações do Hospede	52
Figura 26 - Tela de Reserva.....	53

LISTA DE GRÁFICOS

Gráfico 1 - Ranking 2014 de Ferramentas e Tecnologias Java.....	17
---	----

LISTA DE TABELAS

Tabela 1 - Modelo Relacional e Modelo Orientado ao Objeto	32
---	----

LISTA DE ABREVIATURAS E SIGLAS

Checkilists	Lista de Validação de Requisitos do sistema
Check-in	É a hora que o cliente dá entrada na hospedagem
Check-out	É a hora que o cliente dá finaliza sua hospedagem
CRUD	Create, Read, Update e Delete. São as quatro operações básicas realizadas no banco de dados (criar, ler, atualizar e excluir).
CSS	Cascading Style Sheet (Folhas de Estilo em Cascata)
DAO	Data Access Object
EJB	Enterprise Java Beans
Framework	Conjunto de componentes com um objetivo específico
HTML	Linguagem de Marcação de Hipertexto (HyperText Markup Language)
JPA	Java Persistence API é o padrão de persistência de dados no Java
JSF	JavaServer Faces
LGPL	GNU Lesser General Public License, permite ao desenvolvedor restringir o que será disponibilizado no código fonte para utilização por terceiros.
MVC	Modelo-Visão-Controle (Model-View-Controller)
MySQL	Linguagem de Consulta Estruturada (Structured Query Language)
ORM	Mapeamento Objeto Relacional (Object Relational Mapper)
Paas	Platform as a Service (Plataforma como Serviço)
POJO	Plain Old Java Objects (Velho e Simples Objeto Java)
SGBD	Sistema de Gerenciamento de Banco de Dados
SGBD	Sistema de Gerenciamento de Banco de Dados
SQL	Linguagem de Consulta Estruturada (Structured Query Language)
Stakeholders	É a parte interessada que utilizara o sistema proposto.
UI	Interface de Usuário (User Interface)
UML	Linguagem de Modelação Unificada (Unified Modeling Language)
UNOPAR	Universidade Norte do Paraná

URL	É um endereço web
W3C	(World Wide Web Consortium) é a organização sem fins lucrativos que tem por objetivo potencializar o máximo das tecnologias da Web.
XHTML	Linguagem Extensível de Marcação de Hipertexto (Extensible HyperText Markup Language), é baseada no XML
XML	Linguagem Extensível de Marcação (Extensible Markup Language)

SUMÁRIO

1	Introdução	13
2	Justificativa.....	14
3	Objetivos	15
3.1	Objetivo geral	15
3.2	Objetivos específicos.....	15
4	Desenvolvimento.....	16
5	Tecnologia.....	17
5.1.1	IDE Eclipse	18
5.1.1.1	Apache Maven	18
5.1.2	Linguagem de Programação Java	19
5.1.3	Servlet Container	19
5.1.3.1	Apache Tomcat.....	20
5.1.4	Servlets	21
5.1.4.1	JavaServer Faces (JSF)	22
5.1.5	MVC.....	24
5.1.5.1	Model - Modelo	25
5.1.5.2	View - Visão	25
5.1.5.2.1	Estrutura das Páginas	26
5.1.5.2.2	Estilo das Páginas	26
5.1.5.2.3	Primefaces	29
5.1.5.3	Controller	30
5.1.6	Banco de dados	30
5.1.6.1	MySQL	31
5.1.7	ORM (Object Relational Mapper – Mapeamento Objeto Relacional)	31
5.1.7.1	Hibernate	33
5.1.7.2	DAO – Data Access Object.....	35
5.1.8	Segurança do Sistema.....	36
5.1.8.1	Spring Security.....	37
5.1.9	Ireports e Jasper Reports.....	38
6	Levantamento de Requisitos.....	40
6.1	Descrição do sistema existente	40
6.2	Descrição do Sistema Proposto	40

6.3	Requisitos Funcionais.....	40
6.3.1	Os Atores e suas hierarquias no Sistema	41
6.3.1.1	O Administrador	41
6.3.1.2	O Hospede.....	41
6.3.1.3	O Visitante	42
6.4	Requisitos Não Funcionais	42
6.4.1	Do cadastro.....	42
6.4.2	Da exclusão e alteração de: Apartamentos, hospedes e reservas	43
6.5	Validação de Requisitos	43
6.6	Gerenciamento de requisitos utilizados	43
7	Diagrama de caso de Uso	45
8	Diagrama de Classes	46
9	Telas (Prototipação).....	47
9.1	Tela de Login	47
9.2	Telas de Cadastro e Edição	48
9.3	Telas de Listagem e Exclusão	50
10	Código Fonte	54
10.1	Código Fonte SQL	54
10.2	Código Fonte web.xml	57
10.3	Código Fonte Pom.xml	59
10.4	Código Fonte VIEW - CADASTROS.....	62
10.5	Código Fonte da página com dados salvos no banco de dados com possibilidade de exclusão e alteração	66
10.6	Código Fonte da página com dados salvos no banco de dados	69
10.7	Código Fonte TEMPLATE	70
10.8	Código Fonte MODEL	74
10.9	Código Fonte CONTROLLER.....	77
10.10	Código Fonte DAO.....	79
10.11	Código Fonte CONVERTER	80
10.12	Código Fonte SERVICE	81
10.13	Código Fonte NEGOCIOEXCEPTION.JAVA	82
10.14	Código Fonte persistence.xml.....	82
10.15	Código Fonte EntityManagerProducer.java	83
10.16	Código Fonte applicationcontext.xml	84

11	Conclusão.....	85
	Referências	86
	Apêndices	89
	Apêndice A – Ficha de Entrevista	90

1 INTRODUÇÃO

A utilização de aplicativos via internet tem se tornado cada vez mais uma tendência, um exemplo que essas aplicações estão em alta é o surgimento diário de grandes sites que tem funcionalidades iguais e até superiores que muitos aplicativos Desktop.

A facilidade de acesso à internet através de celulares, tablets e outros dispositivos móveis proporciona uma nova forma de desenvolvimento de aplicações é executada do lado do servidor, onde os usuários não necessitam mais realizar a instalação de qualquer programa, para usá-lo basta apenas acessar a aplicação na internet, outra vantagem é que tais aplicativos sempre estarão atualizados, pois não obriga os usuários a fazerem nenhuma atualização de nenhum software.

Na realização deste Projeto foi requerido uma implementação de um sistema web que armazene os dados em um banco de dados, para atender esta solicitação será criado um Projeto de Reserva de Quartos via internet, onde os hóspedes poderão realizar seu cadastro e escolher o quarto e efetuar a sua pré-reserva.

2 JUSTIFICATIVA

Para facilitar a comunicação de clientes e fornecedor faz se necessária a utilização de um intermediário, onde o site atua como tal, muitas vezes o cliente tem que ficar percorrendo de hotel em hotel para encontrar uma vaga ou até mesmo para saber o valor de hospedagem que lhe convém, já pelo lado do prestador de serviço que pode perder a oportunidade de alugar um quarto pois o telefone estava ocupado ou seu site não possuía informações atualizadas dos valores praticados por ele.

Sendo assim, o sistema apresentado oferece o recurso de intermediador entre as partes, facilitando a concretização do negócio entre as partes, uma vez que o cliente pode de qualquer dispositivo conectado à internet fazer consultas de quartos disponíveis, visualizar os valores praticados. Já o fornecedor do quarto tem seu produto disponível 24 horas para consultas e reservas.

3 OBJETIVOS

3.1 OBJETIVO GERAL

Este projeto tem como objetivo implantar um sistema de reservas, possibilitando ao hotel ofertar seus serviços de reservas de quartos pela internet, permite que o hospede possa efetuar seu pré-cadastro que será posteriormente validado no check-in do estabelecimento.

3.2 OBJETIVOS ESPECÍFICOS

O sistema proposto oferecerá ao dono do estabelecimento um controle de clientes e de quartos, através do sistema será possível saber quais os quartos que estão livres e ocupados e a sua disponibilidade para datas futuras.

O dono do estabelecimento poderá inserir, consultar, alterar e excluir apartamentos. No cadastro de clientes o estabelecimento poderá efetuar operações de bloqueio de reserva por cliente.

Para efetuar essas operações os usuários deverão ter privilégios de administrador do sistema, evitando que dados sejam alterados indevidamente por terceiros.

4 DESENVOLVIMENTO

Para atender as necessidades do sistema em questão, o sistema foi concebido utilizando tecnologias Java para o desenvolvimento das páginas WEB.

Inicialmente, o sistema foi dividido em partes utilizando o padrão MVC; Foram criados os documentos com a extensão XHTML como parte visual com a utilização de divs com tags CSS.

Posteriormente foi criado classes Java na camada Model para serem controladas pelos Controllers, os Controles que são classes Java que possuem as regras de negócio da aplicação onde ficam localizados os Manager Beans.

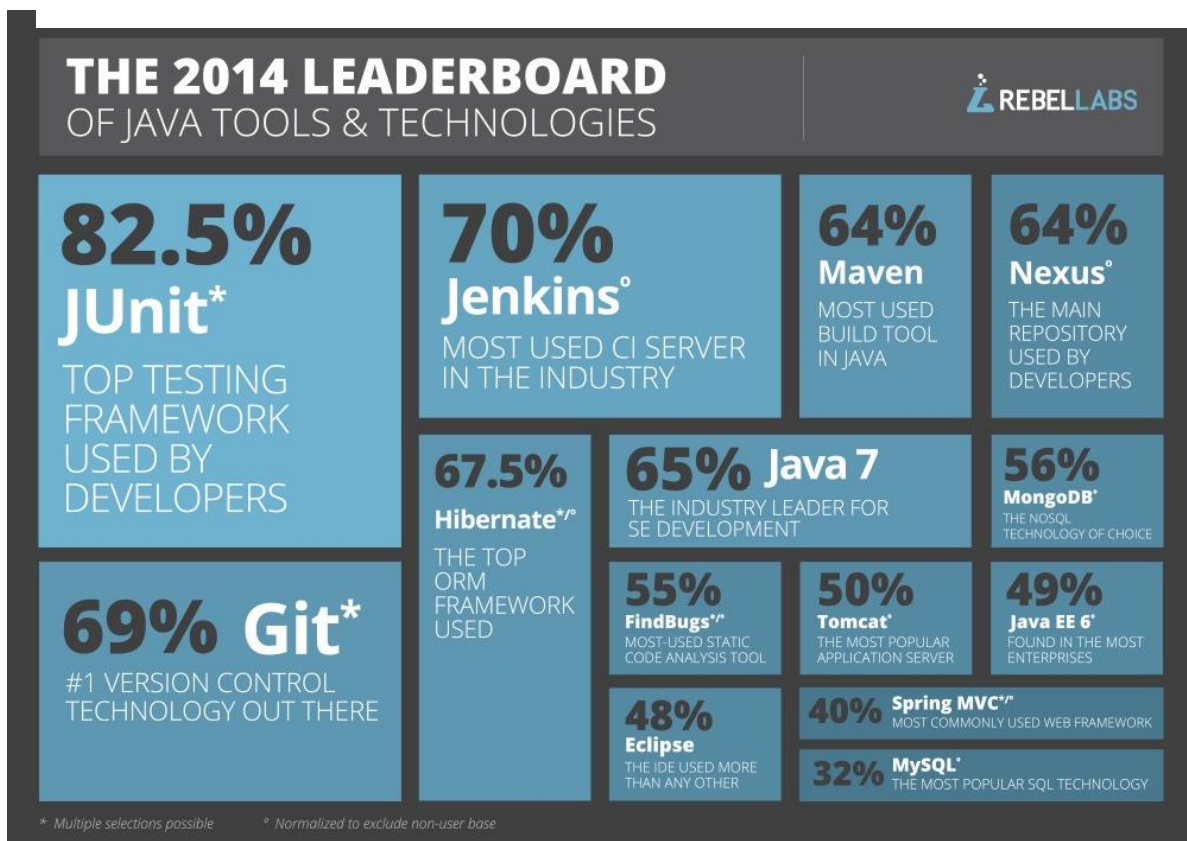
As páginas foram desenvolvidas com intuito de serem responsivas e atrativas para os usuários visitantes do site, por comodidade e facilidade de manutenção serão utilizados templates como bases para as páginas.

5 TECNOLOGIA

A maioria das ferramentas e frameworks escolhidos para serem utilizados no desenvolvimento deste trabalho é recomendado pelos desenvolvedores que utilizam a plataforma Java EE atualmente.

Segundo uma pesquisa realizada pelo site ZeroTurnAround (<http://zeroturnaround.com/rebellabs/>) que realizou uma entrevista global a 2.164 Profissionais Java em 2014, foi constatada qual a porcentagem que essas tecnologias são utilizadas.

Gráfico 1 - Ranking 2014 de Ferramentas e Tecnologias Java



Fonte: REBELLABS. Disponível em: < http://pages.zeroturnaround.com/Java-Tools-Technologies.html?utm_source=Java%20Tools%20&%20Technologies%202014&utm_medium=allreports&utm_campaign=rebellabs&utm_rebellabsid=88/>. Acesso em: 25 maio 2014.

Das 14 tecnologias apontadas no ranking de 2014 serão utilizadas nesse trabalho 07 tecnologias, são elas: Eclipse, Hibernate, Maven, Java 7, Java EE 6, Tomcat e o MySQL.

5.1.1 IDE Eclipse

A escolha pela ferramenta Eclipse como ferramenta de desenvolvimento foi devido à grande quantidade de plug-ins disponíveis para a instalação, para utilizar com o Eclipse basta descompactar o arquivo e o mesmo já está pronto para uso, o Eclipse está disponível para vários sistemas operacionais como o Windows, Linux e Mac OS. O Eclipse é open-source e é patrocinado por grandes empresas de tecnologia do mundo que pode ser verificado no link (<https://eclipse.org/membership/showAllMembers.php>), dentre elas podemos destacar a Intel, Cisco, Oracle, Red Hat e etc.

A ferramenta Eclipse foi inicialmente desenvolvida pela IBM e depois doada para a comunidade. Hoje o Eclipse não é apenas uma IDE de desenvolvimento Java, mas uma plataforma de desenvolvimento de código aberto. (LUCKOW, Décio Heinzemann; MELO, Alexandre Altair. Programação Java para a Web. São Paulo: Novatec Editora, 2010, p. 46).

Figura 1 - Logotipo Oficial do Eclipse



Fonte: Eclipse Logos. Disponível em: <<http://www.eclipse.org/artwork/>>. Acesso em: 12 maio 2014.

5.1.1.1 Apache Maven

O Maven é responsável por gerenciar dependências de bibliotecas em projetos Java. Projetos feitos em Maven possuem um arquivo POM.XML (Project Object Model), é nesse arquivo que são declaradas as dependências que serão utilizadas pelo projeto, após declarar as dependências nesse arquivo o Maven faz o download automaticamente das bibliotecas jars e suas dependências declaradas nesse arquivo.

A utilização do Maven evita que o desenvolvedor tenha que ficar baixando os jars direto do site dos fabricantes dos frameworks utilizados, pois até que se ache o arquivo correto consome um grande tempo, com a utilização do Maven os downloads são realizados automaticamente, pode acontecer em alguns

casos do Maven não possuir a biblioteca que precisamos em seu repositório, mas basta adiciona-la na lista de repositórios que o Maven realiza o download. A utilização do Maven em projetos com vários desenvolvedores cria um padrão de projeto, pois a utilização do arquivo POM.XML garante que os desenvolvedores sempre estarão utilizando as mesmas versões de bibliotecas, pois no arquivo POM.XML é declarada a versão do jar que está sendo utilizada.

Maven é uma ferramenta da Apache Software Foundation para gerenciamento de dependências e automação de *build*, principalmente em projetos Java. Um projeto que usa Maven possui um arquivo XML (*pom.xml*) que descreve o projeto, suas dependências, detalhes do *build*, diretórios, plugins requeridos, etc. (ANDRADE, Thiago Faria de. Java EE 7 com JSF, PrimeFaces e CDI. São Paulo: AlgaWorks Softwares, Treinamentos e Serviços LTDA, 2013, p. 20).

5.1.2 Linguagem de Programação Java

O desenvolvimento da aplicação será feito utilizando a linguagem Java 7, utilizando a plataforma Java Enterprise Edition (Java EE 6) que é o Java para aplicações corporativas. A escolha dessa linguagem foi devido a sua maturidade, estabilidade e segurança. Outro fator que mais chama a atenção para essa linguagem é a quantidade de frameworks disponíveis no mercado, o que facilita e muito o desenvolvimento de qualquer aplicação tanto Web quanto Desktop. Segundo Caelum (Apostila FJ21, Versão 17.0.6, 2014, p. 215), “O mercado de frameworks Java é imenso.” O Java possui o desenvolvimento Orientado ao Objeto que aborda técnicas como herança, polimorfismo, encapsulamento e abstração facilitando o processo de desenvolvimento de aplicações.

A linguagem Java nos dias de hoje é utilizada por grandes bancos, pois fornece extrema segurança. Também é utilizada por grandes empresas que desejam trafegar uma grande quantidade de dados e necessita de estabilidade e portabilidade entre outras empresas. (GONÇALVES, Edson. Desenvolvendo Aplicações Web, 2007, p. 7).

5.1.3 Servlet Container

Para que páginas Java Web funcionem, as requisições são enviadas

para o servidor de aplicação, o servidor recebe essas requisições e as processa, após processar essas requisições o servidor devolve uma resposta através de uma página para o solicitante.

Segundo ANDRADE, (2010, p. 21) “Um container web é um programa que recebe requisições HTTP, executa componentes Java (Servlets) e devolve para o usuário (browser) código HTML, além de todos os outros recursos necessários (como imagens, vídeos, folhas de estilo e etc). Na verdade, trata-se de um servidor web (como o Apache HTTPD), porém com outras funcionalidades para executar código Java de acordo com a especificação Java EE”.

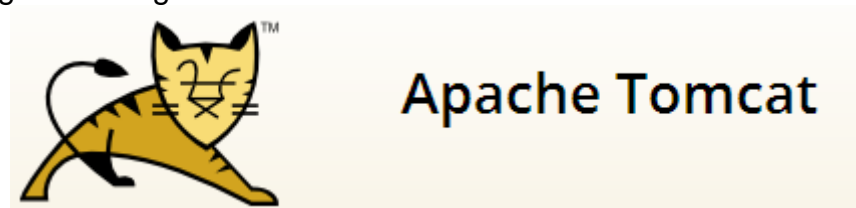
Containers são interfaces entre componentes e funcionalidades de baixo nível específicas de uma plataforma. Para uma aplicação web desenvolvida em Java ou um componente corporativo ser executado, eles precisam ser implantados em um container.

Os containers também são chamados de servidores de objetos, ou servidores de aplicação, pois oferecem serviços de infra-estrutura para execução de componentes. (ANDRADE, Thiago Faria de. Java EE 7 com JSF, PrimeFaces e CDI. São Paulo: AlgaWorks Softwares, Treinamentos e Serviços LTDA, 2013, p. 14).

