

**UNIVERSIDADE ESTADUAL DO PIAUÍ - UESPI**  
**CAMPUS PROF. ALEXANDRE ALVES DE OLIVEIRA**  
**BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO**

**ALESSANDRA BRAÚNA DE MEIRELES**

**DESENVOLVIMENTO DE UMA APLICAÇÃO WEB PARA**  
**GERENCIAMENTO DE RESERVA DE RECURSOS UTILIZANDO O**  
**FRAMEWORK DJANGO**

**PARNAÍBA – PI**

**2010**

Handwritten text in the upper middle section.

Handwritten text on the left side.

Handwritten text on the left side.

Handwritten text in the center.

Handwritten text on the right side.

Handwritten text on the left side.

Handwritten text at the bottom left.

Handwritten text at the bottom left.

**ALESSANDRA BRAÚNA DE MEIRELES**

**DESENVOLVIMENTO DE UMA APLICAÇÃO WEB PARA GERENCIAMENTO DE  
RESERVA DE RECURSOS UTILIZANDO O FRAMEWORK DJANGO**

Monografia apresentada ao curso de Bacharelado em Ciência da Computação da Universidade Estadual do Piauí – UESPI, Campus Prof. Alexandre Alves de Oliveira, como parte das exigências da disciplina de Estágio Supervisionado, requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Dario Brito Calçada.

Co-orientador: Eyder Franco Sousa Rios.

**PARNAÍBA – PI  
2010**

FICHA CATALOGRÁFICA ELABORADA PELO BIBLIOTECÁRIO  
HERNANDES ANDRADE SILVA CRB-3/936

M514d Meireles, Alessandra Braúna de

Desenvolvimento de uma aplicação web para gerenciamento de reserva de recursos utilizando o framework Django / Alessandra Braúna de Meireles. – Parnaíba: 2010.

78f : il.

Monografia apresentada ao Curso de Bacharelado em Ciências da Computação, Universidade Estadual do Piauí - UESPI, Parnaíba - 2010.

Orientador: Prof. Dario Calçada.

1. Linguagem de Programação. 2. Framework Django. 3. Python – Linguagem de Programação. I. Título.

CDD – 001.642 4

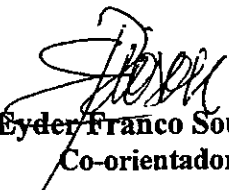


## Ata de Apresentação de Trabalho de Conclusão de Curso

Aos dez dias do mês de setembro de dois mil e dez, às 09h30, na Sala 202 do Campus Prof. Alexandre Alves Oliveira - UESPI, na presença da Banca Examinadora, presidida pelo Prof. Dario Brito Calçada e composta pelos membros efetivos os professores Eyder Franco Sousa Rios e Denival Araújo dos Santos, a aluna **Alessandra Braúna de Meireles** apresentou o Trabalho de Conclusão de Curso intitulado **Desenvolvimento de uma Aplicação Web para Gerenciamento de Reserva de Recursos Utilizando o Framework Django**, como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciência da Computação. Aberta a sessão pública, o candidato teve a oportunidade de expor o trabalho. Após a exposição, o aluno foi arguido oralmente e avaliado em sessão reservada pelos membros da Banca, nos termos do Regulamento Geral dos Trabalhos de Conclusão de Curso da Universidade Estadual do Piauí, tendo obtido nota 8,6 (oito pontos e seis décimos). A Banca concluiu pela **aprovação do candidato, sem restrições**, resultado este divulgado ao aluno e demais presentes. Nada mais havendo a tratar, eu, Prof. Dario Brito Calçada lavrei a presente ata que será lida e assinada por mim, pelos membros da Banca Examinadora e pelo candidato. Parnaíba(PI), 10 de setembro de 2010.

### Banca Examinadora

  
Prof. Esp Dario Brito Calçada, UESPI  
Orientador

  
Prof. MsC Eyder Franco Sousa Rios, UESPI  
Co-orientador

  
Prof. Esp Denival Araújo dos Santos, IFPI

### Candidato

  
Alessandra Braúna de Meireles

À minha família pelo apoio e confiança, em especial à minha mãe por tudo que representa para mim e pela certeza que partilhamos do mesmo sentimento de vitória por mais uma etapa concluída.

## AGRADECIMENTOS

- Agradeço a Deus por ter me dado força, confiança e coragem para prosseguir;
- Ao Professor e Orientador Dario Calçada pela dedicação e apoio;
- Ao Professor, Coordenador e Co-orientador Eyder Rios pela disponibilidade e paciência;
- A todos os professores do curso de Bacharelado em Ciência da Computação;
- Aos grandes amigos conquistados, especialmente a Júlia Theresa, Marcel Moura, Conrado Sampaio, Anderson Passos, Josué Machado e Danilo Borges;
- Aos demais colegas de turma pelo precioso e divertido tempo passado juntos nesses quatro anos;
- Aos amigos da instituição Valquíria e Rodrigo pelo apoio dado durante o período de conclusão desse trabalho;

## RESUMO

Existem inúmeras ferramentas computacionais que são voltadas para a construção de sistemas web e, dependendo da escolha destas, o desenvolvimento pode ser dado de maneira pouco profissional ou custosa. Atualmente a criação de websites cresce rapidamente e para os desenvolvedores se torna uma atividade cansativa, que consome muito tempo e algumas vezes de difícil manutenção. A tecnologia Django foi desenvolvida em 2003, por uma equipe de desenvolvedores web em Kansas, EUA, e a utilização desse framework web facilita o trabalho de programadores, proporcionando a criação rápida e ágil de aplicações com código limpo, de alto desempenho e elegante. A maioria dos problemas encontrados durante a implementação das aplicações web já possuem alguma solução em Django, e o que não é resolvido pelo próprio framework é facilmente programável. Django é uma ferramenta complexa, mas de simples entendimento. É baseado na linguagem de programação Python, que é de alto nível e de código aberto, o que reforça a sua utilização. Ele também faz uso de uma antiga e bastante conhecida linguagem de marcação, presente na estrutura dos códigos de inúmeros sites espalhados pela web, a linguagem HTML.

**Palavras-Chave:** Web. Django. Framework. Python. HTML.



## ABSTRACT

There are several computing tools that are for development of web systems, and depending of the choice, the development can be unprofessional and difficult. Currently the construction of websites grows quickly and this becomes a tiring work for the developers, that consumes a lot of time and sometimes it is hard to keep. The Django technology was developed in 2003, by a team of web developers in Kansas, USA, and the use of this framework facilitates the programmers work, providing a fast and agile creation of applications with a clean, high performance and elegant code. Most of the problems found during the implementation of web applications already have some solution in Django and what is not resolved by the own framework it is easily programmable. Django is a complex tool, but it is simple to understand. It is based in the programming language Python, that is a high level language and open source, what reinforce the use of it. Django uses too an old and very popular markup language, present in the structure of several sites spread in the web, the language HTML.