Os containers disponibilizam serviços de infraestrutura para as aplicações, proporcionando os requisitos necessários para que os sistemas funcionem adequadamente, tirando essa responsabilidade do desenvolvedor.

5.1.3.1 Apache Tomcat

Figura 2 - Logo do Tomcat



Fonte: Tomcat Logo. Disponível em: <<http://tomcat.apache.org/legal.html>>. Acesso em: 12 maio 2014.

O Container utilizado no decorrer desse trabalho será o Apache Tomcat por ser um servidor para aplicações web e por implementar as especificações Java EE (J2EE) como Servlet e JSP. O Tomcat é um software livre e um servidor leve pois possui o mínimo para aplicações Java Web rodarem. Existem outros servidores como o GlassFish Server da Oracle, JBoss Application Server da

Red Hat dentre outros, cada um desses servidores possui suas particularidades e especificações.

O Apache Tomcat é um contêiner Java e um Servidor web ao mesmo tempo. Ele suporta a execução das tecnologias Java Servlet e JavaServer Pages (JSP), o que permite que o Java funcione para um ambiente web. (LUCKOW, Décio Heinzemann; MELO, Alexandre Altair. Programação Java para a Web. São Paulo: Novatec Editora, 2010, p. 33).

5.1.4 Servlets

Servlets são classes Java e por esse motivo podem utilizar recursos nativos da linguagem, os Servlets estendem a classe `javax.servlet.http.HttpServlet`, possuem a funcionalidade de receber as solicitações através protocolo HTTP, essas solicitações podem ser `doGet`, `doPost` e etc.

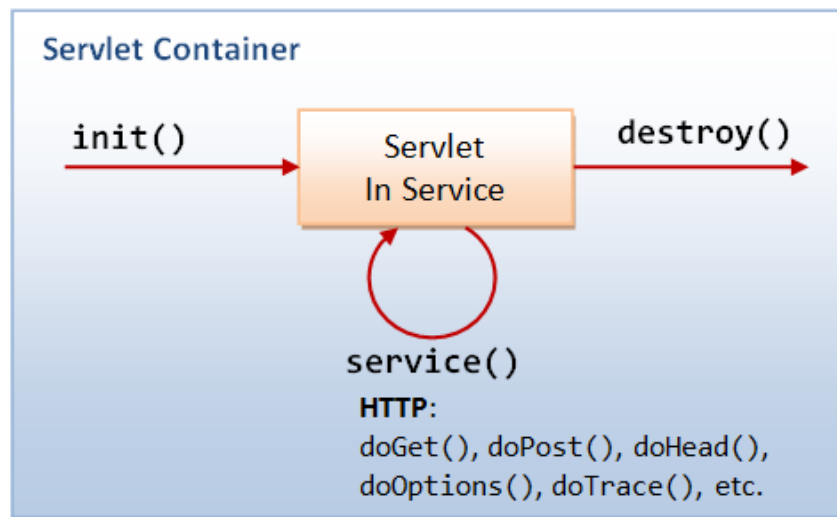
Segundo GONÇALVES (2007, p. 29) “Servlets são classes Java, desenvolvidas de acordo com uma estrutura bem definida, e que, quando instaladas junto a um Servidor que implemente um Servlet Container (um servidor que permita a execução de Servlets, muitas vezes chamado de Servidor de Aplicações Java), podem tratar requisições recebidas de clientes”.

Tecnicamente falando, um "servlet" é uma classe Java no Java EE que obedece à API Java Servlet, um protocolo pelo qual uma classe Java pode responder a requisições. Servlets podem, em princípio, se comunicar sobre qualquer protocolo cliente-servidor, mas eles são na maioria das vezes usados com o protocolo HTTP. Desta forma, "servlet" normalmente é usado como uma abreviação para "servlet HTTP". Assim, um desenvolvedor de software pode usar um servlet para adicionar conteúdo dinâmico para um servidor web usando a plataforma Java. O conteúdo gerado normalmente é HTML, mas pode ser outro dado como XML. Servlets podem manter estado em variáveis de sessão através de muitas transações de servidores, usando cookies HTTP ou reescrita de URL. (Disponível em <<http://pt.wikipedia.org/wiki/Servlet>>. Acesso em: 22 maio 2014).

Os Servlets possuem um tempo de vida que é controlado pelo Container, logo após criar o Servlet através de solicitações HTTP, o Container chama o método `init()` para carregar os recursos para serem executados, para que os pedidos sejam executados, processados e posteriormente após respondam a solicitação o Container chama o método `service()`, esse método possui dois parâmetros `ServletRequest` que possui a requisição do cliente e `ServletResponse` que contém a resposta do Servlet. Após receber a resposta solicitada e verificada se

não possui mais solicitações se faz necessário liberar os recursos para outras possíveis solicitações, nesse momento o Container chama outro método o `destroy()` para a liberação dos recursos que estavam sendo utilizados na requisição do Servlet.

Figura 3 - Ciclo de Vida do Servlet



Fonte: Servlets : Calling init, service and destroy explicitly. Idle Brains. Disponível em: <<http://idlebrains.org/tutorials/java-tutorials/servlets-init-service-destroy/>>. Acesso em: 24 maio 2014.

5.1.4.1 JavaServer Faces (JSF)

Figura 4 - Logo JavaServer Faces



Fonte: Winner of the JSF Spec Logo Contest: Wilber Saca. Disponível em: <<https://weblogs.java.net/blog/edburns/archive/2011/05/11/winner-jsf-spec-logo-contest-wilber-saca>>. Acesso em: 24 maio 2014.

O JavaServer Faces (JSF) é um framework para desenvolvimento de interfaces para aplicações Java EE, esse framework é baseado no Modelo MVC, que é um modelo de desenvolvimento que separa o desenvolvimento em camadas.

O JSF é definido pela JCP (Java Community Process) que é a entidade que define as especificações da tecnologia Java, o JCP é constituído por grandes empresas da área de tecnologia, tais como: IBM, Oracle, dentre outras. Atualmente a versão disponível é a Mojarra com o JSF 2.2.

Devido esse apoio e especificações para os especialistas LUCKOW e MELO (2010, p. 72) “Isso torna o JSF imediatamente um padrão de mercado. O fato de ser uma especificação formal e segura permite que essas empresas e outras que se interessarem invistam na tecnologia e desenvolvam ferramentas e componentes para o desenvolvimento web do JSF”.

O JSF possui um conjunto de componentes visuais prontos, essas interfaces recebem o nome de User Interface (UI), mas caso seja necessário podem ser adicionados outros componentes de terceiros, alguns pagos e outros gratuitos, mas se por ventura o desenvolvedor necessitar de algo muito específico o mesmo pode desenvolver os componentes que achar necessário, as tags de códigos do JSF são parecidas com as tags do HTML o que facilita o desenvolvimento pela familiaridade dos comandos.

O JSF possui recursos como validadores de campos que retornam uma mensagem para o usuário da aplicação que o dado informado é diferente do que está sendo esperado pelo sistema, essas validações vão desde campos em branco até dados digitados erroneamente, essas verificações ocorrem após a conversão dos dados enviados pela aplicação. Essa conversão é necessária porque os dados enviados através de páginas web utilizam o protocolo HTTP gerando o conteúdo no formato texto, o JSF converte esses textos automaticamente para serem processados no servidor de uma forma mais específica. Segundo LUCKOW e MELO (2010, p. 339) “O JavaServer Faces já tem por padrão conversores para todos os tipos de dados numéricos do Java, como BigDecimal, BigInteger, Byte, Double, Float, Integer, Long e Number, para o tipo lógico boolean, para Enumerations e tipo texto, como o Character. A conversão para String não exige conversor, pois os dados sempre trafegam em texto”.

JavaServer Faces (JSF) é um framework MVC baseado em Java para a construção de interfaces de usuário baseadas em componentes para aplicações web. Possui um modelo de programação dirigido a eventos, abstraindo os detalhes da manipulação dos eventos e organização dos componentes, permitindo que o programador se concentre na lógica da aplicação. (Disponível em: <<http://pt.wikipedia.org/wiki/Jsfc>>. Acesso em 25 maio 2014).

Segundo (LUKNOW, MELO) “Existem quatro passos básicos para iniciar um desenvolvimento utilizando o JSF:

- Criação da classe Beaking Bean (classe Bean).
- Mapeamento da classe Bean.
- Criação da página JSF.
- Mapeamento da navegação entre páginas.

Esses passos não são uma regra no desenvolvimento de aplicações JSF, mas servem como uma orientação dos passos que desenvolver deverá seguir no desenvolvimento de uma aplicação.

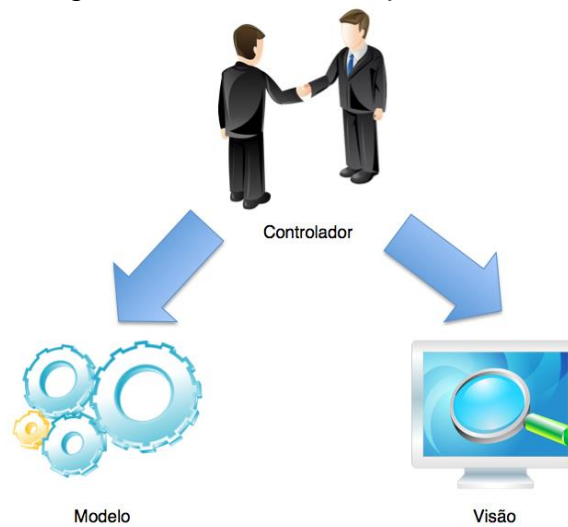
5.1.5 MVC

Como forma de desenvolvimento será utilizado a arquitetura MVC (Model-View-Controller), essa arquitetura sugere a separação em três camadas de desenvolvimento (Modelo, Visão e Controle), essa separação facilita o desenvolvimento e manutenção do código. Segundo LUCKOW e MELO (2010, p. 180) “... o principal objetivo do MVC não é definir como separar em camadas, mas sim definir como as camadas devem interagir”. Utilizando esse conceito o sistema fica modular, onde a troca de uma tecnologia não tem grandes interferências de uma camada para a outra.

A separação lógica da aplicação nestas partes assegura que a camada Modelo não sabe nada praticamente do que é exibido; restringido por representar as partes de componentes do problema que é resolvido pela aplicação. Igualmente, a camada de Apresentação só está relacionada a exibir os dados e não com implementar lógica de negócios que é controlada pela camada Modelo. O Controlador, como um gerenciador de tráfego, dirige as apresentações a serem exibidas e com as devidas mudanças de dados e recuperações vindas da camada Modelo. (GONÇALVES, Edson. Desenvolvendo Aplicações Web, 2007, p. 7).

A arquitetura MVC diz que a camada Model não pode interagir com a camada View e nem a camada View com a Model. O Controller atua como um intermediador dessas duas camadas e é o único elo de comunicação entre essas camadas. A figura 5 ilustra a comunicação entre as camadas, percebe-se claramente que o Controller é o intermediador entre as duas partes. O grande foco dessa arquitetura é desacoplar a relação entre visão e modelo.

Figura 5 - Diagrama do Fluxo da Arquitetura MVC



Fonte: FILHO. Criando aplicações Web com EJB e padrões de projeto. Disponível em: <<http://www.devmedia.com.br/criando-aplicacoes-web-com-ejb-e-padroes-de-projeto-revista-easy-java-magazine-28/27480>>. Acesso em: 01 jun 2014.

5.1.5.1 Model - Modelo

A camada Model possuem classes Java, comumente chamadas de POJO (Plain Old Java Objects). Para ANDRADE (2010 p.36) “As classes de entidade devem seguir o estilo JavaBeans, com métodos getters e setters. É obrigatório que essas classes possuam um construtor sem argumentos”. Essas classes podem conter Annotations nos atributos das classes. Quando da utilização de ORM estas também devem possuir métodos sobrescritos da classe Object, esses métodos são o equals() e o hashCode(), isso é feito para assegurar que um objeto é realmente o que está sendo utilizado, assim evitamos alterações indevidas.

O Model (Modelo) é o objeto que representa os dados do programa. Maneja esses dados e controlam todas suas transformações. Esse modelo não tem conhecimento específico dos controladores (Controller) e das apresentações (Views), nem sequer contém referência a eles. Portanto, o Model são as classes que trabalham no armazenamento e busca de dados. (GONÇALVES, Edson. Desenvolvendo Aplicações Web, 2007, p. 386).

5.1.5.2 View - Visão

A camada View (Apresentação) é onde estão as telas de interação entre o usuário e o aplicativo, todo design da aplicação está contido nessa camada,

elas são formadas por páginas web e seus componentes. Nessa camada que os usuários fazem solicitações. Essas solicitações são processadas pelo Controller, que posteriormente, faz interações com a camada Model, após obter o resultado o Controller apresenta o resultado da solicitação do usuário em uma tela da camada View.

A View (Apresentação) é o que maneja a apresentação visual dos dados representados pelo Model. Em resumo, é a responsável por apresentar os dados resultantes do Model ao usuário. (GONÇALVES, Edson. Desenvolvendo Aplicações Web, 2007, p. 386).

5.1.5.2.1 Estrutura das Páginas

Para a estruturação das páginas web será utilizada a linguagem XHTML (Extensible HyperText Markup Language). O XHTML é baseado no XML, é considerada uma evolução da linguagem HTML porém com uma estrutura mais robusta. O XHTML surgiu porque com as páginas em HTML não era possível a visualização das páginas em certos dispositivos, como celulares, tvs e etc. A W3C padronizou o XHTML como linguagem padrão para a Web devido a flexibilidade do XML.

O XHTML é, antes de tudo, um arquivo XML e, por esse motivo, segue todas as regras de sintaxe e estrutura XML. Diversas melhorias na estrutura do HTML foram feitas na criação do XHTML. Essas melhorias visaram retirar tags que caíram em desuso ou que entravam em conflito com os novos objetivos propostos para o HTML. (LUCKOW, Décio Heinzemann; MELO, Alexandre Altair. Programação Java para a Web. São Paulo: Novatec Editora, 2010, p. 105).

5.1.5.2.2 Estilo das Páginas

Para a formatação da aparência visual da aplicação foi utilizado o CSS (Cascading Style Sheets), esse arquivo contém configurações de formatação da aparência visual dos elementos da página web. O CSS centraliza todas as formatações de estilo em um único arquivo, facilitando a localização e manutenção das páginas web. A utilização do CSS facilita a manutenção do código pois separa a formatação da aparência visual do conteúdo das páginas.

CSS é um acrônimo para Cascading Style Sheets, que podemos traduzir para Folhas de Estilo em Cascata. Os estilos definem o layout, por exemplo, a cor, o tamanho, o posicionamento e são declarados para um determinado elemento HTML. O CSS altera as propriedades visuais daquele elemento e, por cascata, todos os seus elementos filhos. Em geral, o HTML é usado para estruturar conteúdos da página (tabelas, parágrafos, div etc) e o CSS formata e define a aparência. (CAELUM, Java para Desenvolvimento WEB Curso FJ-21. 2014, p. 193).

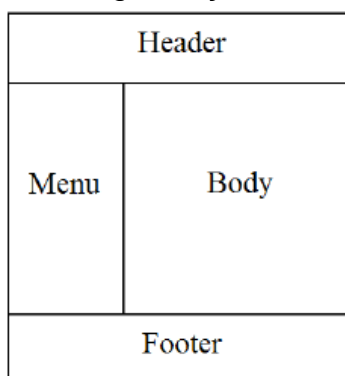
Atualmente existem uma grande gama de frameworks CSS já prontos, bastando apenas inclui-la no seu site, com o objetivo de obter uma melhor aparência na aplicação, na aplicação proposta será utilizado o Bootstrap. O Bootstrap foi desenvolvido pela equipe do Twitter, além da aparência visual com ele é possível ajustar o site para a resolução do dispositivo que o acessa.

Como descrito por CAELUM (2014, p.176) “O Bootstrap traz uma série de recursos: Reset CSS, Estilo visual base pra maioria de tags, Ícones, Grids prontos pro uso, Componentes CSS, Plugins JavaScript, Tudo responsivo e mobile-first”. Isso vem tornando sua utilização mais popular entre os desenvolvedores front-end, pois além de possuir uma aparência amigável ele ainda é responsivo, tornando o acesso disponível em qualquer dispositivo que tenha um browser.

Como forma de padronização das páginas será utilizado a tecnologia de Facelets para construção de templates. Os templates possuem uma base estrutural que podem ser utilizados em todas as páginas da aplicação. A utilização de Facelets facilita a manutenção da página, pois ao aplicar uma mudança na estrutura do Template ela se propaga na aplicação inteira. Outra vantagem é a utilização de componentes, basta declara-los no Template que eles estarão disponíveis para utilização em todas as páginas que fazem o uso destes templates.

A utilização de templates facilita a padronização da aplicação, já que conseguimos deixar fixo tudo aquilo que raramente muda, como topo, rodapé e menu lateral das páginas, bem como informar somente a parte que muda, que geralmente é o meio da página. (CORDEIRO, Aplicações Java para web com JSF e JPA. 2010, p. 226).

Figura 6 - Exemplo de Organização de um Template



Fonte: Cordeiro (2010, p. 226)

Segundo o especialista Coelho (JSF Eficaz. 2010, p. 68) aponta as seguintes vantagens na utilização de Facelets com XHTML:

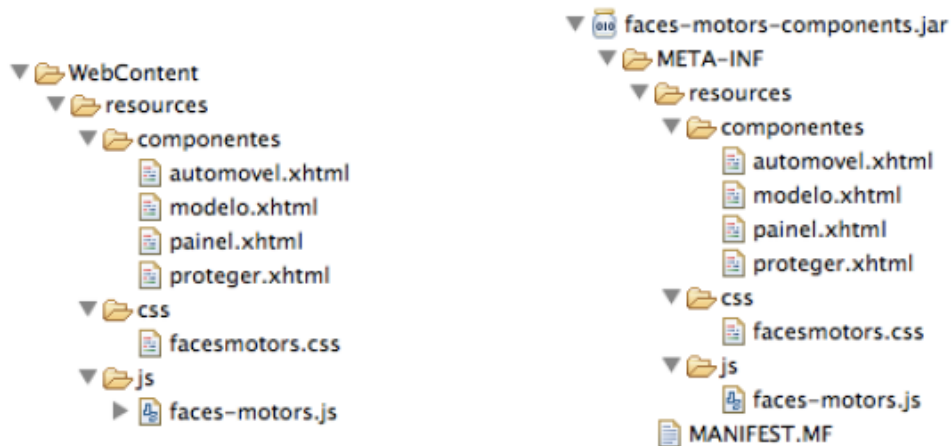
- XHTML com Facelets podem executar até 50% mais rápido que JSP;
- Todas as tags utilizadas nas páginas são declaradas de modo dinâmico, não necessitam estar declaradas em um TLD (Tag Library Descriptor);
- Facelets permitem Templates de modo avançado, em que a reutilização do código é muito alta.