**Keywords:** Web. Django. Framework. Python. HTML.

## LISTA DE FIGURAS

<b>Figura 01:</b> Comunicação por protocolos .....	18
<b>Figura 02:</b> Requisição e Resposta HTTP.....	19
<b>Figura 03:</b> Tabela de cores .....	28
<b>Figura 04:</b> Código HTML .....	29
<b>Figura 05:</b> Apresentação de documento HTML no navegador .....	30
<b>Figura 06:</b> Transformação em código de byte .....	38
<b>Figura 07:</b> O interpretador Python.....	39
<b>Figura 08:</b> Visão geral do funcionamento Django .....	41
<b>Figura 09:</b> HTTP request e HTTP response .....	45
<b>Figura 10:</b> Administração do Django – login.....	52
<b>Figura 11:</b> Administração do Django – página principal .....	53
<b>Figura 12:</b> Administração do Django – lista de mudanças.....	53
<b>Figura 13:</b> Administração do Django – formulário de edição.....	54
<b>Figura 14:</b> Prompt de comando .....	57
<b>Figura 15:</b> Diretório do projeto.....	58
<b>Figura 16:</b> Tela de boas-vindas do Django.....	59
<b>Figura 17:</b> Interface do banco de dados.....	60
<b>Figura 18:</b> Visualização do arquivo settings.py no Notepad ++ .....	60
<b>Figura 19:</b> Prompt de comando .....	61
<b>Figura 20:</b> Tela inicial da aplicação.....	63
<b>Figura 21:</b> Tela de login para usuário.....	63
<b>Figura 22:</b> Tela de opções do usuário.....	64
<b>Figura 23:</b> Tela exibe uma lista dos recursos .....	64
<b>Figura 24:</b> Tela exibe uma lista com os tipos de recursos .....	65

<b>Figura 25:</b> Tela exibe os recursos relativos ao tipo escolhido.....	65
<b>Figura 26:</b> Tela exibe uma lista das reservas feitas para o recurso escolhido .....	66
<b>Figura 27:</b> Tela inicial da aplicação.....	66
<b>Figura 28:</b> Tela de login do operador .....	67
<b>Figura 29:</b> Tela de opções do operador .....	67
<b>Figura 30:</b> Tela de pesquisa.....	68
<b>Figura 31:</b> Tela exibe um formulário para registro de reservas.....	68
<b>Figura 32:</b> Tela exibe uma lista de todas as reservas feitas.....	69
<b>Figura 33:</b> Tela da administração do Django para a aplicação.....	70
<b>Figura 34:</b> Tela exibe listas de mudanças para objetos da aplicação .....	70
<b>Figura 35:</b> Tela exibe formulário de edição para objeto da aplicação.....	71

## LISTA DE TABELAS

<b>Tabela 01:</b> Sequências de escape .....	25
<b>Tabela 02:</b> Principais tags de formação de texto .....	27
<b>Tabela 03:</b> Símbolos mais comuns das expressões Regulares .....	47
<b>Tabela 04:</b> Comandos do DOS .....	56

## LISTA DE SIGLAS

**ANSI** – American National Standards Institute  
**API** – Application Programming Interface  
**ARPA** – Advanced Research Projects Agency  
**ARPANET** - Advanced Research Projects Agency Network  
**BDFL** – Benevolent Dictator for Life  
**BSD** – Berkeley Software Distribution  
**CERN** – European Organization for Nuclear Research  
**CSS** – Cascading Style Sheets  
**DNS** – Domain Name Service  
**DOS** – Disk Operating System  
**DRY** – Don't Repeat Yourself  
**GPL** – General Public License  
**HTML** – HyperText Markup Language  
**HTTP** – HyperText Transfer Protocol  
**IBM** – International Business Machines  
**IDE** – Interface Development Environment  
**IES** – Instituição de Ensino Superior  
**IETF** – Internet Engineering Task  
**MS DOS** – Microsoft Disk Operating System  
**MVC** – Model-View-Controller  
**NCSA** – National Center Supercomputing Applications  
**NSFNET** – National Science Foundation Network  
**PDA** – Personal Digital Assistant  
**PEP** – Python Enhancement Proposals  
**POO** – Programação Orientada a Objetos  
**PSF** – Python Software Foundation  
**PVM** – Python Virtual Machine  
**RGB** – Red, Green, Blue  
**SGML** – Standard Generalized Markup Language  
**SQL** – Structure Query Language  
**TCP/IP** – Transmission Control Protocol/Internet Protocol  
**URL** – Uniform Resource Locator

**W3C** – World Wide Web Consortium

**WWW** – World Wide Web

**XHTML** – Extensible HyperText Markup Language

**XML** – Extensible Markup Language

## SUMÁRIO

<b>1 INTRODUÇÃO</b>	<b>14</b>
<b>2 A INTERNET</b>	<b>15</b>
2.1 O FUNCIONAMENTO DA INTERNET E DA WEB	16
2.2 HIPERTEXTO E HIPERMÍDIA	17
2.3 O PROTOCOLO HTTP	17
2.3.1 Como o protocolo HTTP trabalha	18
2.4 A URL E O SERVIÇO DE NOMES DNS	19
<b>3 A LINGUAGEM DE MARCAÇÃO HTML</b>	<b>21</b>
3.1 HISTÓRICO	22
3.2 ELEMENTOS BÁSICOS DE UMA PÁGINA HTML	23
3.2.1 Atributos	24
3.2.2 Caracteres de escape	25
3.2.3 Estrutura do documento	25
3.2.4 As cores em HTML	27
3.3 CRIAÇÃO DE UM DOCUMENTO HTML	28
3.3.1 Sobre o código	30
<b>4 A LINGUAGEM DE PROGRAMAÇÃO PYTHON</b>	<b>32</b>
4.1 HISTÓRICO	33
4.2 CARACTERÍSTICAS DA LINGUAGEM	34
4.3 COMPILAÇÃO E INTERPRETAÇÃO	37
4.3.1 O bytecode	37
4.3.2 Python Virtual Machine (PVM)	38
4.3.3 Modos de execução	38
<b>5 O FRAMEWORK DJANGO</b>	<b>40</b>
5.1 HISTÓRICO	40
5.2 VISÃO GERAL DO FUNCIONAMENTO	41
5.3 OS OBJETOS: HttpRequest e HttpResponse	42
5.3.1 HttpRequest	42
5.3.1.1 Entendendo como Django lida com a requisição	42
5.3.2 HttpResponse	44
5.4 O PADRÃO MVC	45
5.4.1 Models	46
5.4.2 URLs e Views	46
5.4.2.1 URLs	46
5.4.2.2 Views	48
5.4.3 Templates	49
5.5 O SITE DE ADMINISTRAÇÃO DO DJANGO	51
<b>6 IMPLEMENTAÇÃO</b>	<b>55</b>
6.1 REQUISITOS	55
6.2 METODOLOGIA	56
6.2.1 Criando um projeto	56
6.2.2 Preparando uma base de dados	59
6.2.3 Criando a aplicação	61

6.2.4 Iniciando a aplicação .....	61
6.2.5 Utilizando a interface de administração do Django .....	62
6.3 O SISTEMA DE GERENCIAMENTO DE RESERVA DE RECURSOS .....	62
<b>7 CONSIDERAÇÕES FINAIS .....</b>	<b>72</b>
<b>REFERÊNCIAS BIBLIOGRÁFICAS .....</b>	<b>73</b>
<b>APÊNDICE A – DIAGRAMA DE CLASSES.....</b>	<b>75</b>
<b>APÊNDICE B – DIAGRAMA DE CASOS DE USO.....</b>	<b>76</b>
<b>APÊNDICE C – DIAGRAMA DE SEQUÊNCIA .....</b>	<b>77</b>
<b>APÊNDICE D – DIAGRAMA DE TRANSIÇÃO DE ESTADO .....</b>	<b>78</b>



## 1 INTRODUÇÃO

Dentro da Universidade Estadual do Piauí, especificamente no Campus Prof. Alexandre Alves de Oliveira (UESPI-Parnaíba) existe um sistema de reserva não automatizado para os recursos disponíveis. Atualmente, a Instituição Superior de Ensino (IES) dispõe dos seguintes recursos: data show, retroprojektor, sala de vídeo, laboratório de informática e auditório. O processo para realizar o empréstimo ou a reserva pode ser considerado confuso, pois cada recurso é tratado num departamento diferente e sob a autorização de funcionários com competências distintas. Nesse sistema, algumas das restrições para determinados tipos de recursos são inviáveis e as regras referentes aos registros de reservas são parcialmente falhas e por isso nem sempre são respeitadas. Além disso, o processo é realizado manualmente, tornando-o ainda mais suscetível a falhas humanas.