Com a utilização desses recursos ganhasse tempo no desenvolvimento e facilita futuras alterações estruturais, pois uma alteração feita replica no site inteiro, centraliza todas as configurações visuais em um único local, provendo uma melhor organização do conteúdo das páginas.

Foi convencionado a partir do JSF 2 um local padrão para armazenamento de arquivos de CSS, imagens e scripts, esses arquivos devem ser armazenados na pasta resources dentro da WebContent ou Webapp dependendo do padrão adotado para o desenvolvimento. Para a utilização de uma imagem basta indicar em qual pasta ela se encontra e qual o nome da imagem, isso vale também para scripts e arquivos CSS. Podemos ainda criar vários subdiretórios se necessário dentro da pasta resources para uma melhor organização de conteúdo.

Abaixo é mostrada uma estrutura básica de como deve ser criado a pasta resources:

Figura 7 - Exemplo de Hierarquia da pasta resources



Fonte: (CORDEIRO, Aplicações Java para web com JSF e JPA. 2010, p. 167).

Ao utilizarmos esse recurso não precisamos nos preocupar em declarar todo o contexto para localizarmos uma imagem, pois ao informar a library que é a pasta onde se encontra o arquivo e o nome que é o nome do arquivo o JSF já localiza automaticamente. Outro recurso é que podemos utilizar expressões regulares para compor o nome da imagem.

A base de tudo é a pasta resources, que pode tanto estar na raiz do contexto web quanto na pasta META-INF dentro de um jar ou no classpath da aplicação. Temos basicamente três componentes básicos do JSF que lidam com recursos que são `h:outputScript`, `h:outputStylesheet` e `h:graphicImage`. Em comum nesses três componentes temos basicamente duas propriedades: `library` e `name`. (CORDEIRO, Aplicações Java para web com JSF e JPA. 2010, p. 167).

5.1.5.2.3 Primefaces

Figura 8 - Logo Primefaces



Fonte: Primefaces. Disponível em: <<http://www.primefaces.org>>. Acesso em: 24 maio 2014.

Atualmente existem vários frameworks com componentes visuais prontos para uso em aplicações, dentre eles podemos destacar três: o Primefaces,

RichFaces e IceFaces, todos eles são Open Source.

A escolha pelo Primefaces foi com base na comparação feita pelo site mastertheboss.com, onde o Primefaces foi o que teve melhor pontuação em diversos quesitos, outros fatores que contribuíram para essa escolha foi que ele possui uma UI (User Interface) agradável que pode ser customizada com uso de CSS, por possuir uma grande quantidade de componentes, ter uma documentação com exemplos e melhores práticas para cada um dos componentes disponíveis, descrevendo os atributos que podem ser usados por esses.

É considerado por muitos o mais avançado do mercado. Ele tem diversas funcionalidades prontas que facilitam e muito a vida do desenvolvedor. Possui mais de 130 componentes em seu showcase com código que explica como utilizar. (COELHO, Hébert. JSF Eficaz As melhores práticas para o desenvolvedor web Java, 2010, p. 93).

5.1.5.3 Controller

O Controller trabalha como intermediador entre as camadas View e Model. As requisições são geradas na camada View através de solicitações feitas por usuários e para atender essas requisições, o Controller realiza consultas na camada Model que ao terminar apresenta o resultado solicitado em uma página web.

O Controller (Controlador) é o objeto que responde as ordens executadas pelo usuário, atuando sobre os dados apresentados pelo modelo, decidindo como o Modelo deverá ser alterado ou deverá ser revisto e qual Apresentação deverá ser exibida. (GONÇALVES, Edson. Desenvolvendo Aplicações Web..., 2007, p. 386).

5.1.6 Banco de dados

Para que uma aplicação seja consistente faz-se necessário salvar os dados inseridos pelos usuários de alguma maneira. Esses dados podem ser salvos em arquivos de texto e/ou um banco de dados.

A necessidade de se salvar essas informações são para que posteriormente possam ser consultadas, alteradas e caso seja necessário deletadas.

5.1.6.1 MySQL

Figura 9 - Logo MySql



Fonte: MySQL. Disponível em: <<http://www.mysql.com/about/legal/logos.html>>. Acesso em: 25 maio 2014.

O MySQL é um sistema de gerenciamento de banco de dados (SGBD) relacional que é muito utilizado em aplicações web pelo seu desempenho, flexibilidade, capacidade de armazenamento e escalabilidade. Seus comandos são feitos utilizando a sintaxe SQL (Structured Query Language). O MySQL tem licença GNU General Public Licence (GPL), indicando que ele pode ser utilizado gratuitamente. Segundo LUCKOW e MELO (2010, p. 63) “Para que o MySQL possa ser utilizado gratuitamente, o aplicativo que o utiliza tem que seguir a licença GPL.”.

O MySQL segundo seu próprio site “dev.mysql.com” utiliza o Modelo ACID (Atomicidade - Atomicity, Consistência - Consistency, Isolamento - Isolation e Durabilidade - Durability), essas características são primordiais para que um banco de dados seja funcional, caso houver falha em alguma dessas características o dado não é confiável, pois não há garantia de integridade das informações. Para controlar os acessos, MySQL tem a opção de controle de acesso por usuários e/ou por grupos.

5.1.7 ORM (Object Relational Mapper – Mapeamento Objeto Relacional)

O ORM é utilizado para conversão de dados, essa conversão pode ser realizada de duas formas, a primeira é transformar objetos em dados relacionais para serem persistidos no banco de dados e a segunda é recuperar os dados que estão no banco de dados e transformá-los em objetos. Essa conversão é necessária devido os dados terem a estrutura diferente. Existem diversos frameworks que fazem essa conversão de dados, devido a esse motivo surgiu a especificação Java

Persistence API (JPA) para padronizar essas ferramentas ORM no desenvolvimento de aplicações Java, o JPA é uma especificação EJB, com as regras de como esses frameworks de ORM devem trabalhar, não é implementado nenhum código por parte dela.

Object Relational Mapping (ORM) quer dizer basicamente: “transformar as informações de um banco de dados, que estão no modelo relacional para classes Java, no paradigma Orientado a Objetos de um modo fácil”. Um framework ORM vai ajudá-lo na hora de realizar consultas, manipular dados e até mesmo a retirar relatórios complexos. A verbosidade de um SQL com JDBC não será necessária, nem mesmo os incansáveis loops que fazemos para popular objetos. (COELHO, Hébert. JPA Eficaz As melhores práticas de persistência de dados em Java. São Paulo: Casa do Código, 2010, p. 3).

Abaixo é apresentada uma tabela comparativa entre Modelo Relacional e Modelo Orientado ao Objeto.

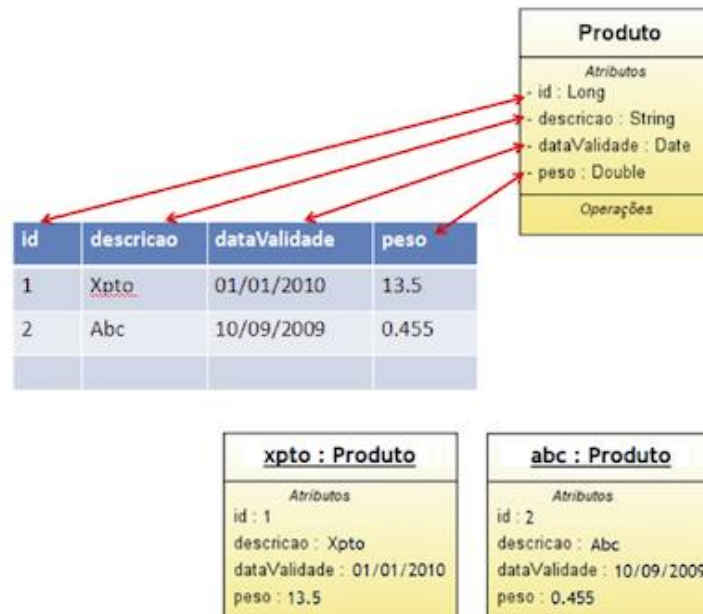
Tabela 1 - Modelo Relacional e Modelo Orientado ao Objeto

Modelo Relacional	Modelo OO
Tabela	Classe
Linha	Objeto
Coluna	Atributo
-	Método
Chave Estrangeira	Associação

Fonte: ANDRADE (2013, p. 31)

Na figura 10 é representado a conversão realizada pelo ORM, onde ele transforma as duas tuplas do modelo relacional em instancias de objetos. Nessa mesma figura também é possível visualizar a relação entre os atributos da classe Java com os atributos que estão na entidade do banco de dados.

Figura 10 - Relação dos atributos do Modelo Relacional com o Modelo Orientado ao Objeto



Fonte: (SAKURAI. JPA 2.0 – Descrever os conceitos básicos do Mapeamento Objeto Relacional (ORM). Disponível em: <<http://www.universidadejava.com.br/docs/jpa20-descreverosconceitosbasicosdomapeamentoobjetorelacionalorm>>. Acesso em: 11 jun 2014.).

Atualmente grandes empresas estão utilizando essa tecnologia devido a facilidade e velocidade no desenvolvimento, praticamente pode-se desenvolver aplicações sem ter que digitar comandos SQL, bastando somente criarem o banco de dados que posteriormente o framework que utiliza o JPA constrói as entidades automaticamente. Para usufruir dessa tecnologia deve-se fazer declaração das tags do framework escolhido no seu código que a ferramenta ORM realiza a codificação a de forma transparente.

5.1.7.1 Hibernate

Figura 11 - Logo Hibernate



Fonte: Hibernate. Disponível em: <<http://hibernate.org>>. Acesso em: 25 maio 2014.

Como framework ORM, será utilizado o framework Hibernate ele

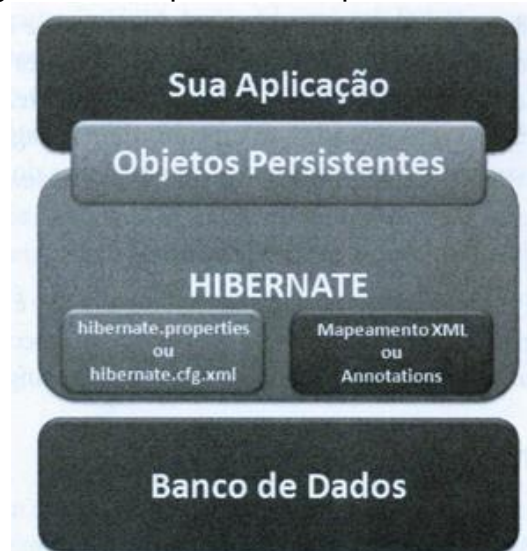
segue a implementação do JPA. Ele surgiu antes mesmo que JPA e foi criado pelo Gavin King, foi devido ao seu sucesso que deu origem a especificação JPA.

O uso do Hibernate é demasiadamente simples, para sua utilização devemos inserir seus jars na pasta de nossa aplicação ou declarar suas dependências no arquivo POM.XML e posteriormente, declarar as configurações que serão utilizadas pelo Hibernate e mapear os atributos nas classes.

O Hibernate contém diversas funcionalidades como Engenharia Reversa, Exportação de Schema facilitado e anotações próprias para simplificar a utilização da API. (COELHO, Hébert. JPA Eficaz As melhores práticas de persistência de dados em Java. São Paulo: Casa do Código, 2010, p. 5).

Segundo LUCKOW e MELO (2010. p. 122) “Apesar de ter tantas ferramentas e classes compondo seu núcleo, a arquitetura do Hibernate pode ser simplificada.” A figura 12 ilustra a arquitetura simplificada do Hibernate, onde na camada do Hibernate é apresentado duas formas de mapeamento de uma classe Java, uma via XML e outra através de Annotations. Não é recomendado utilizar os dois ao mesmo tempo devido a problemas de conflito e redundância de execução.

Figura 12 - Arquitetura simplificada do Hibernate



Fonte: (LUCKOW, MELO. Programação Java para a Web. 2010, p. 123).

O mapeamento realizado por Annotations segue o padrão Enterprise Java Beans (EJB), com a utilização das Annotations as configurações dos metadados ficam centralizadas nas classes Java, caso fossem utilizados arquivos XML para os mapeamentos seriam necessários a criação de arquivo XML para cada

classe mapeada, devido a esse fato foi optado a criação da aplicação utilizando Annotations. Segundo GONÇALVES (2007. p. 525) “Com anotações, o Hibernate fica bem mais simples e você acaba usando menos o XML, em relação ao mapeamento de tabelas. ”.

Para validação dos campos da aplicação em alguns casos foi empregado o uso do Hibernate Validator, que faz as validações necessárias antes de persistir os dados. Sua utilização através de Annotations é simples, basta inserir as bibliotecas necessárias na aplicação e declarar a anotação no atributo em que se deseja utiliza-lo. Com a utilização dessas anotações a validação de CPF, CNPJ, e-mail entre outros se tornou mais simples, pois não é mais necessário criar scripts em Javascript para validações em formulários.

5.1.7.2 DAO – Data Access Object

O Data Access Object é responsável por fazer acesso ao banco de dados, ela é uma classe que possui métodos CRUD (Create, Read, Update e Delete) relacionados ao objeto.

Sempre que você precisa acessar um banco de dados que está mantendo seu modelo de objetos, é melhor empregar o padrão DAO. O Padrão DAO fornece uma interface independente, no qual você pode usar para persistir objetos de dados. A ideia é colocar todas as funcionalidades encontradas no desenvolvimento de acesso e trabalho com dados em um só local, tomando simples sua manutenção. Tipicamente um DAO inclui métodos para inserir, selecionar, atualizar e excluir objetos de um banco de dados. Dependendo de como você implementa o padrão DAO, você poderá ter um DAO para cada classe de objetos em sua aplicação ou poderá ter um único DAO que é responsável por todos os seus objetos. (GONÇALVES, Edson. Desenvolvendo Aplicações Web..., 2007, p. 399).

Utilizando o DAO podemos separa as operações de banco de dados em um local específico e quando for necessário basta chamar o método dessa classe.

Para comunicação entre o Hibernate e o Banco de dados, devemos utilizar o arquivo Persistence.xml, nesse arquivo são declaradas as url's de conexão com o banco de dados, usuário e senha para concretização dessa conexão, bem como o dialeto utilizado para comunicação com o banco de dados. Esse arquivo fica dentro da pasta META-INF por padrão.

Segundo COELHO (JPA Eficaz As melhores práticas... 2010. p. 7)

“O arquivo Persistence.xml é o ponto de partida de qualquer aplicação que use a JPA.”. Isso porque esse arquivo possui as configurações do banco que serão utilizados pela a aplicação, caso mude o banco de dados basta apenas realizar alterações nesse arquivo e o programa já está pronto para ser utilizado.

5.1.8 Segurança do Sistema

A proteção de sistemas de informação tem se tornado cada vez mais vital para diversos seguimentos de empresas, a exposição dessas informações deve ser limitada apenas para usuários autorizados.

O acesso a tais informações deve ser entregue aos usuários através de algum meio de autenticação, seja ela utilizando login e senha ou fazendo-se uso de outros meios que venha por ventura comprovar a autenticidade do usuário que pretende realizar o acesso.

Posteriormente após a liberação do acesso ao sistema para o usuário, entra outra questão relativa a autorizações de acesso que a credencial permite. Essa credencial é referente ao que o usuário pode acessar. Ela pode ser feita dividindo o sistema em grupos ou ainda delegando permissões para os usuários específicos.

São características básicas da segurança da informação os atributos de confidencialidade, integridade, disponibilidade e autenticidade, não estando esta segurança restrita somente a sistemas computacionais, informações eletrônicas ou sistemas de armazenamento. (Disponível em:< http://pt.wikipedia.org/wiki/Segurança_da_informação>. Acesso em 05 set. 2014).

Outro conjunto de informações associadas às credenciais são suas permissões de acesso, normalmente chamadas de papéis(roles) ou autoridade (authorities), que informam ao sistema quais os limites de atuação do indivíduo. O processo de verificação das permissões do usuário é a autorização, que ocorre sempre após a autenticação no sistema. (WEISSMANN, 2012, p. 228).

5.1.8.1 Spring Security

O Spring Security é um framework de controle de acesso, ele faz parte do Spring Framework e pode ser utilizado separadamente em outras aplicações Java EE, não sendo necessário que seja uma aplicação desenvolvida com o framework Spring.

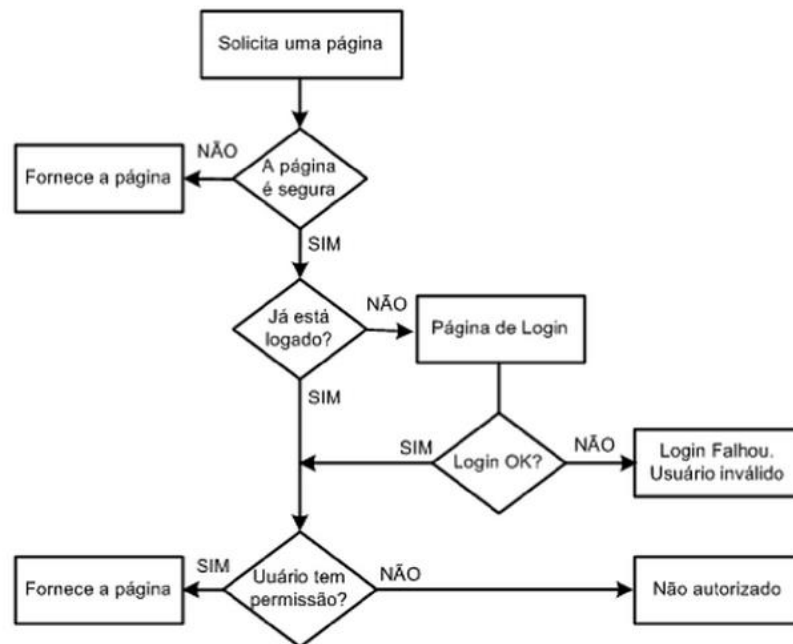
Sua utilização consiste em baixar as bibliotecas necessárias e inseri-las na aplicação, como esse projeto utiliza o Maven, basta apenas inserir suas tags específicas no arquivo POM.XML que o mesmo se encarrega de fazer o download do Spring Security e suas dependências. Precisaremos incluir alguns ouvintes e filtros no arquivo Web.xml, para que o mesmo funcione a contento.

Sua configuração é feita em um arquivo chamado applicationContext.xml, nele é possível declarar os acessos na aplicação, concedendo permissões para usuários ou grupo de usuários. Para WEISSMANN (2012, p. 227), “O Spring Security é um framework de controle de acesso, que nos permite definir de forma declarativa quem acessara o que em nossos sistemas.”.

Ele cria uma página de login automaticamente, mas caso o desenvolvedor queira pode criar uma outra página, bastando utilizar os nomes dos campos pertinentes ao Spring Security.

O direcionamento de acessos do Spring Security é realizado da seguinte forma como a Figura 13, simplesmente verifica se tem acesso e se tem permissão para efetuar a alteração desejada, caso não tenha permissão ou não possua acesso é mostrada uma mensagem de acesso negado.

Figura 13 - Controle de Acesso Spring Security



Fonte: (LUCKOW, MELO. Programação Java para a Web. 2010, p. 234).