Considerando a explanação desses fatores, uma possível alternativa para viabilizar esse esquema de reservas é a implantação de um sistema computacional para fazer esse controle.

Desde a criação da Internet, observa-se que o volume de informações distribuídas pela rede aumenta progressivamente. Como é o meio mais fácil, acessível, rápido e interativo de divulgação, atualmente é desejável que todas as aplicações sejam disponibilizadas via Internet. Por isso foi feita a escolha da tecnologia Django para implementação do software. Django é uma ferramenta – mais especificamente um framework – voltado para a construção de aplicações web. Ele oferece toda uma estrutura que possibilita ao programador desenvolver códigos de alto nível, de forma rápida e pouco custosa.

Nesse trabalho será abordada a análise da implementação de uma aplicação web que utiliza a tecnologia Django, esclarecendo sobre o funcionamento desse framework e a respeito de outras tecnologias e tópicos que são relevantes para a compreensão do mesmo.

## 2 A INTERNET

No final da década de cinquenta, no auge da Guerra Fria, o departamento de defesa dos Estados Unidos concebeu a Advanced Research Projects Agency (ARPA), que tinha como função liderar as pesquisas de ciência e tecnologia aplicáveis às Forças Armadas. Um dos objetivos era desenvolver projetos em conjunto, sem o inconveniente da distância física ou de perder dados e informações de uma base que tivesse sido destruída em combate. Assim, foi criada em 1969 a Advanced Research Projects Agency Network (ARPANET), que interligou computadores de universidades e centros de pesquisa envolvidos com projetos militares. Nesse período, computadores eram raros e o seu uso em rede mais raro ainda. Os primeiros usuários eram cientistas, e usavam a rede para trocar mensagens de correio eletrônico e se conectar de forma remota a outros computadores que se encontravam distantes. O uso dos computadores em rede se mostrou tão útil, que as universidades decidiram interligar seus departamentos, mesmo aqueles que não estavam envolvidos com projetos militares.

Em 1986, foi criada a National Science Foundation Network (NSFNET), que tinha por objetivo prover o acesso a centros de supercomputação e passou a fazer parte da ARPANET. Ela se tornou o *backbone* (espinha dorsal) das duas redes, depois chamada de Internet. “Qualquer centro que achasse seus serviços convenientes podia se amarrar ao nó mais próximo simplesmente pagando as despesas de uma linha dedicada de dados.” (ROCHA, 2000, p. 03).

Dessa maneira a Internet crescia, ainda sem nenhuma organização central, e em pouco tempo já interligava os maiores centros de pesquisa do mundo. A expansão da rede era grandiosa, diferente de sua popularidade que ainda era restrita ao meio acadêmico. Utilizar os serviços da rede era uma tarefa complexa, que exigia conhecimentos específicos.

Em 1992, surgiu a World Wide Web (WWW) que propiciou o crescimento explosivo da Internet. Esse era a evolução de um projeto desenvolvido por Tim Berners-Lee, no European Organization for Nuclear Research (CERN) em Genebra. Trata-se de um laboratório Europeu para Física de Partículas, uma das maiores instituições científicas do mundo, com laboratórios distribuídos em várias cidades localizadas em dezenove países na Europa. Berners-Lee demonstrou como a informação se perdia diariamente no CERN, ambiente classificado por ele como “um modelo em miniatura do resto do mundo em alguns anos”. O sistema era inicialmente nomeado como Mesh, mas recebeu depois o nome de World Wide Web. Foi através do hipertexto que a Web finalmente pode organizar suas informações.