Sempre que alguém tentar acessar alguma pasta do sistema que tenha segurança garantida pelo Spring Security, será apresentada automaticamente uma tela de login para que o visitante se identifique. (LUCKOW, Décio Heinzemann; MELO, Alexandre Altair. Programação Java para a Web. São Paulo: Novatec Editora, 2010, p. 234).

5.1.9 Ireports e Jasper Reports

Para geração dos relatórios, foi utilizado o Jasper Reports como framework de geração dos relatórios, esse framework possibilita a geração de arquivos pdf, html, xls e etc.

JasperReports: essa biblioteca é o motor que transforma o arquivo JRMXL em um relatório exportado para uma das extensões suportadas (PDF, HTML etc.) usando uma fonte de dados que foi especificada no arquivo. (LUCKOW, Décio Heinzemann; MELO, Alexandre Altair. Programação Java para a Web. São Paulo: Novatec Editora, 2010, p. 494).

O Ireports é um programa gráfico que possibilita a criação de relatórios de forma simplificada, bastando apenas realizar a conexão com a base de dados (texto, csv ou banco de dados) e arrastar os componentes para a criação dos

relatórios.

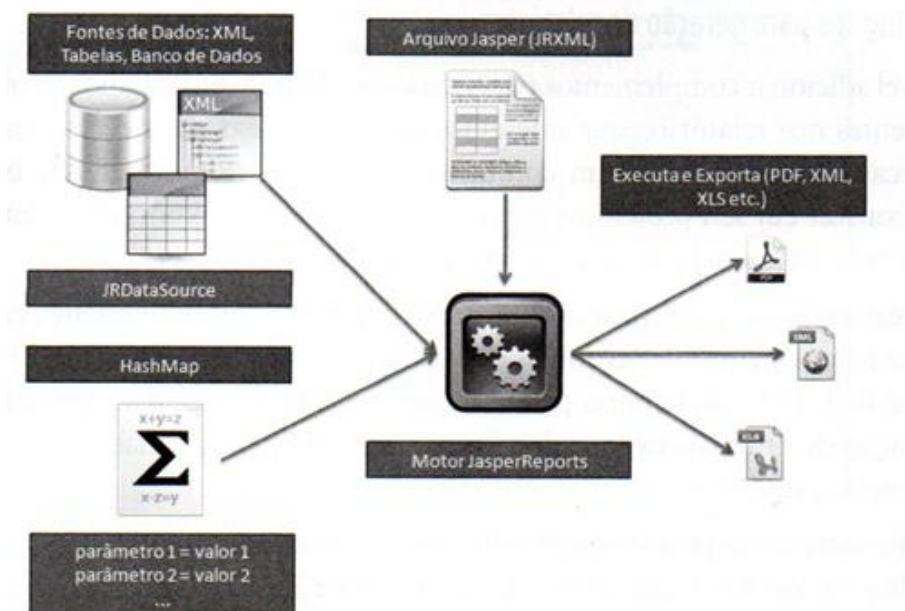
IReport: é uma ferramenta visual. É a partir dela, em um ambiente totalmente estilo clique e arraste, que podemos criar os arquivos XML(JRXML) necessários ao JasperReports. Na ferramenta é possível inserir tipos de objetos em nossos relatórios, como textos estáticos, figuras geométricas, imagens, gráficos, sub-relatorios e grupos de dados, entre outros. (LUCKOW, Décio Heinzemann; MELO, Alexandre Altair. Programação Java para a Web. São Paulo: Novatec Editora, 2010, p. 494).

Tanto o Jasper Reports e o Ireports são open source e são desenvolvidos pela mesma empresa a JasperForge, sua licença é baseada no modelo LGPL que deixa a cargo do desenvolvedor o código fonte que ele deseja compartilhar para a comunidade de desenvolvedores.

Os arquivos gerados pela ferramenta Ireports possuem a extensão JRXML, ele é necessário para o Jasper Reports transforme os dados em relatórios.

Na figura 14 é apresentado de forma simplificada a execução de um relatório e sua arquitetura de ferramentas.

Figura 14 - Fluxo de Execução de um Relatório



Fonte: (LUCKOW, MELO. Programação Java para a Web. 2010, p. 495).

6 LEVANTAMENTO DE REQUISITOS

Como forma de levantamento de requisitos foi utilizada a técnica de formulário com questões apresentado no Apêndice A, foram entrevistados quatro estabelecimentos que trabalham com hospedagem de clientes.

6.1 DESCRIÇÃO DO SISTEMA EXISTENTE

As páginas dos estabelecimentos visitados apresentam apenas conteúdos estáticos que dificultavam a manutenção das tarifas cobradas pelo estabelecimento.

Não era possível aos clientes saber os valores corretos praticados pelos estabelecimentos e se o mesmo possuía quartos disponíveis para reservas.

6.2 DESCRIÇÃO DO SISTEMA PROPOSTO

O sistema aqui proposto permite ao estabelecimento o total controle do valor das tarifas cobradas por apartamento. Possibilita ao hospede consultar o valor das tarifas cobradas pelo estabelecimento e a disponibilidade de quartos para reserva.

O funcionário do estabelecimento podem cadastrar os valores dos quartos e efetuar reservas dos mesmos para os clientes e consultar se existem quartos disponíveis para alocação em diferentes datas.

Os clientes podem realizar seu pré-cadastro no site, para posteriormente serem validados pelo estabelecimento.

6.3 REQUISITOS FUNCIONAIS

O sistema deverá disponibilizar aos usuários administrativos, uma tela com os tipos de acomodações existentes no hotel e seus respectivos valores após uma consulta no banco de dados.

O sistema deverá gravar todas as informações dos hospedes, reservas e apartamentos no banco de dados. Cada uma dessas informações deverá possuir um identificador único na sua respectiva tabela.

O sistema não permitirá cadastro de menores de 18 anos, segundo

a Lei Nº 8.069, de 13 de julho de 1990, que Dispõe sobre o Estatuto da Criança e do Adolescente e dá outras providências. Em seu art. 82 “É proibida a hospedagem de crianças ou adolescentes em hotel, motel, pensão ou estabelecimento congênere, salvo se autorizado ou acompanhado pelos pais ou responsáveis”. Por esse motivo o sistema proposto, obriga ao hospede informar a sua data de nascimento, tendo como escolhas disponíveis apenas datas em que se é possível ser maior de idade.

6.3.1 Os Atores e suas hierarquias no Sistema

Para garantir a segurança do sistema, os usuários foram divididos em categorias, isso se fez necessário para manter a integridade do sistema e possibilidade de delegar acessos e autorizações.

6.3.1.1 O Administrador

É o principal usuário do sistema, esse usuário pode inserir, atualizar, alterar e excluir hospedes, apartamentos e reservas, sempre que haja alguma necessidade. Para isso o mesmo terá uma página com um menu diferenciado, para que haja um maior controle das permissões concedidas.

O sistema proporciona as seguintes funcionalidades nesse perfil:

- Hospedes (incluir, excluir, listar, atualizar e selecionar)
- Apartamentos (incluir, excluir, listar, atualizar e selecionar)
- Reservas (incluir, excluir, listar, atualizar e selecionar)

6.3.1.2 O Hospede

É o usuário que possui cadastro no site, esse cadastro pode ser feito pelo Administrador ou pelo próprio usuário, com essa permissão o usuário pode efetuar reservas de quartos.

O sistema proporciona as seguintes menu de funcionalidades nesse perfil:

- Hospedes (incluir e atualizar)
- Apartamentos (listar)

- Reservas (incluir e selecionar)

6.3.1.3 O Visitante

É o usuário que apenas navega no site, tem permissão apenas para visualização dos valores praticados pelos usuarios

6.4 REQUISITOS NÃO FUNCIONAIS

O sistema apresentado ficará disponível para acesso via web, onde sua hospedagem utiliza o servidor Tomcat, Banco de dados MySql e suporte à programação Java e seus frameworks como Hibernate e o Primefaces.

6.4.1 Do cadastro

O sistema efetuará a inserção de registros em sua entidade específica com todos os campos solicitados no momento do cadastro.

O registro inserido possuirá um identificador único que o diferenciará dos demais registros evitando inconsistência dos dados. Dessa forma, todas informações relacionadas ao relacionamento entre o hospede e sua reserva serão precisas, não possibilitando que reservas efetuadas por um hospede seja registrada em nome de outros hospedes.

O sistema permitirá que apenas o usuário Administrador tenha acesso ao painel de administrativo, onde é possível realizar operações referentes aos Apartamentos, Hospedes e Reservas. Ao usuário hospede só é concedido a permissão de visualização de apartamentos disponíveis para reserva. Alterações de dados como endereço, telefone devem ser solicitados ao Administrador, isso devido a segurança onde caso haja algum mal-entendido entre as partes o hospede mude seu endereço para não ser localizado. O hospede somente poderá efetuar troca de senhas caso o mesmo venha esquecer.

6.4.2 Da exclusão e alteração de: Apartamentos, hospedes e reservas

O sistema permitirá que apenas o usuário Administrador tenha acesso ao painel de gerencia, onde é possível realizar operações referente a inclusão, exclusão e alteração dos Apartamentos, Hospedes e Reservas.

6.5 VALIDAÇÃO DE REQUISITOS

A validação dos requisitos deve ser aplicada durante todo o desenvolvimento da aplicação, uma vez que se um requisito mal formulado for detectado é possível corrigi-lo antes do termino da aplicação, se tal validação for tardia, sua correção pode acarretar em sérios problemas no sistema e grandes manutenções para se chegar no resultado pretendido.

Como forma de validação de requisitos é comumente utilizado as Checklists (Lista de Validação) contendo perguntas precisas referente as solicitações realizadas pelo Stakeholders. Essas questões facilitam a localização de erros e ajudam o desenvolvedor a manter o foco no que lhe foi solicitado desde o princípio até a sua finalização.

A validação de requisitos é o processo pelo qual se verifica se os requisitos definem o sistema que o cliente realmente quer. Ela se sobrepõe à análise, uma vez que está preocupada em encontrar problemas com os requisitos. A validação de requisitos é importante porque erros em um documento de requisitos podem gerar altos custos de retrabalho quando descobertos durante o desenvolvimento ou após o sistema já estar em serviço. (SOMMERVILLE, 2011, p. 76).

Algumas técnicas de validação são comumente utilizadas durante a etapa de validação de requisitos, elas são a validação dos requisitos, a prototipação de telas do sistema e geração de estudo de caso para saber se o sistema proposto irá atender realmente as necessidades apresentadas pelo solicitante.

6.6 GERENCIAMENTO DE REQUISITOS UTILIZADOS

O Gerenciamento de Requisitos é um processo sistemático para analisar, encontrar, validar e documentar os requisitos de um sistema para validação da funcionalidade proposta pelo sistema informatizado.

Para gerenciar os requisitos da aplicação proposta foi utilizado casos de teste para simular um sistema em produção real, com base nesses casos o sistema foi sendo moldado até chegar o mais próximo do sistema proposto nesse trabalho de conclusão de curso.

Outra forma de validação foi a apresentação de telas como protótipos do sistema, estes proporcionavam aos entrevistados uma real ideia do sistema para verificar se o mesmo era satisfatório.

Na figura 15 apresentada abaixo é possível verificar como será uma das telas da aplicação, é apresentada a tela de cadastro de apartamento e suas possíveis mensagens de interação com o usuário. Esse protótipo de tela foi concebido utilizando o programa “Balsamiq Mockups For Desktop”, essa ferramenta facilita a construção de protótipos para a demonstração de como será a tela e melhora o levantamento dos requisitos que a página deverá atender.

Figura 15 - Protótipo da Tela de Cadastro de Apartamento



O protótipo da tela de cadastro de apartamento é exibido em um navegador web. O endereço da URL na barra de endereços é `http://reservas.fabiobpinto.cloudbees.net/restrito/cadastroApartamento.xhtml`. O cabeçalho da página contém o título "Hotel" e uma barra de navegação com links para "Home", "Link", "Link", "Link" e um menu "Dropdown". O título principal da seção é "Cadastro de Apartamentos".

À esquerda, há campos de entrada para:

- Número:*
- Tipo: (menu suspenso com "Tipo" selecionado)
- Cama: (menu suspenso com "Cama" selecionado)
- Valor:*

Abaixo do campo "Valor:*" há um botão "Salvar".

À direita, há duas mensagens de feedback:

- Uma mensagem de erro em um fundo rosa: "Já existe o apartamento 60, ele não pode ser cadastrado."
- Uma mensagem de sucesso em um fundo azul: "Apartamento 560 cadastrado com Sucesso."

Para validação do sistema proposto, a aplicação foi apresentada em estabelecimentos que prestam esse tipo de serviços e avaliados quanto as funcionalidades propostas no que se diz respeito ao cadastro de apartamentos, pré-cadastro dos hospedes e reserva de quartos.

Um protótipo é uma versão inicial de um sistema de software, usado para demonstrar conceitos, experimentar opções de projeto e descobrir mais sobre o problema e suas possíveis soluções. (SOMMERVILLE, 2011, p. 31).

7 DIAGRAMA DE CASO DE USO

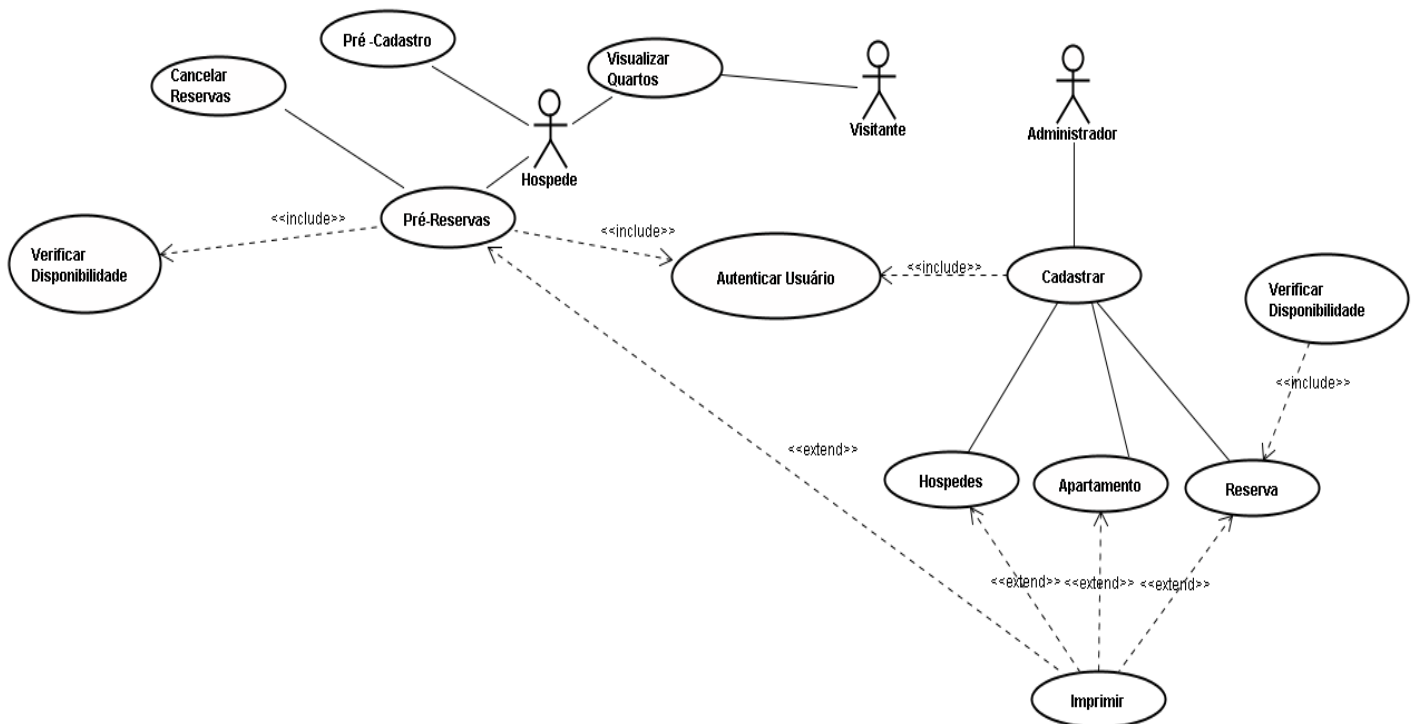
O diagrama de Caso de Uso é utilizado para demonstrar o comportamento do sistema para pessoas de uma forma menos técnica, ele demonstra a relação entre as funções que o sistema terá com os atores e quais possíveis dependências elas podem vir a ter.

Devido a esse diagrama não possuir detalhes técnicos é indicado sua utilização para exemplificar para clientes quais são as funcionalidades de um sistema na perspectiva de um utilizador do sistema.

De acordo com Ramos (2006, p. 66) “Um caso de uso deve descrever o que faz um sistema (ou parte dele), e não como ele é realizado. O foco é, portanto, na visão externa do sistema, ou seja, na visão que os usuários têm dele.”. Sua utilização é a melhor forma de se iniciar a captação de requisitos que um sistema poderá ter, pois sua diagramação é de fácil entendimento para os usuários não técnicos.

Abaixo na Figura 16 é apresentado o Diagrama de Caso de uso do trabalho proposto de reserva de quartos:

Figura 16 - Diagrama de Caso de Uso



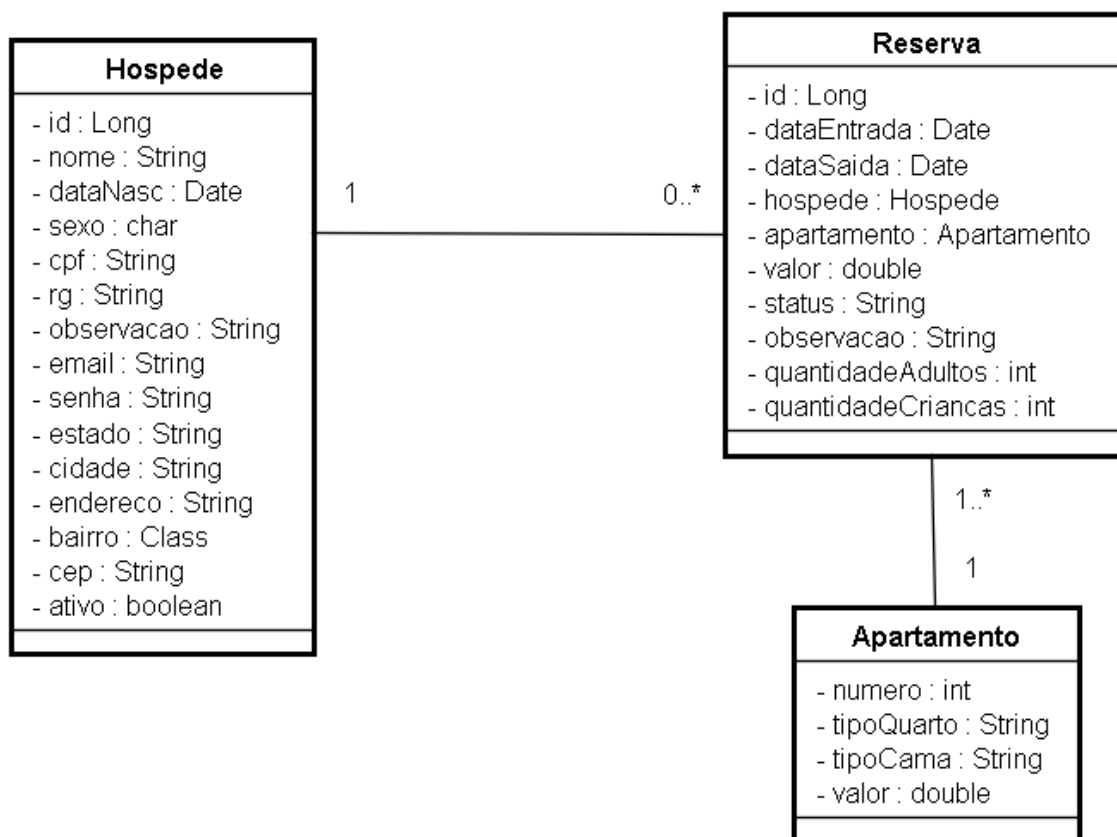
8 DIAGRAMA DE CLASSES

O diagrama de classe especifica quais objetos estão sendo empregados na estruturação da aplicação, com esse diagrama é possível levantar quais os atributos e métodos pertencentes a um objeto e quais os tipos de relações existente entre esses objetos eles.