No ano seguinte, foi desenvolvido o primeiro navegador, o Mosaic. Esse programa foi desenvolvido por um grupo de estagiários do National Center Supercomputing Applications (NCSA) da Universidade de Illinois, baseado numa versão anterior para o sistema Unix. O Mosaic proporcionava pela primeira vez ao usuário uma interface gráfica multimídia para Web. Depois surgiram versões para os sistemas operacionais Windows e Macintosh, isso trouxe um grande número de usuários domésticos para dentro da rede.

Depois de deixarem a universidade, os criadores do Mosaic criaram uma empresa: a Netscape, provavelmente a empresa que teve maior influência nos rumos seguidos pela Web durante sua evolução até os dias atuais.

Ela é capaz de servir de porta de entrada não só a todos os serviços de voz (telefone), televisão, rádio e mídias impressas, sem falar do impacto que está tendo diretamente nos hábitos da sociedade, mudando as regras do comércio e das relações humanas. (ROCHA, 2000, p. 05).

A web possui uma grande importância na história das telecomunicações. Diferente dos outros meios tradicionais de comunicação, ela é considerada uma mídia democrática, pois qualquer indivíduo tem o poder de informação, ou seja, qualquer um que tenha acesso à rede pode receber e publicar informação.

## 2.1 O FUNCIONAMENTO DA INTERNET E DA WEB

Os termos Web e Internet são normalmente confundidos, mas eles possuem significados distintos. A World Wide Web é o nome mais popular dos serviços da Internet, enquanto esta pode ser definida como um conjunto de computadores, provedores de acesso, satélites, cabos e serviços que constituem uma rede mundial, regida por protocolos de comunicação conhecidos como Transmission Control Protocol/Internet Protocol (TCP/IP).

Os servidores Web têm a função de disponibilizar as páginas via web. Um computador conectado à Internet pode utilizar uma aplicação chamada web browser ou simplesmente navegador, ou seja, um programa que é capaz de conectar um servidor e solicitar alguma informação. Ao receber a solicitação, o servidor a procura dentro do seu sistema e devolve uma resposta para o computador que fez a solicitação, chamado de cliente. (PFAFFENBERGER; SCHAFFER; WHITE; KAROW, 2004). Essas máquinas devem falar uma linguagem em comum chamada HyperText Transfer Protocol (HTTP), contudo, não é realmente uma linguagem, mas um protocolo.

## 2.2 HIPERTEXTO E HIPERMÍDIA

Essa é uma abordagem mais dinâmica para organização de informações. Ela traz uma série de benefícios para o leitor/usuário. Por exemplo, pode-se ter uma lista de tópicos que servem como notas explicativas ou material de referência descritiva.