De acordo com Ramos (2006, p. 49) “Os diagramas de classes descrevem a estrutura estática de um sistema, em particular as entidades existentes, suas estruturas internas e relações entre si.”.

Abaixo na Figura é possível verificar a relação entre as entidades Reserva para com as entidades Hospede e Apartamento, onde em uma Reserva tem apenas um hospede e um Apartamento pode ser reservado diversas vezes desde que o mesmo esteja disponível para tal.

Figura 17 - Diagrama de Classe



9 TELAS (PROTOTIPAÇÃO)

A seguir é apresentado algumas das principais telas do Sistema de Reservas de Quartos da aplicação.

9.1 TELA DE LOGIN

Para se ter acesso as telas administrativas e restritas, se faz necessário a solicitação do e-mail e a senha do usuário para conceder a permissão de acesso. Os hóspedes por padrão recebem a autorização de ROLE_USER, já os usuários administradores devem ter sua permissão alteradas diretamente no banco para ROLE_ADMIN.

No sistema proposto existem duas telas de login, uma se localiza na barra de navegação do site e a outra é apresentada caso o usuário tente acessar uma página específica digitando a URL diretamente no navegador.

Abaixo na Figuras 18 e na Figura 19 são apresentadas as telas de login, acessada via menu menu e de acesso via URL respectivamente.

Figura 18 - Tela de Login acessada via Menu



The image shows a web application interface with a dark header containing 'Contato' and 'Login' links. A white modal window titled 'Login' is centered on the screen. Inside the modal, there are two input fields: 'E-mail: *' and 'Senha: *'. Below these fields are two buttons: 'Fechar' (white with a grey border) and 'Login' (solid blue). At the bottom right of the modal, there are two links: 'Cadastrar-se' and 'Esqueci minha senha', both in blue text. The background of the application is a light grey with a faint image of a hotel room. At the bottom of the page, below the modal, the text 'Sistema de Reserva de Quartos.' and 'Fábio Brito Pinto - ead01398352.' is visible.

Figura 19 - Tela de Login via URL



Sistema de Reserva de Quartos.
Fábio Brito Pinto - ead01398352.

Login

E-mail: *

Senha: *

Login

Em todas as telas o acesso é liberado buscando os dados de acesso no banco de dados, caso seja digitada um usuário e senha errado, é apresentada uma mensagem de erro de login, conforme a Figura 20.

Figura 20 - Tela de Erro de Login

Login

Usuário ou senha incorretos!

E-mail: *

Senha: *

Login

9.2 TELAS DE CADASTRO E EDIÇÃO

Tela de cadastro e Edição de Apartamento na Figura 21. O que diferencia uma tela da outra é que se clicar no botão de editar já traz os dados preenchidos para serem alterados.

Figura 21 - Tela de cadastro e Edição de Apartamento

Cadastro de Apartamento	
<input type="text"/> <input type="button" value="Pesquisa"/>	
Id.:	<input type="text"/>
Numero: *	<input type="text"/>
Tipo	<input type="text" value="Selecione"/>
Cama	<input type="text" value="Selecione"/>
Valor: *	<input type="text" value="R\$ 0,00"/>
<input type="button" value="Salvar"/>	




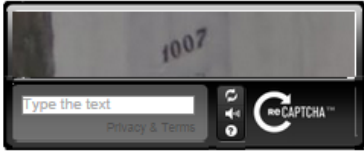
Sistema de Reserva de Quartos.
Fábio Brito Pinto - ead01398352.

Essa tela apresenta no seu lado direito o botão de pesquisa que leva o usuário a uma página com todos os apartamentos cadastrados.

São validados se foi informado o Número e valor do Apartamento, caso não sejam preenchidos é apresentado uma tela informando o erro.

Tela de Cadastro de Hospede apresentada na Figura 22, existem duas telas com as mesmas funções porem a tela do administrador possui um botão de Ativo que deve ser acionado após conferencias dos documentos do hospede e na tela onde o hospede insere seus dados para cadastro possui um captcha para confirmação do cadastro.

Figura 22 - Tela de Cadastro de Hospede














Hotel Início Quartos Contato Login			
Cadastro de Hospede			
Nome: *	<input type="text"/>	Data de nascimento *	<input type="text"/> 
Sexo:	<input type="button" value="Masculino"/> <input type="button" value="Feminino"/>	CPF: *	<input type="text"/>
RG: *	<input type="text"/>	Observação:	<div><div></div></div>
Tel: *	<input type="text"/>	E-mail *	<input type="text"/>
Senha: *	<input type="password"/>	Confirmar a Senha: *	<input type="password"/>
Estado:	Selecione ▼	Cidade:	<input type="text"/>
Endereço:	<input type="text"/>	Bairro:	<input type="text"/>
CEP:	<input type="text"/>	Código de Segurança:	
			<input type="button" value="Salvar"/>

Na tela de edição de cadastro o administrador não precisa preencher o código de segurança e nem informar a senha do hospede.

9.3 TELAS DE LISTAGEM E EXCLUSÃO

A tela de Listagem dos Apartamentos cadastrados é apresentada na Figura 23, essa tela oferece os botões de edição e exclusão do Apartamento, é possível ordenar a tabela clicando nos títulos das colunas, no rodapé da tabela existem três opções para exportar os apartamentos, os formatos são pdf, csv e xls, é gerado um arquivo respectivo apenas com as linhas que estão visíveis na tela.



















Figura 23 - A tela de Listagem dos Apartamentos cadastrados

Hotel Apartamentos Hospedes Reservas Relatórios Sair					
Listagem de Apartamento					
Novo					
Lista de Apartamentos					
(1 of 3) 1 2 3 5					
Id.	Número	Tipo	Cama	Valor	Ação
101	01	Duplo	Solteiro	R\$ 160,00	 
97	02	Duplo	Casal	R\$ 10,00	 
95	03	Duplo	Solteiro	R\$ 30,00	 
96	04	Duplo	Solteiro	R\$ 40,00	 
103	05	Suíte	Casal	R\$ 780,00	 
(1 of 3) 1 2 3 5					
Existem 12 apartamentos cadastrados.					
  					

Caso o usuário clique no botão de Excluir é apresentada uma tela solicitando a confirmação da exclusão ou clique no botão de Editar é redirecionado para a tela de cadastro com os campos preenchidos do Apartamento que será alterado. Há possibilidade de se incluir um novo apartamento clicando no botão Novo.

Na tela de listagem de hospedes na Figura 24 possui três botões nas ações, são eles respectivamente: visualizar os dados do hospede, editar as informações do hospede e excluir o hospede. Podemos realizar filtros nas pesquisas com os campos de nome e CPF, ou ainda ordenar por e-mail a lista de hospedes cadastrados. Se preferir podemos exportar a lista dos hospedes em PDF, CSV e XLS.

Figura 24 - Tela de Listagem de Hospede

Listagem de Hospedes				
Novo				
Lista de Hospedes				
(1 of 2) 1 2 10				
Id	Nome	E-mail	CPF	Ação
14	Carlos Meneses	carlos.meneses@gmail.com	287.470.647-73	  
18	Claudio Lopes	claudio@gmail.com	513.365.635-16	  
13	Fábio Brito Pinto	fabio@bol.com.br	068.878.236-16	  
16	Jairo Bernardes	jairo@hotmail.com	372.225.436-19	  
15	Maria Cecilia	mariaceci@gmail.com	034.832.775-79	  
(1 of 2) 1 2 10				
Existem 6 hospedes cadastrados.				
  				

A escolha por inserção do botão de visualizar os dados do hospede foi devido a quantidade de informações que o hospede possui, se colocasse todos os campos na tabela de listagem algumas informações ficariam ilegíveis, na Figura 25 é mostrada a tela com os detalhes do hospede selecionado, temos nessa tela a opção de impressão dos dados do cliente apresentado.

Figura 25 - Tela com Informações do Hospede

			
Id:	14	Nome:	Carlos Meneses
Data Nasc.	19/10/1990	Sexo	M
CPF	287.470.647-73	RG	MG 23.989.229
Observação	Sou fumante	Telefone	(35)9999-9999
e-mail	carlos.meneses@gmail.com	Endereço	Rua 7 de Setembro
Bairro	Centro	Cidade	Virginia
CEP	83.778-377	Estado	MG
Ativo	😊		

(1 of 2) 1 2 10

Existem 6 hospedes cadastrados.

A escolha por inserção do botão de visualizar os dados do hospede foi devido a quantidade de informações que o hospede possui, se colocasse todos os campos na tabela de listagem algumas informações ficariam ilegíveis, na Figura 25 é mostrada a tela com os detalhes do hospede selecionado, temos nessa tela a opção de impressão dos dados do cliente apresentado.

Na figura 26 é apresentada a tela onde o administrador efetua a reserva, é selecionada as datas de Check-in e Check-out o apartamento disponível nessa data e o hospede que está efetuando a reserva, alguma observação referente a hospedagem, em Status é possível cancelar, deixar reservado e colocar como concluído, informar a quantidade de adultos e crianças, o valor é calculado automaticamente, mas se preciso o usuário que está cadastrando a reserva pode mudar o valor.

Figura 26 - Tela de Reserva

Hotel

Início

Apartamentos

Hospedes

Reservas

Relatórios

Sair

Cadastro de Reservas

Pesquisa

Id.:	<input type="text"/>	Data do Check-in *	<input type="text" value="___/___/___"/>
Data de Check-out *	<input type="text"/>	Apartamento *	Selecione o apartamento
Hospede *	Selecione o hospede	Observação:	<div></div>
Status:	Selecione	Quantidade de Adultos: *	<input type="text" value="0"/>
Quantidade de Crianças: *	<input type="text" value="0"/>	Valor: *	<input type="text"/>
Salvar			

10 CÓDIGO FONTE

A seguir é apresentado alguns dos principais códigos fontes do Sistema de Reservas de Quartos da aplicação.

10.1 CÓDIGO FONTE SQL

Código fonte gerado pelo MySQL Workbench ao exportar o arquivo de backup do banco de dados, esse arquivo apresenta a estrutura do banco de dados, com alguns registros incluídos no banco pelo sistema, como a aplicação utiliza o Hibernate para persistência no banco, ele gera as query's necessárias automaticamente.

Comando para criar a base de dados:

```
CREATE DATABASE IF NOT EXISTS `hospedagem` ;
```

Comandos gerados pelo arquivo de exportação do MySQL Workbench 6.0:

```
CREATE DATABASE IF NOT EXISTS `hospedagem` /*!40100
DEFAULT CHARACTER SET utf8 */;
```

```
USE `hospedagem`;
```

```
DROP TABLE IF EXISTS `apartamento`;
```

```
CREATE TABLE `apartamento` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `cama` varchar(255) DEFAULT NULL,
  `numero` varchar(5) NOT NULL,
  `tipo` varchar(255) DEFAULT NULL,
  `valor` double NOT NULL,
```

```
PRIMARY KEY (`id`),
```

```
UNIQUE KEY `UK_ssnydhfw99utr3lo6t06m13kh` (`numero`)
```

```
) ENGINE=InnoDB AUTO_INCREMENT=110 DEFAULT
CHARSET=utf8;
```

```

INSERT INTO `apartamento` VALUES
(82,'Casal','50','Duplo',30),(95,'Solteiro','03','Duplo',30),(96,'Solteiro','04','Duplo',40),(9
7,'Casal','02','Duplo',10),(100,'Casal','13','Duplo',30),(101,'Solteiro','01','Duplo',160),(1
02,'Casal','10','Duplo',100),(103,'Casal','05','Suíte',780),(104,'Casal','06','Duplo',230),(
105,'Solteiro','07','Suíte',230),(106,'Casal','08','Suíte',320),(107,'Casal','98','Duplo',10.
9),(108,'Casal','09','Duplo',100),(109,'Casal','33','Suíte',100);

```

```

DROP TABLE IF EXISTS `hospede`;

```

```

CREATE TABLE `hospede` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `ativo` tinyint(1) NOT NULL,
  `bairro` varchar(255) DEFAULT NULL,
  `cep` varchar(255) DEFAULT NULL,
  `cidade` varchar(255) DEFAULT NULL,
  `cpf` varchar(255) NOT NULL,
  `dataNascimento` date DEFAULT NULL,
  `email` varchar(255) NOT NULL,
  `endereco` varchar(255) DEFAULT NULL,
  `estado` varchar(255) DEFAULT NULL,
  `nome` varchar(60) NOT NULL,
  `observacao` varchar(255) DEFAULT NULL,
  `rg` varchar(255) DEFAULT NULL,
  `senha` varchar(255) NOT NULL,
  `sexo` char(1) NOT NULL,
  `telefone` varchar(255) DEFAULT NULL,
  `permissao` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `UK_pf0vnd57p2ejcof5fotdy9phd` (`cpf`),
  UNIQUE KEY `UK_ai798jdhf53boitstv2c7puct` (`email`)
) ENGINE=InnoDB AUTO_INCREMENT=22 DEFAULT

```

CHARSET=utf8;

```

INSERT INTO `hospede` VALUES (13,1,'Morada','37.266-
626','Itamonte','068.878.236-16','1983-02-13','fabio@bol.com.br','Rua
Nego','MG','Fábio Brito Pinto','MG 11.828.828','fabio423','M','(35)9119-
4611','ROLE_ADMIN'),(14,1,'Centro','83.778-377','Virginia','287.470.647-73','1990-
10-19','carlos.meneses@gmail.com','Rua 7 de Setembro','MG','Antônio Carlos
Meneses Junior','Sou fumante','MG 23.989.229','carlos123','M','(35)9999-
9999','ROLE_USER'),(15,0,'Novo Horizonte','83.883-883','Itamonte','034.832.775-
79','1982-09-19','mariaceci@gmail.com','Rua das Flores','MG','Maria Cecilia','MG
393.393.222','maria','F','(35)9919-1191','ROLE_USER'),(16,0,'Centro','83.883-
883','São Paulo','372.225.436-19','1990-01-01','jairo@hotmail.com','Rua
XV','SP','Jairo Bernardes','SP 12.198.121','jairo123','M','(12)1212-
1212','ROLE_USER'),(17,1,'Vila Perrone','77.377-377','Carmo de
minas','652.549.153-31','1982-01-13','thais@gmail.com','Rua das Garças','MG','Thais
Junqueira','MG 12.983.764','thais123','F','(35)9919-
9191','ROLE_USER'),(18,0,'Centro','34.242-342','Jacarei','513.365.635-16','1988-09-
30','claudio@gmail.com','Rua Ministro Godoi','SP','Claudio Lopes','SP
633.983.229','claudio','M','(34)3434-3434','ROLE_USER'),(19,0,'Centro','83.883-
883','Passa Quatro','245.274.026-84','1983-05-06','erick@gmail.com','Av. Jair
Rodrigues','MG','Erick Junior','SSP 837.234.212','erick123','M','(32)4234-
2342','ROLE_USER'),(20,1,'Centro','Itamonte','050.203.496-31','1980-07-
12','cristiane_ita100@hotmail.com','rua 11 de
setembro','MS','Cristiane','1123654','111111','F','(32)9108-
6789','ROLE_USER'),(21,1,'moradas do bosque','86.856-
453','Varginia','803.442.792-60','1995-04-14','centro_anaclara@gmail.com','rua 11 de
setembro','MG','Ana Clara','676890598099','123456789','F','(89)4757-
4646','ROLE_USER');

/*!40000 ALTER TABLE `hospede` ENABLE KEYS */;

```

```
DROP TABLE IF EXISTS `reserva`;
```

```
CREATE TABLE `reserva` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `dataEntrada` datetime DEFAULT NULL,
  `dataSaida` datetime DEFAULT NULL,
  `hospede` tinyblob,
  `observacao` varchar(255) DEFAULT NULL,
  `qtdAdultos` int(11) NOT NULL,
  `qtdCrianças` int(11) NOT NULL,
  `status` varchar(255) DEFAULT NULL,
  `valor` double NOT NULL,
  `codigo_apartamento` bigint(20) DEFAULT NULL,
  `quantidadeAdultos` int(11) NOT NULL,
  `quantidadeCrianças` int(11) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `FK_gc8qpb77hd7gpf33y657o9pmc` (`codigo_apartamento`),
  CONSTRAINT `FK_gc8qpb77hd7gpf33y657o9pmc` FOREIGN KEY
(`codigo_apartamento`) REFERENCES `apartamento` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

10.2 CÓDIGO FONTE WEB.XML

Código fonte do arquivo web.xml, esse arquivo contém informações de como a aplicação deve se comportar referente a utilização do JSF e sobre a utilização do Primefaces com o tema do Bootstrap.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
  version="3.0">

  <display-name>Unopar</display-name>
  <welcome-file-list>
```

```

        <welcome-file>index.html</welcome-file>
    </welcome-file-list>

<!-- Para funcionamento do Spring Security -->
    <listener>
        <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-class>
        </listener>
    <listener>
        <listener-
class>org.springframework.security.web.session.HttpSessionEventPublisher</listener-
class>
        </listener>

    <listener>
        <listener-
class>org.jboss.weld.environment.servlet.Listener</listener-class>
    </listener>
    <servlet>
        <servlet-name>Faces Servlet</servlet-name>
        <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>Faces Servlet</servlet-name>
        <url-pattern>*.xhtml</url-pattern>
    </servlet-mapping>
    <context-param>
        <param-name>javax.faces.PROJECT_STAGE</param-name>
        <param-value>Production</param-value>
    </context-param>

<!-- Para colocar o Captcha na tela de cadastro do hospede -->
    <context-param>
        <param-name>primefaces.PRIVATE_CAPTCHA_KEY</param-name>
        <param-value>6Le9-PkSAAAAAMgXzyoFi6f3gcs9-E9N1Y0_fj1N</param-value>
    </context-param>
    <context-param>
        <param-name>primefaces.PUBLIC_CAPTCHA_KEY</param-name>
        <param-value>6Le9-PkSAAAAABFNrpvA_6EqdaQh53bSYHdGEQ97</param-value>
    </context-param>

    <filter>
        <filter-name>springSecurityFilterChain</filter-name>
        <filter-
class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
    </filter>
    <filter-mapping>
        <filter-name>springSecurityFilterChain</filter-name>
        <url-pattern>/*</url-pattern>
        <dispatcher>FORWARD</dispatcher>
        <dispatcher>REQUEST</dispatcher>
    </filter-mapping>

</web-app>

```

10.3 CÓDIGO FONTE POM.XML

Código fonte do arquivo pom.xml, esse arquivo faz a configuração das dependências dos arquivos externos que serão utilizados na aplicação, sua utilização é trivial, basta declarar quais arquivos jars necessários que o mesmo realiza o download dele e de suas dependências. Ele é utilizado pelo Maven para realizar os downloads dessas dependências.

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>br.com.unopar</groupId>
  <artifactId>Unopar</artifactId>
  <version>1.0.0-SNAPSHOT</version>

  <packaging>war</packaging>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

  <build>
    <finalName>Unopar</finalName>
    <plugins>
      <plugin>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.1</version>
        <configuration>
          <source>1.7</source>
          <target>1.7</target>
        </configuration>
      </plugin>
    </plugins>
  </build>

  <dependencies>
    <!-- Mojarra (Implementação JSF 2) -->
    <dependency>
      <groupId>org.glassfish</groupId>
      <artifactId>javax.faces</artifactId>
      <version>2.1.21</version>
      <scope>compile</scope>
    </dependency>

    <!-- Primefaces 5 -->
    <dependency>
      <groupId>org.primefaces</groupId>
      <artifactId>primefaces</artifactId>
      <version>5.0</version>
    </dependency>

    <!-- JasperReports -->
```

```

<dependency>
  <groupId>net.sf.jasperreports</groupId>
  <artifactId>jasperreports</artifactId>
  <version>5.6.0</version>
</dependency>

<!-- Apache poi - Excel -->
<dependency>
  <groupId>org.apache.poi</groupId>
  <artifactId>poi</artifactId>
  <version>3.10-FINAL</version>
</dependency>

<!-- Jar do Theme do Bootstrap -->
<dependency>
  <groupId>org.primefaces.themes</groupId>
  <artifactId>bootstrap</artifactId>
  <version>1.0.10</version>
</dependency>

<!-- Weld (implementação do CDI) -->
<dependency>
  <groupId>org.jboss.weld.servlet</groupId>
  <artifactId>weld-servlet</artifactId>
  <version>1.1.10.Final</version>
  <scope>compile</scope>
</dependency>

<!-- OmniFaces (utilitários para JSF) -->
<dependency>
  <groupId>org.omnifaces</groupId>
  <artifactId>omnifaces</artifactId>
  <version>1.4.1</version>
  <scope>compile</scope>
</dependency>

<!-- Núcleo do Hibernate -->
<dependency>
  <groupId>org.hibernate</groupId>
  <artifactId>hibernate-core</artifactId>
  <version>4.2.6.Final</version>
  <scope>compile</scope>
</dependency>

<!-- Hibernate Validator -->
<dependency>
  <groupId>org.hibernate</groupId>
  <artifactId>hibernate-validator</artifactId>
  <version>5.1.1.Final</version>
</dependency>

<!-- Implementação de EntityManager da JPA -->
<dependency>
  <groupId>org.hibernate</groupId>
  <artifactId>hibernate-entitymanager</artifactId>
  <version>4.2.2.Final</version>
  <scope>compile</scope>

```

```

</dependency>

<!-- Driver JDBC do MySQL -->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>5.1.25</version>
    <scope>compile</scope>
</dependency>

<!-- Spring Security (autenticação e autorização) -->
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-core</artifactId>
    <version>3.2.5.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-web</artifactId>
    <version>3.2.5.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-config</artifactId>
    <version>3.2.5.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-jdbc</artifactId>
    <version>3.2.5.RELEASE</version>
</dependency>

<!-- Teste -->
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-taglibs</artifactId>
    <version>3.2.5.RELEASE</version>
</dependency>
<dependency>
<groupId>org.springframework.security</groupId>
<artifactId>facelets-taglib-jsf20-spring-3</artifactId>
<version>0.5</version>
</dependency>

</dependencies>

<!-- Repositorio PrimeFaces -->
<repositories>
    <repository>
        <id>prime-repo</id>
        <name>PrimeFaces Maven Repository</name>
        <url>http://repository.primefaces.org</url>
        <layout>default</layout>
    </repository>

    <repository>
        <id>public-jboss</id>

```



```

        <name>public-jboss</name>
        <url>http://repository.jboss.org/nexus/content/groups/public-jboss</url>
    </repository>
</repositories>

</project>

```

10.4 CÓDIGO FONTE VIEW - CADASTROS

Código fonte das telas de cadastro da aplicação, essas telas possuem referências a alguns contextos no seu cabeçalho.

Abaixo é apresentado o código fonte da página de cadastro de hospede cadastroHospede.xhtml em que o próprio hospede realiza seu cadastro no site, são realizadas validações se os campos requeridos foram preenchidos e se possuem dados válidos, todos os controles de dados preenchidos são validados pelos controllers no momento em que o hospede tenta realizar o seu cadastro no site:

```

<ui:composition template="/template/layoutExterno.xhtml"
    xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:ui="http://java.sun.com/jsf/facelets"
    xmlns:p="http://primefaces.org/ui" xmlns:o="http://omnifaces.org/ui">

    <ui:define name="titulo">Hotel</ui:define>

    <ui:define name="menu_interno">
        <ui:include src="/template/menuPublico.xhtml" />
    </ui:define>

    <ui:define name="corpo">
        <f:view contentType="text/html">
            <f:metadata>
                <o:viewParam name="hospede"
                    value="#{cadastroHospedeBean.hospede.id}" />
            </f:metadata>
            <h:form role="form" id="form">
                <p:panel header="Cadastro de Hospede" id="panel">
                    <p:focus context="panel" />
                    <p:growl id="messages" showDetail="false"
                        autoUpdate="true" closable="true" />

                    <p:panelGrid columns="4" id="grid"
                        styleClass="table-responsive">
                        <p:outputLabel value="Nome:" for="nome" />
                        <p:inputText
                            value="#{cadastroHospedeBean.hospede.nome}" required="true" id="nome"
                            onclick="this.value = ' '; " requiredMessage="Precisa inserir o seu nome" />

```

```

        <p:outputLabel value="Data de nascimento"
for="dataNascimento" />
        <p:calendar id="dataNascimento"
value="#{cadastroHospedeBean.hospede.dataNascimento}" pattern="dd/MM/yyyy"
navigator="true" showOn="button" mode="popup" popupIconOnly="true" locale="pt_BR"
yearRange="-111:+0" required="true" mask="99/99/9999"
maxdate="#{dateBean.maxdate}" requiredMessage="É necessário informar a Data de
Nascimento" />

        <p:outputLabel value="Sexo:" for="sexo" />
        <p:selectOneButton id="sexo"

value="#{cadastroHospedeBean.hospede.sexo}">
        <f:selectItem itemValue="M" itemLabel="Masculino"
/>
        <f:selectItem itemValue="F" itemLabel="Feminino"
/>
        </p:selectOneButton>

        <p:outputLabel value="CPF:" for="cpf" />
        <p:inputMask id="cpf"
value="#{cadastroHospedeBean.hospede.cpf}" mask="999.999.999-99" required="true"
requiredMessage="É obrigatorio a digitação do CPF">
        <f:validateLength maximum="14" for="cpf" />
        </p:inputMask>

        <p:outputLabel value="RG:" for="rg" />
        <p:inputMask id="rg"
value="#{cadastroHospedeBean.hospede.rg}" required="true" requiredMessage="É
obrigatorio a digitação do RG">
        </p:inputMask>
        <p:outputLabel value="Observação:"
for="observacao" />
        <p:inputTextarea id="observacao"
value="#{cadastroHospedeBean.hospede.observacao}" cols="25"
rows="5" autoResize="false" />
        <p:outputLabel value="Tel:" for="tel" />
        <p:inputMask id="tel"
value="#{cadastroHospedeBean.hospede.telefone}"
required="true" requiredMessage="É obrigatorio a digitação do Telefone"
mask="(99)9999-9999">
        </p:inputMask>
        <p:outputLabel value="E-mail" for="email" />
        <p:inputText id="email" required="true"
value="#{cadastroHospedeBean.hospede.email}" requiredMessage="É necessário
informar o seu E-mail" validatorMessage="O seu e-mail não é válido">
        <f:validateRegex
pattern="^[_A-Za-z0-9-\\+](\\.[_A-Za-z0-9-\\+])*@[A-Za-z0-9-]+(\\.[A-
Za-z]{2,})$" />
        </p:inputText>
        <p:outputLabel value="Senha:" for="senha" />
        <p:password id="senha"
value="#{cadastroHospedeBean.hospede.senha}" feedback="false"
match="confirmaSenha" required="true" requiredMessage="Insira sua senha" size="8"
maxlength="20" validatorMessage="O tamanho da sua senha precisar ser entre 5 e 20
caracteres ou não coincidem">
        <f:validateLength maximum="20" minimum="5" />
        </p:password>

```

```

        <p:outputLabel value="Confirmar a Senha:"
for="confirmaSenha" />
        <p:password id="confirmaSenha"
value="#{cadastroHospedeBean.hospede.senha}" required="true"
requiredMessage="Confirme sua senha" size="8" maxLength="20" />
        <p:outputLabel value="Estado:" for="estado" />

<h:selectOneMenu value="#{cadastroHospedeBean.hospede.estado}"
class="form-control" id="estado">
    <f:selectItem itemValue="" itemLabel="Selecione" />
    <f:selectItem itemValue="AC" itemLabel="Acre" />
    <f:selectItem itemValue="AL" itemLabel="Alagoas" />
    <f:selectItem itemValue="AM" itemLabel="Amazonas" />
    <f:selectItem itemValue="AP" itemLabel="Amapá" />
    <f:selectItem itemValue="BA" itemLabel="Bahia" />
    <f:selectItem itemValue="CE" itemLabel="Ceará" />
    <f:selectItem itemValue="DF" itemLabel="Distrito Federal" />
    <f:selectItem itemValue="ES" itemLabel="Espírito Santo" />
    <f:selectItem itemValue="GO" itemLabel="Goiás" />
    <f:selectItem itemValue="MA" itemLabel="Maranhão" />
    <f:selectItem itemValue="MG" itemLabel="Minas Gerais" />
    <f:selectItem itemValue="MS" itemLabel="Mato Grosso do Sul" />
    <f:selectItem itemValue="MT" itemLabel="Mato Grosso" />
    <f:selectItem itemValue="PA" itemLabel="Pará" />
    <f:selectItem itemValue="PB" itemLabel="Paraíba" />
    <f:selectItem itemValue="PE" itemLabel="Pernambuco" />
    <f:selectItem itemValue="PI" itemLabel="Piauí" />
    <f:selectItem itemValue="PR" itemLabel="Paraná" />
    <f:selectItem itemValue="RJ" itemLabel="Rio de Janeiro" />
    <f:selectItem itemValue="RN" itemLabel="Rio Grande do Norte" />
    <f:selectItem itemValue="RO" itemLabel="Rondônia" />
    <f:selectItem itemValue="RR" itemLabel="Roraima" />
    <f:selectItem itemValue="RS" itemLabel="Rio Grande do Sul" />
    <f:selectItem itemValue="SC" itemLabel="Santa Catarina" />
    <f:selectItem itemValue="SE" itemLabel="Sergipe" />
    <f:selectItem itemValue="SP" itemLabel="São Paulo" />
    <f:selectItem itemValue="TO" itemLabel="Tocantins" />
</h:selectOneMenu>

    <p:outputLabel value="Cidade:" for="cidade" />
    <p:inputText id="cidade" value="#{cadastroHospedeBean.hospede.cidade}" />
    <p:outputLabel value="Endereço:" for="endereco" />
    <p:inputText id="endereco"
value="#{cadastroHospedeBean.hospede.endereco}" />
    <p:outputLabel value="Bairro:" for="bairro" />
    <p:inputText id="bairro" value="#{cadastroHospedeBean.hospede.bairro}" />
    <p:outputLabel value="CEP:" for="cep" />
    <p:inputMask id="cep" mask="99.999-999"
value="#{cadastroHospedeBean.hospede.cep}" />
    <p:outputLabel value="Código de Segurança:" />
    <p:captcha label="Captcha" theme="blackglass"/>
    <p:outputLabel/>
    <p:outputLabel/>
    <p:outputLabel/>
    <p:commandButton value="Salvar" update=":form" ajax="false"

```

```

validateClient="true" icon="ui-icon-disk" action="#{cadastroHospedeBean.salvar()}"
/>
    </p:panelGrid>
    </p:panel>
    </h:form>
</f:view>
</ui:define>
</ui:composition>

```

Abaixo é apresentado o código fonte da página de cadastro de apartamentos cadastroApartamento.xhtml, essa página é de acesso somente do administrador, são realizadas na página validações se os campos requeridos foram preenchidos e se possuem valores válidos, todos os controles de dados preenchidos são validados pelos controllers no momento em que o administrador tenta persistir os dados no banco de dados:

```

<ui:composition template="/template/layoutExterno.xhtml"
    xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:ui="http://java.sun.com/jsf/facelets"
    xmlns:p="http://primefaces.org/ui" xmlns:o="http://omnifaces.org/ui">

    <ui:define name="titulo">Hotel</ui:define>

    <ui:define name="menu_interno">
        <ui:include src="/template/menuAdmin.xhtml" />
    </ui:define>

    <ui:define name="corpo">
        <f:view contentType="text/html">
            <f:metadata>
                <o:viewParam name="apartamento"
                    value="#{cadastroApartamentoBean.apartamento}" />
            </f:metadata>
            <h:form id="form">
                <p:panel header="Cadastro de Apartamento" id="panel">
                    <p:focus context="panel" />
                    <p:growl id="messages" showDetail="false" autoUpdate="true" closable="true"
/>
                    <p:toolbar style="margin-top: 20px">
                        <p:toolbarGroup align="right">
                            <p:button value="Pesquisa" outcome="ListagemApartamento.xhtml" />
                        </p:toolbarGroup>
                    </p:toolbar>
                    <p:panelGrid columns="2" id="painel" styleClass="table-responsive"
columnClasses="rotulo, campo">
                        <p:outputLabel value="Id.:" />
                        <p:inputText value="#{cadastroApartamentoBean.apartamento.id}"
readonly="true" disabled="true" id="id" />
                        <p:outputLabel value="Numero:" for="numero" />
                        <p:inputText value="#{cadastroApartamentoBean.apartamento.numero}"

```

```

required="true" id="numero" requiredMessage="Precisa inserir o numero do
Apartamento">
    </p:inputText>
    <p:outputLabel value="Tipo" for="tipo" />
    <h:selectOneMenu value="#{cadastroApartamentoBean.apartamento.tipo}"
class="form-control" id="tipo">
        <f:selectItem itemValue="" itemLabel="Selecione" />
        <f:selectItem itemValue="Suíte" itemLabel="Suíte" />
        <f:selectItem itemValue="Duplo" itemLabel="Duplo" />
        <f:selectItem itemValue="Luxo" itemLabel="Luxo" />
        <f:selectItem itemValue="Simples" itemLabel="Simples" />
        <f:selectItem itemValue="Premium" itemLabel="Premium" />
    </h:selectOneMenu>
    <p:outputLabel value="Cama" for="cama" />
    <h:selectOneMenu value="#{cadastroApartamentoBean.apartamento.cama}"
class="form-control" id="cama">
        <f:selectItem itemValue="" itemLabel="Selecione" />
        <f:selectItem itemValue="Solteiro" itemLabel="Solteiro" />
        <f:selectItem itemValue="Casal" itemLabel="Casal" />
    </h:selectOneMenu>
    <p:outputLabel value="Valor:" for="valor" />
    <p:inputText value="#{cadastroApartamentoBean.apartamento.valor}"
id="valor" required="true" requiredMessage="É obrigatorio a digitação
do Valor">
        <f:convertNumber type="currency" currencySymbol="R$" />
        <f:validateDoubleRange minimum="0.01" />
    </p:inputText>
    <p:outputLabel value="" />
    <p:commandButton value="Salvar" update=":form" validateClient="true"
icon="ui-icon-disk" action="#{cadastroApartamentoBean.salvar()}" />
</p:panelGrid>
<p:panelGrid columns="1" styleClass="table-responsive">
</p:panelGrid>
</p:panel>
</h:form>
</f:view>
</ui:define>
</ui:composition>

```

10.5 CÓDIGO FONTE DA PÁGINA COM DADOS SALVOS NO BANCO DE DADOS COM POSSIBILIDADE DE EXCLUSÃO E ALTERAÇÃO

Abaixo é apresentado o código da página de listagem de Hospedes que é disponível somente aos usuários com permissão de administrador.

Código da página listagemHospedes.xhtml:

```

<ui:composition template="/template/LayoutExterno.xhtml"
xmlns="http://www.w3.org/1999/xhtml"
xmlns:h="http://java.sun.com/jsf/html"
xmlns:f="http://java.sun.com/jsf/core"
xmlns:ui="http://java.sun.com/jsf/facelets"
xmlns:p="http://primefaces.org/ui">

```

```

<ui:define name="titulo">Hotel</ui:define>

<ui:define name="menu_interno">
    <ui:include src="/template/menuAdmin.xhtml" />
</ui:define>

<ui:define name="corpo">
    <p:panel header="Listagem de Hospedes" id="panel">
        <p:toolbar style="margin-top: 20px">
            <p:toolbarGroup>
                <p:commandButton value="Novo" id="botaoNovo"
                    action="cadastroHospede.xhtml" />
            </p:toolbarGroup>
        </p:toolbar>
        <h:form id="form">
            <p:growl id="msgs" showDetail="true" />
            <p:dataTable id="tabelaHospede" var="hospede"
                sortBy="#{hospede.nome}" widgetVar="tabelaHospede"
                value="#{pesquisaHospedeBean.hospedes}" paginator="true" rows="5"
                rowKey="#{hospede.id}" emptyMessage="Nenhum Hospede encontrado!"
                paginatorTemplate="{CurrentPageReport} {FirstPageLink}
{PreviousPageLink} {PageLinks} {NextPageLink} {LastPageLink}
{RowsPerPageDropDown}"
                rowsPerPageTemplate="10,20,30,40,50,100">
                <f:facet name="header">Lista de Hospedes</f:facet>
                <p:column headerText="Id" sortBy="#{hospede.id}">
                    <h:outputText value="#{hospede.id}" />
                </p:column>
                <p:column headerText="Nome" sortBy="#{hospede.nome}"
                    filterBy="#{hospede.nome}" filterMatchMode="contains">
                    <h:outputText value="#{hospede.nome}" />
                </p:column>
                <p:column headerText="E-mail" sortBy="#{hospede.email}">
                    <h:outputText value="#{hospede.email}" />
                </p:column>
                <p:column headerText="CPF" filterBy="#{hospede.cpf}"
                    filterMatchMode="contains">
                    <h:outputText value="#{hospede.cpf}" />
                </p:column>
                <p:column style="width:145px;text-align: center" headerText="Ação"
                    exportable="false">
                    <p:commandButton update=":form:detalheHospede"
                        oncomplete="PF('hospedeDialog').show()" icon="ui-icon-search"
                        title="Visualizar">
                        <f:setPropertyActionListener value="#{hospede}"
                            target="#{pesquisaHospedeBean.hospedeSelecionado}" />
                    </p:commandButton>
                    <p:button outcome="cadastroHospede.xhtml" icon="ui-icon-pencil"
                        title="Editar">
                        <f:param name="hospede" value="#{hospede.id}" />
                    </p:button>
                    <p:commandButton icon="ui-icon-trash" title="Excluir"
                        oncomplete="confirmacaoExclusao.show()" process="@this"
                        update=":form:tabelaHospede"
                        action="#{pesquisaHospedeBean.excluir()}">
                        <p:confirm header="Exclusão de Hospede"
                            message="Deseja excluir o Hospede ?" icon="ui-icon-alert" />
                        <f:setPropertyActionListener
                            target="#{pesquisaHospedeBean.hospedeSelecionado}"

```

```

        value="#{hospede}" />
    </p:commandButton>
</p:column>
<f:facet name="footer">
Existem #{pesquisaHospedeBean.hospedes.size()} hospedes cadastrados.
<p:separator />
    <p:commandLink ajax="false">
    <p:graphicImage library="images" name="pdf.png"
    title="Gerar em PDF" />
    <p:dataExporter type="pdf" target="tabelaHospede"
    encoding="UTF-8" fileName="hospedes" />
    </p:commandLink>
    <p:commandLink ajax="false">
    <p:graphicImage library="images" name="csv.png"
    title="Gerar em CSV" />
    <p:dataExporter type="csv" target="tabelaHospede"
    encoding="UTF-8" fileName="hospedes" />
    </p:commandLink>
    <p:commandLink ajax="false">
    <p:graphicImage library="images" name="excel.png"
    title="Gerar em Excel" />
    <p:dataExporter type="xls" target="tabelaHospede"
    encoding="UTF-8" fileName="hospedes" />
    </p:commandLink>
</f:facet>
</p:dataTable>
<p:confirmDialog global="true" showEffect="fade"
widgetVar="confirmacaoExclusao">
    <p:commandButton value="Sim" type="button"
onclick="confirmacaoExclusao.hide()"
styleClass="ui-confirmdialog-yes" icon="ui-icon-check"
update=":form:tabelaHospede" />
    <p:commandButton value="Não" type="button"
styleClass="ui-confirmdialog-no" icon="ui-icon-close" />
</p:confirmDialog>
<p:dialog header="Informação do Hospede" widgetVar="hospedeDialog"
modal="true" showEffect="fade" hideEffect="fade" resizable="false">
    <p:outputPanel id="detalheHospede" style="text-align:center;">
    <p:panelGrid columns="4"
rendered="#{not empty pesquisaHospedeBean.hospedeSelecionado}">
        <f:facet name="header">
        <p:graphicImage library="images" name="Logo.png"
        styleClass="content" />
        </f:facet>
        <h:outputText value="Id:" />
        <h:outputText
        value="#{pesquisaHospedeBean.hospedeSelecionado.id}" />
        <h:outputText value="Nome:" />
        <h:outputText
        value="#{pesquisaHospedeBean.hospedeSelecionado.nome}" />
        <h:outputText value="Data Nasc."
        sortBy="#{pesquisaHospedeBean.hospedeSelecionado.dataNascimento}" />
        <h:outputText
        value="#{pesquisaHospedeBean.hospedeSelecionado.dataNascimento}">
        <f:convertDateTime pattern="dd/MM/YYYY" />
        </h:outputText>
        <h:outputText value="Sexo" />
        <h:outputText
        value="#{pesquisaHospedeBean.hospedeSelecionado.sexo}" />

```



```

        <h:outputText value="CPF" />
        <h:outputText
value="#{pesquisaHospedeBean.hospedeSelecionado.cpf}" />
        <h:outputText value="RG" />
        <h:outputText
value="#{pesquisaHospedeBean.hospedeSelecionado.rg}" />
        <h:outputText value="Observação" />
        <h:outputText
value="#{pesquisaHospedeBean.hospedeSelecionado.observacao}" />
        <h:outputText value="Telefone" />
        <h:outputText
value="#{pesquisaHospedeBean.hospedeSelecionado.telefone}" />
        <h:outputText value="e-mail" />
        <h:outputText
value="#{pesquisaHospedeBean.hospedeSelecionado.email}" />
        <h:outputText value="Endereço" />
        <h:outputText
value="#{pesquisaHospedeBean.hospedeSelecionado.endereco}" />
        <h:outputText value="Bairro" />
        <h:outputText
value="#{pesquisaHospedeBean.hospedeSelecionado.bairro}" />
        <h:outputText value="Cidade" />
        <h:outputText
value="#{pesquisaHospedeBean.hospedeSelecionado.cidade}" />
        <h:outputText value="CEP" />
        <h:outputText
value="#{pesquisaHospedeBean.hospedeSelecionado.cep}" />
        <h:outputText value="Estado" />
        <h:outputText
value="#{pesquisaHospedeBean.hospedeSelecionado.estado}" />
        <h:outputText value="Ativo" />
        <p:graphicImage library="images"
name="usuario_ativo_#{pesquisaHospedeBean.hospedeSelecionado.ativo}.png" />
        <p:commandButton type="button" title="Imprimir"
icon="ui-icon-print">
        <p:printer target="detalheHospede" />
        </p:commandButton>
    </p:panelGrid>
</p:outputPanel>
</p:dialog>
</h:form>
</p:panel>
</ui:define>
</ui:composition>

```

10.6 CÓDIGO FONTE DA PÁGINA COM DADOS SALVOS NO BANCO DE DADOS

Abaixo é apresentado o código da página pública que apresenta aos visitantes os apartamentos cadastrados e seus respectivos valores, esses dados são buscados da base de dados cadastrada pelo administrador.

Código da página quartos.xhtml:

```
<ui:composition template="/template/layoutExterno.xhtml">
```



```

xmlns="http://www.w3.org/1999/xhtml"
xmlns:h="http://java.sun.com/jsf/html"
xmlns:f="http://java.sun.com/jsf/core"
xmlns:ui="http://java.sun.com/jsf/facelets"
xmlns:p="http://primefaces.org/ui">

<ui:define name="titulo">Hotel</ui:define>

<ui:define name="menu_interno">
    <ui:include src="/template/menuPublico.xhtml" />
</ui:define>

<ui:define name="corpo">
    <div class="container">
        <h1>Quartos</h1>
        <p:dataTable widgetVar="dataTableWidgetVar" id="tabelaApart"
var="apart" value="#{pesquisaApartamentoBean.apartamentos}"
editable="true" rowKey="#{apart.id}" cellspacing="0" cellpadding="2"
emptyMessage="Nenhum Apartamento encontrado!" sortBy="#{apart.numero}">
            <p:column headerText="Número">
                <h:outputText value="#{apart.numero}" />
            </p:column>
            <p:column headerText="Tipo">
                <h:outputText value="#{apart.tipo}" />
            </p:column>
            <p:column headerText="Cama">
                <h:outputText value="#{apart.cama}" />
            </p:column>
            <p:column headerText="Valor">
                <h:outputText value="#{apart.valor}" >
                <f:convertNumber pattern="R #,##0.00"/>
            </h:outputText>
            </p:column>
        </p:dataTable>
    </div>
</ui:define>
</ui:composition>

```

10.7 CÓDIGO FONTE TEMPLATE

Códigos fontes do template utilizado, menu do sistema e do rodapé. Foram criados três menus um para acesso administrativo, um para acesso público e outro para acesso de hospede logado, o que os diferencia são os links que mandam para as páginas.

Abaixo é apresentado o código fonte do menu template layoutExterno.xhtml, que utiliza Facelets:

```

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
xmlns:h="http://java.sun.com/jsf/html"
xmlns:p="http://primefaces.org/ui"
xmlns:f="http://java.sun.com/jsf/core"
xmlns:ui="http://java.sun.com/jsf/facelets">

```

```

<h:head>
    <title><ui:insert name="titulo">Hotel</ui:insert>
    </title>
    <h:outputStylesheet library="css" name="estilo.css" />
    <h:outputStylesheet library="css" name="bootstrap.css" />
    <h:outputScript library="js" name="bootstrap.js" />
    <h:outputScript library="js" name="bootstrap.min.js" />
</h:head>
<h:body>

    <div id="menu">
        <ui:insert name="menu_interno">
            <ui:include src="/template/menuExterno.xhtml" />
        </ui:insert>
    </div>
    <div id="conteudo">
        <ui:insert name="corpo" />
    </div>
    <div id="rodape">
        <ui:include src="/template/rodape.xhtml" />
    </div>
</h:body>
</html>

```

Abaixo é apresentado o código fonte do menu público:

```

<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:p="http://primefaces.org/ui"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:ui="http://java.sun.com/jsf/facelets">

    <ui:composition>
        <h:head>
            <title>Hotel</title>
            <h:outputStylesheet library="css" name="bootstrap.css" />
            <h:outputScript library="js" name="bootstrap.js" />
            <h:outputScript library="js" name="bootstrap.min.js" />
        </h:head>
        <div>
            <nav class="navbar navbar-inverse" role="navigation">
                <div class="navbar-header">
                    <button type="button" class="navbar-toggle collapsed"
data-toggle="collapse" data-target="#bs-example-navbar-collapse-9">
<span class="sr-only">Toggle navigation</span> <span class="icon-bar"></span>
<span class="icon-bar"></span> <span class="icon-bar"></span>
</button>
<a class="navbar-brand" href="index.xhtml">Hotel</a>
                </div>
                <div class="navbar-collapse collapse" id="bs-example-navbar-collapse-9"
style="height: 1px;">
                    <ul class="nav navbar-nav">
                        <li><a href="index.xhtml">Início</a></li>
                        <li><a href="quartos.xhtml">Quartos</a></li>
                        <li><a href="contato.xhtml">Contato</a></li>

```

```

        <li data-toggle="modal" data-target="#myModal">
        <h:link>Login</h:link></li>
    </ul>
</div>
</nav>
</div>

<!-- Modal -->
<div class="modal fade" id="myModal" tabindex="-1" role="dialog"
    aria-labelledby="myModalLabel" aria-hidden="true">
    <div class="modal-dialog">
        <div class="modal-content">
            <div class="modal-header">
                <button type="button" class="close" data-
dismiss="modal" aria-hidden="true"></button>
                <h4 class="modal-title" id="myModalLabel">Login</h4>
            </div>

<form action="#{request.contextPath}/j_spring_security_check"
    method="post" >
        <div class="modal-body">
            <h:outputLabel for="j_username" value="E-mail: * " />
            <h:inputText id="j_username" required="true" />
            <br></br>
            <h:outputLabel for="j_password" value="Senha: * " />
            <h:inputSecret id="j_password" required="true"/>
        </div>
        <div class="modal-footer">
            <button type="button" class="btn btn-default" data-
dismiss="modal">Fechar</button>
            <h:commandButton class="btn btn-primary" value="Login" />
            <p:separator />
            <h:link>
            <a href="cadastroHospede.xhtml" target="_self">Cadastrar-se </a>
            </h:link>
            <br />
            <h:link>Esqueci minha senha</h:link>
        </div>
    </form>
    </div>
    <!-- /.modal-content -->
</div>
<!-- /.modal-dialog -->
</div>
</ui:composition>
</html>

```

Abaixo é apresentado o código fonte do menu restrito, nesse menu é realizado uma consulta se quem fez o acesso é Administrador ou Hospede, e a partir desse ponto é apresentado o menu com as opções:

```

<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:p="http://primefaces.org/ui"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:ui="http://java.sun.com/jsf/facelets"
    xmlns:o="http://omnifaces.org/ui"

```

```

xmlns:sec="http://www.springframework.org/security/facelets/tags">

<ui:composition>
  <h:head>
    <title>Hotel</title>
    <h:outputStylesheet library="css" name="bootstrap.css" />
    <h:outputScript library="js" name="bootstrap.js" />
    <h:outputScript library="js" name="bootstrap.min.js" />
  </h:head>
  <div>
    <nav class="navbar navbar-inverse" role="navigation">
      <div class="navbar-header">
        <button type="button" class="navbar-toggle collapsed"
          data-toggle="collapse" data-target="#bs-example-navbar-collapse-9">
          <span class="sr-only">Toggle navigation</span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </button>
      <a class="navbar-brand" href="">Hotel</a>
    </div>
    <div class="navbar-collapse collapse"
      id="bs-example-navbar-collapse-9" style="height: 1px;">
      <ul class="nav navbar-nav ">
        <li><a href="index.xhtml">Início</a></li>
        <sec:ifAnyGranted roles="ROLE_USER">
          <li><a href="#">Meus Dados</a></li>
          <li><a href="#">Minhas Reservas</a></li>
        </sec:ifAnyGranted>

        <sec:ifAnyGranted roles="ROLE_ADMIN">
          <li><a
href=" ../admin/ListagemApartamento.xhtml">Apartamentos</a></li>
          <li><a href=" ../admin/ListagemHospedes.xhtml">Hospedes</a></li>
          <li><a href=" ../admin/ListagemReservas.xhtml">Reservas</a></li>
          <li><a href=" ../admin/relatorios.xhtml">Relatórios</a></li>
        </sec:ifAnyGranted>
          <li><a
href="#{request.contextPath}/j_spring_security_logout">Sair</a></li>
        </ul>
      </div>
    </nav>
  </div>
</ui:composition>

</html>

```

Abaixo é apresentado o código fonte do rodape.xhtml:

```

<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
  xmlns:ui="http://java.sun.com/jsf/facelets"
  xmlns:p="http://primefaces.org/ui"
  xmlns:h="http://java.sun.com/jsf/html">

  <ui:composition>
    <p:graphicImage library="images" name="Logo.png" />
    <br />
    <p:outputLabel value="Sistema de Reserva de Quartos." />
  </ui:composition>

```

```

        <br />
        <p:outputLabel value="Fábio Brito Pinto - ead01398352." />
    </ui:composition>

</html>

```

10.8 CÓDIGO FONTE MODEL

Como forma de demonstração de código de classes Modelo (POJO) será apresentado a classe Hospede.java, essa classe possui os principais tipos de dados disponíveis em Java, ela também possui Annotations do JPA e Hibernate, essa classe possui implementações de Annotations do Hibernate Validation, que é capaz de realizar validações na camada de apresentação, controle e persistência, possibilitando mais uma forma de validar a integridade dos dados que serão persistidos no banco de dados.

Em classes Modelo se faz necessário a implementação dos métodos hashCode() e equals() que são herdados da classe Object do Java, isso é para garantir que objetos semelhantes não sejam considerados como o mesmo objeto.

Segue o código abaixo:

```

package br.com.unopar.modelo;

import java.io.Serializable;
import java.util.Date;

import javax.persistence.*;

import org.hibernate.validator.constraints.Email;
import org.hibernate.validator.constraints.Length;
import org.hibernate.validator.constraints.br.CPF;

@Entity
@Table(name = "hospede")
public class Hospede implements Serializable {

    private static final long serialVersionUID = -8779725522629768002L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column(name = "nome", nullable = false)
    private String nome;
    @CPF
    @Column(unique = true, nullable = false)

```

```

    private String cpf;
    @Temporal(TemporalType.DATE)
    private Date dataNascimento;
    private String rg;
    private String observacao;
    @Email
    @Column(unique = true, nullable = false)
    private String email;
    @Column(nullable = false)
    @Length(min = 5, max = 20, message = "A senha deve ter entre 5 a 20
caracteres")
    private String senha;
    private String telefone;
    private String endereco;
    private String estado;
    private String bairro;
    private String cidade;
    private String cep;
    private char sexo;
    private boolean ativo;
    private String permissao;
    public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }
    public String getNome() {
        return nome;
    }
    public void setNome(String nome) {
        this.nome = nome;
    }
    public String getCpf() {
        return cpf;
    }
    public void setCpf(String cpf) {
        this.cpf = cpf;
    }
    public Date getDataNascimento() {
        return dataNascimento;
    }
    public void setDataNascimento(Date dataNascimento) {
        this.dataNascimento = dataNascimento;
    }
    public String getRg() {
        return rg;
    }
    public void setRg(String rg) {
        this.rg = rg;
    }
    public String getObservacao() {
        return observacao;
    }
    public void setObservacao(String observacao) {
        this.observacao = observacao;
    }
    public String getEmail() {
        return email;
    }

```

```
}  
public void setEmail(String email) {  
    this.email = email;  
}  
public String getSenha() {  
    return senha;  
}  
public void setSenha(String senha) {  
    this.senha = senha;  
}  
public String getTelefone() {  
    return telefone;  
}  
public void setTelefone(String telefone) {  
    this.telefone = telefone;  
}  
public String getEndereco() {  
    return endereco;  
}  
public void setEndereco(String endereco) {  
    this.endereco = endereco;  
}  
public String getEstado() {  
    return estado;  
}  
public void setEstado(String estado) {  
    this.estado = estado;  
}  
public String getBairro() {  
    return bairro;  
}  
public void setBairro(String bairro) {  
    this.bairro = bairro;  
}  
public String getCidade() {  
    return cidade;  
}  
public void setCidade(String cidade) {  
    this.cidade = cidade;  
}  
public String getCep() {  
    return cep;  
}  
public void setCep(String cep) {  
    this.cep = cep;  
}  
public char getSexo() {  
    return sexo;  
}  
public void setSexo(char sexo) {  
    this.sexo = sexo;  
}  
public boolean isAtivo() {  
    return ativo;  
}  
public void setAtivo(boolean ativo) {  
    this.ativo = ativo;  
}  
public String getPermissao() {
```

```

        return permissao;
    }
    public void setPermissao(String permissao) {
        this.permissao = permissao;
    }
    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + ((cpf == null) ? 0 : cpf.hashCode());
        result = prime * result + ((email == null) ? 0 : email.hashCode());
        result = prime * result + ((id == null) ? 0 : id.hashCode());
        return result;
    }
    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        Hospede other = (Hospede) obj;
        if (cpf == null) {
            if (other.cpf != null)
                return false;
        } else if (!cpf.equals(other.cpf))
            return false;
        if (email == null) {
            if (other.email != null)
                return false;
        } else if (!email.equals(other.email))
            return false;
        if (id == null) {
            if (other.id != null)
                return false;
        } else if (!id.equals(other.id))
            return false;
        return true;
    }
}
}

```

10.9 CÓDIGO FONTE CONTROLLER

O Controller atua como intermediário antes de persistir os dados advindos das telas, abaixo podemos visualizar o Controller do cadastro de Hospedes.

```
package br.com.unopar.controller;
```



```

import java.io.Serializable;

import javax.annotation.PostConstruct;
import javax.faces.bean.ViewScoped;
import javax.inject.Inject;
import javax.inject.Named;

import br.com.unopar.modelo.Hospede;
import br.com.unopar.service.CadastroHospedeService;
import br.com.unopar.util.jsf.FacesUtil;

@Named
@ViewScoped
public class CadastroHospedeBean implements Serializable {

    private static final long serialVersionUID = -4572775863171471066L;

    @Inject
    private CadastroHospedeService cadastroHospede;

    private Hospede hospede;

    public void salvar() {
        try {
            this.cadastroHospede.salvar(hospede);
            FacesUtil.addSuccessMessage("Hospede "
                + hospede.getNome()
                + " cadastrado com Sucesso.");
            this.limpar();
        } catch (Exception e) {
            FacesUtil.addErrorMessage("O hospede "
                + hospede.getNome()
                + " não pode ser cadastrado.");
        }
    }

    @PostConstruct
    public void init() {
        this.limpar();
    }

    private void limpar() {
        hospede = new Hospede();
    }

    public Hospede getHospede() {
        return hospede;
    }

    public void setHospede(Hospede hospede) {
        this.hospede = hospede;
    }

}

```

10.10 CÓDIGO FONTE DAO

São classes responsáveis por fazer acesso no banco de dados, é uma forma de realizar uma divisão lógica do sistema. Abaixo podemos visualizar o código do DAO do Hospede.

```
package br.com.unopar.dao;

import java.io.Serializable;
import java.util.List;

import javax.inject.Inject;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceException;

import br.com.unopar.modelo.Hospede;
import br.com.unopar.service.NegocioException;
import br.com.unopar.util.jpa.Transactional;

public class HospedeDAO implements Serializable {

    private static final long serialVersionUID = 6077319112955438350L;

    @Inject
    private EntityManager em;

    public void salvar(Hospede hospede) {
        if(hospede.getPermissao() == "" || hospede.getPermissao() == null){
            hospede.setPermissao("ROLE_USER");
        }
        em.merge(hospede);
    }

    public List<Hospede> buscarTodos() {
        return em.createQuery("from Hospede", Hospede.class).getResultList();
    }

    @Transactional
    public void excluir(Hospede hospede) throws NegocioException {
        Hospede hospedeTemp = em.find(Hospede.class, hospede.getId());
        try{
            em.remove(hospedeTemp);
            em.flush();
        }catch(PersistenceException e){
```

```

        throw new NegocioException("Este hospede não pode ser
excluído");
    }
}

    public Hospede buscarPeloCodigo(Long codigo){
        return em.find(Hospede.class, codigo);
    }
}

```

10.11 CÓDIGO FONTE CONVERTER

Quando precisamos resgatar um registro no banco para convertê-lo para Objeto ou converter um Objeto para um registro no banco temos que utilizar o Converter para realizar essa conversão, abaixo segue o Converter do Apartamento.

```

package br.com.unopar.converter;

import javax.faces.component.UIComponent;
import javax.faces.context.FacesContext;
import javax.faces.convert.Converter;
import javax.faces.convert.FacesConverter;

import br.com.unopar.dao.ApartamentoDAO;
import br.com.unopar.modelo.Apartamento;
import br.com.unopar.util.cdi.CDIServiceLocator;

@FacesConverter(forClass = Apartamento.class )
public class ApartamentoConverter implements Converter{

    private ApartamentoDAO apartamentoDAO;

    public ApartamentoConverter(){
        this.apartamentoDAO =
CDIServiceLocator.getBean(ApartamentoDAO.class);
    }

    @Override
    public Object getAsObject(FacesContext context, UIComponent component,
String value) {
        Apartamento retorno = null;
        if(value != null){
            retorno = this.apartamentoDAO.buscarPeloCodigo(new
Long(value));
        }
        return retorno;
    }

    @Override
    public String getAsString(FacesContext context, UIComponent component,
Object value) {
        if (value != null) {
            Long codigo = ((Apartamento)value).getId();

```

```

        String retorno = (codigo == null ? null : codigo.toString());
        return retorno;
    }
    return "";
}

}

```

10.12 CÓDIGO FONTE SERVICE

Os services serve como regras de negócio, com ele verificamos se está faltando algo para que o objeto seja persistido no banco.

```

package br.com.unopar.service;

import java.io.Serializable;

import javax.inject.Inject;

import br.com.unopar.dao.ApartamentoDAO;
import br.com.unopar.modelo.Apartamento;
import br.com.unopar.util.jpa.Transactional;

public class CadastroApartamentoService implements Serializable {

    private static final long serialVersionUID = -1968484565736205993L;

    @Inject
    private ApartamentoDAO apartamentoDAO;
    @Transactional
    public void salvar(Apartamento apartamento) throws NegocioException{
        if(apartamento.getNumero() == null ||
        apartamento.getNumero().trim().equals("")){
            throw new NegocioException("O Apartamento deve conter um
            numero e um valor!");
        }
        if(apartamento.getValor() == 0){
            throw new NegocioException("O Apartamento precisa ter um
            valor.");
        }
        apartamentoDAO.salvar(apartamento);
    }
}

```

10.13 CÓDIGO FONTE NEGOCIOEXCEPTION.JAVA

Como forma de personalizar as mensagens de erro no sistema, foi implementada a classe `NegocioException` que estende a classe `Exception`, abaixo é apresentado seu código.

```
package br.com.unopar.service;

public class NegocioException extends Exception {

    private static final long serialVersionUID = -1771807272939597055L;

    public NegocioException(String message) {
        super(message);
    }

}
```

10.14 CÓDIGO FONTE PERSISTENCE.XML

Código fonte do arquivo `Persistence.xml` que faz a configuração do JPA para utilizar o Hibernate na aplicação e os parâmetros para o acesso ao banco de dados da aplicação proposta. Nele é declarado o nome da unidade persistente que recebe o nome `unoparPU` com o tipo de transação `RESOURCE_LOCAL` onde indicamos que nós iremos decidir quando uma transação deve ocorrer e não deixar o JPA decidir isso.

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.0"
    xmlns="http://java.sun.com/xml/ns/persistence"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd">

    <persistence-unit name="unoparPU"
        transaction-type="RESOURCE_LOCAL">

        <class>br.com.unopar.modelo.Apartamento</class>
        <class>br.com.unopar.modelo.Hospede</class>
        <properties>
            <property name="javax.persistence.jdbc.url"
value="jdbc:mysql://localhost:3306/hospedagem" />
            <property name="javax.persistence.jdbc.user" value="root" />
            <property name="javax.persistence.jdbc.password" value="root"
/>

            <property name="javax.persistence.jdbc.driver"
value="com.mysql.jdbc.Driver" />
```

```

        <!-- validate | update | create | create-drop -->
        <property name="hibernate.hbm2ddl.auto" value="update" />
        <property name="hibernate.show_sql" value="true" />
        <property name="hibernate.format_sql" value="true" />
    </properties>
</persistence-unit>

</persistence>

```

10.15 CÓDIGO FONTE ENTITYMANAGERPRODUCER.JAVA

Código fonte do EntityManagerProducer.java, esse arquivo é responsável por criar uma instancia do EntityManagerFactory, essa instancia é responsável gerenciar as entidades de persistência do banco de dados e objetos, ela recebe como parâmetro a unidade que será persistida que foi declarada no arquivo Persistence.xml.

Segue o código do arquivo:

```

package br.com.unopar.util.jpa;

import javax.enterprise.context.ApplicationScoped;
import javax.enterprise.context.RequestScoped;
import javax.enterprise.inject.Disposes;
import javax.enterprise.inject.Produces;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

@ApplicationScoped
public class EntityManagerProducer {

    private EntityManagerFactory factory;

    public EntityManagerProducer() {
        this.factory = Persistence.createEntityManagerFactory("Unopar");
    }

    @Produces
    @RequestScoped
    public EntityManager create() {
        return factory.createEntityManager();
    }

    public void close(@Disposes EntityManager manager) {
        manager.close();
    }

}

```

10.16 CÓDIGO FONTE APPLICATIONCONTEXT.XML

O arquivo applicationContext.xml contém as configurações do Spring Security, ele possui também as configurações de acesso ao banco de dados para que seja verificado o e-mail e senha dos usuários, podemos perceber ainda que nesse arquivo possui as informações de quais grupos de usuários podem acessar certas pastas na estrutura do site, nele também encontrarmos o direcionamento para a tela de login e para tela de erro caso algum usuário digite a senha errada.

```
<beans:beans xmlns="http://www.springframework.org/schema/security"
    xmlns:beans="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
http://www.springframework.org/schema/security
http://www.springframework.org/schema/security/spring-security-3.2.xsd">
    <http auto-config="true" access-denied-page="/publico/negado.xhtml">
        <intercept-url pattern="/admin/**" access="ROLE_ADMIN" />
        <intercept-url pattern="/restrito/**" access="ROLE_USER,ROLE_ADMIN"
    />
        <form-login login-page="/publico/Login.xhtml" default-target-
url="/restrito/index.xhtml"
            authentication-failure-
url="/publico/Login.xhtml?login_error=1" />
    </http>
    <authentication-manager>
        <authentication-provider>
            <jdbc-user-service data-source-ref="dataSource"
users-by-username-query="SELECT email, senha, 1 FROM hospede WHERE email=?"
authorities-by-username-query="SELECT email, permissao FROM hospede WHERE email=?"
        />
    </authentication-provider>
</authentication-manager>

    <beans:bean id="dataSource"
        class="org.springframework.jdbc.datasource.DriverManagerDataSource">
        <beans:property name="url"
            value="jdbc:mysql://localhost:3306/hospedagem" />
        <beans:property name="driverClassName" value="com.mysql.jdbc.Driver"
    />
        <beans:property name="username" value="root" />
        <beans:property name="password" value="root" />
    </beans:bean>

</beans:beans>
```

11 CONCLUSÃO

Conclui que a tecnologia Java é muito extensa e que existe várias opções de frameworks para a mesma solução, nesse trabalho apresentado tive o intuito de mostrar um pequeno pedado desse mundo que é a plataforma Web do Java que é o Java EE.

Percebe-se que a tecnologia Java tem evoluído a cada nova versão e que essas versões passam por uma entidade avaliadora, onde uma nova especificação só é lançada após uma votação dos membros desse conselho, aprender uma tecnologia que segue esses preceitos é de suma importância, pois ela provavelmente se tornará padrão de mercado.

Outro fator percebido para uma aplicação é o layout apresentado nas páginas, deve ser levado em consideração o posicionamento das imagens, a tipografia das letras e a harmonia das cores. Uma aplicação mesmo que funcional se não levar em consideração esses preceitos estará fardada ao fracasso.

Partindo desse pressuposto a aplicação apresentada é interativa com os usuários, retornando informações sobre algo

Os objetivos gerais e específicos foram atendidos e juntamente com eles foi possível aprimorar meus conhecimentos no mundo do desenvolvimento web utilizando os frameworks Java.

Posteriormente caso algum estabelecimento venha a utilizar a solução proposta, ela ganhará agilidade, eficiência e autonomia para modificação nos dados cadastrados via sistema e banco de dados.

REFERÊNCIAS

ANDRADE, Thiago Faria de. **DWJSF – Desenvolvimento Web com JavaServer Faces**. 2. ed. São Paulo: AlgaWorks Softwares, Treinamentos e Serviços LTDA, 2010.

ANDRADE, Thiago Faria de. **Java EE 7 com JSF, PrimeFaces e CDI**. São Paulo: AlgaWorks Softwares, Treinamentos e Serviços LTDA, 2013.

BALSAMIQ. **Balsamiq Mockups For Desktop**. Disponível em: <<http://balsamiq.com/products/mockups/>>. Acesso em: 11 jun. 2014.

BRAINS, Idle. **Servlets : Calling init, service and destroy explicitly**. Disponível em: <<http://idlebrains.org/tutorials/java-tutorials/servlets-init-service-destroy/>>. Acesso em: 24 maio 2014.

BRASIL. **Lei nº 8.069, de 13 de julho de 1990. Estatuto da Criança e do Adolescente. Art. 82**. Brasília, DF, 13 de Julho de 1990. Disponível em <http://www.planalto.gov.br/ccivil_03/leis/L8069.htm>. Acesso em: 11 jul. 2014.

CAELUM. **Desenvolvimento Web com HTML, CSS e JavaScript Curso WD-43**. São Paulo, 2014.

CAELUM. **Java para Desenvolvimento Web Curso FJ-21**. São Paulo, 2014.

COELHO, Hébert. **JPA Eficaz As melhores práticas de persistência de dados em Java**. São Paulo: Casa do Código, 2010.

COELHO, Hébert. **JSF Eficaz As melhores práticas para o desenvolvedor web Java**. São Paulo: Casa do Código, 2010.

CORDEIRO, Giliard. **Aplicações Java para web com JSF e JPA**. São Paulo: Casa do Código, 2010.

ECLIPSE. **Eclipse Logos**. Eclipse. 2014. Disponível em: <<http://www.eclipse.org/artwork/>> Acesso em: 12 maio 2014.

FILHO, Luís Carlos Querino. **Criando aplicações Web com EJB e padrões de projeto**. Devmedia. 2014. Disponível em: <<http://www.devmedia.com.br/criando-aplicacoes-web-com-ejb-e-padroes-de-projeto-revista-easy-java-magazine-28/27480>> Acesso em: 01 jun. 2014.

GONÇALVES, Edson. **Desenvolvendo Aplicações Web com JSP, Servlets, JavaServer Faces, Hibernate, EJB 3 Persistence e Ajax**. 1. ed. São Paulo: Editora Ciência Moderna, 2007.

HIBERNATE. **Logo Hibernate**. Hibernate. 2014. Disponível em:
<<http://hibernate.org>> Acesso em: 22 maio 2014.

LUCKOW, Décio Heinzemann; MELO, Alexandre Altair. **Programação Java para a Web**. São Paulo: Novatec Editora, 2010.

MASTERTHEBOSS. **PrimeFaces vs RichFaces vs IceFaces**. MasterTheBoss. 2014. Disponível em: <<http://www.mastertheboss.com/richfaces/primefaces-vs-richfaces-vs-icefaces>> Acesso em: 05 jun. 2014.

MYSQL. **Logo MySQL**. MySQL. 2014. Disponível em:
<<http://www.mysql.com/about/legal/logos.html>> Acesso em: 25 maio 2014.

MYSQL. **MySQL and the ACID Model**. MySql. 2014. Disponível em:
<<http://dev.mysql.com/doc/refman/5.6/en/mysql-acid.html>> Acesso em: 07 jun. 2014.

PRIMEFACES. **Logo PrimeFaces**. Primefaces. 2014. Disponível em:
<<http://www.primefaces.org>> Acesso em: 24 maio 2014.

RAMOS, Ricardo Argenton. **Treinamento prático em UML. Desenvolva e Gerencie seus projetos com essa sensacional ferramenta**. São Paulo: Digerati Books, 2006.

REBELLABS. **The 2014 Leaderboard of Java Tools & Technologies**. Rebellabs by ZeroTurnAround. 2014. Disponível em: <http://pages.zereturnaround.com/Java-Tools-Technologies.html?utm_source=Java%20Tools%20%20Technologies%202014&utm_medium=allreports&utm_campaign=rebellabs&utm_rebellabsid=88> Acesso em: 25 maio 2014.

SAKURAI, Rafael. **JPA 2.0 – Descrever os conceitos básicos do Mapeamento Objeto Relacional (ORM)**. Universidade Java. 2014. Disponível em:
<<http://www.universidadejava.com.br/docs/jpa20-descreverosconceitosbasicosdomapeamentoobjetorelacionalorm>> Acesso em: 11 jun. 2014.

SOMMERVILLE, Ian. **Engenharia de Software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011.

TOMCAT. **Tomcat Logo**. Tomcat. 2014. Disponível em:
<<http://tomcat.apache.org/legal.html>> Acesso em: 12 maio 2014.

UNIVERSIDADE FEDERAL DO PARANÁ. Biblioteca Central. **Normas para apresentação de trabalhos**. 3. ed. Curitiba: UFPR, 1992. v. 2.

WEBLOGS. **Winner of JSF Spec Logo Contest: Wilber**. Weblogs. 2014. Disponível em: <<https://weblogs.java.net/blog/edburns/archive/2011/05/11/winner-jsf-spec-logo-contest-wilber-saca>> Acesso em: 24 maio 2014.

WEISSMANN, Henrique Lobo. **Vire o jogo com Spring Framework**. São Paulo: Casa do Código, 2012.

WIKIPEDIA. **Adobe Photoshop**. Wikipédia. 2014. Disponível em: <http://pt.wikipedia.org/wiki/Adobe_Photoshop> Acesso em: 22 maio 2014.

WIKIPEDIA. **JSF**. Wikipédia. 2014. Disponível em: <<http://pt.wikipedia.org/wiki/Jsف>> Acesso em: 25 maio 2014.

WIKIPEDIA. **Segurança da informação**. Wikipédia. 2014. Disponível em: <http://pt.wikipedia.org/wiki/Segurança_da_informação> Acesso em: 05 set. 2014.

WIKIPEDIA. **Servlet**. Wikipédia. 2014. Disponível em: <<http://pt.wikipedia.org/wiki/Servlet>> Acesso em: 22 maio 2014.

Apêndices

Apêndice A – Ficha de Entrevista



Universidade Norte do Paraná

Ficha de Entrevista para
Trabalho de Conclusão de
Curso

Informações sobre o Estabelecimento						
Nome do Estabelecimento:						
Endereço:		Cidade:				
Bairro:		CEP:		Estado:		
Telefone:		Responsável:				
Tipo do Estabelecimento						
<input type="checkbox"/>	Albergue	<input type="checkbox"/>	Camping	<input type="checkbox"/>	Apart-Hotel	<input type="checkbox"/>
<input type="checkbox"/>	Hotel	<input type="checkbox"/>	Pousada	<input type="checkbox"/>	Outros: (Qual)	
Quantidade de Acomodações						
<input type="checkbox"/>	1 a 10 quartos	<input type="checkbox"/>	11 a 20 quartos	<input type="checkbox"/>	21 a 30 quartos	<input type="checkbox"/>
Informações sobre o Estabelecimento						
O estabelecimento Possui site:		<input type="checkbox"/> Sim <input type="checkbox"/> Não	Se sim, qual o endereço:			
O estabelecimento usa ou usaria algum sistema de reservas on-line:		<input type="checkbox"/> Sim <input type="checkbox"/> Não	Porque:			
Em sua opinião o que um sistema de Reservas precisa ter						
01-						
02-						
03-						
04-						
05-						
06-						
07-						
08-						
09-						
10-						
Comentários						

Assinatura do Responsável

Data