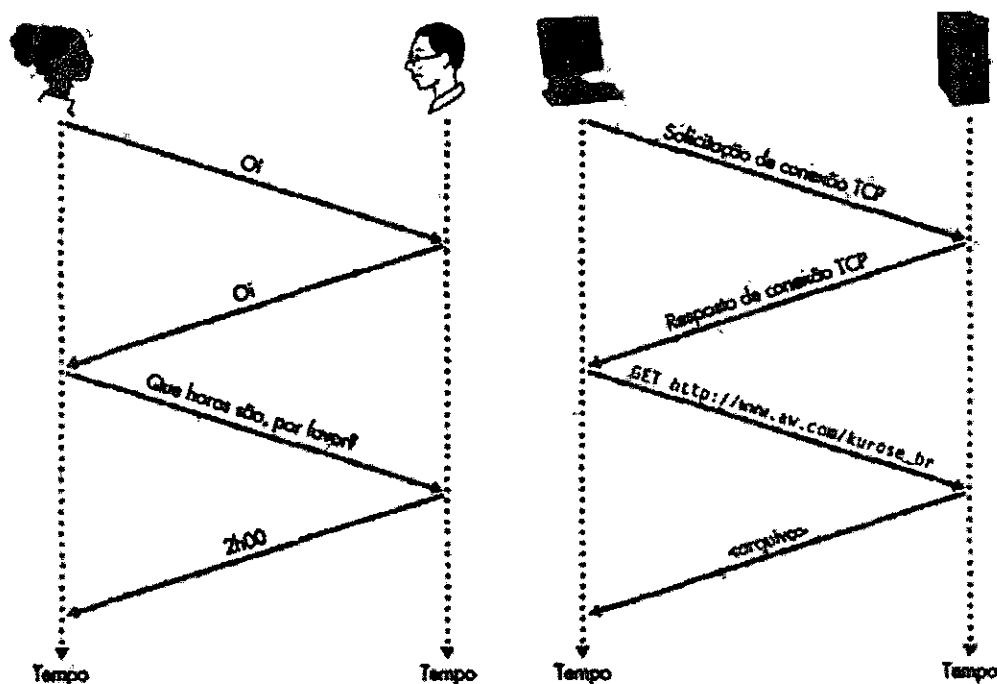
Esse termo foi criado por um pioneiro da computação, Theodore Nelson. Ele é uma analogia a um conceito de ficção científica — o salto de hiperespaço — que apesar de fisicamente impossível, permitiria a uma nave ir imediatamente de um sistema estelar para outro. O hipertexto ou *hypertext* é um tipo de texto que contém hiperlinks ou *hyperlinks* que definem uma forma não linear de publicar informações, onde as palavras do próprio texto podem levar a outras ações, como a outro documento, outra seção do mesmo documento ou até outro sistema de informação. Ele é baseado na ligação entre dois pontos chamados âncoras e essas ligações entre as âncoras são nomeadas de vínculos ou simplesmente links. Os links são implementados dentro da estrutura do texto, através da utilização da HyperText Markup Language (HTML). A identificação de links em um documento web é simples, normalmente trata-se de um termo ou uma frase sublinhada com uma coloração destacada do restante do texto, ou ainda pode ser utilizada uma imagem para realizar a função.

O sistema de hipermídia trabalha da mesma forma que o de hipertexto, com a diferença de incluir além de texto, também gráficos, sons, vídeos e animações. A vantagem trazida pelo hipertexto é que diferente da maneira sequencial de ler um livro, por exemplo, ele oferece ao usuário a habilidade de escolher o próprio caminho através do material exibido, podendo escolher as partes do texto que mais interessam e se direcionar para elas.

A Web pode ser vista como um grande computador baseado em um sistema de hipermídia e a ação de “pular” entre uma página e outra é chamada de *surfing* (surfear).

## 2.3 O PROTOCOLO HTTP

“Um protocolo define o formato e a ordem das mensagens trocadas entre duas ou mais entidades comunicantes, bem como as ações realizadas na transmissão e/ou no recebimento de uma mensagem ou outro evento.” (KUROSE, 2005, p. 29)

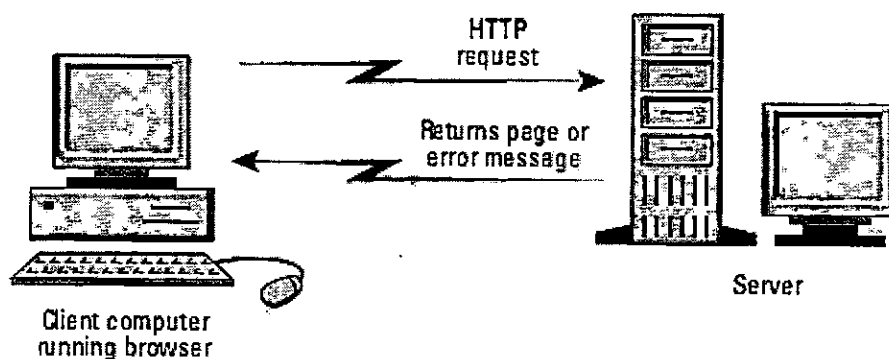


**Figura 1:** Comunicação por protocolos  
 Fonte: *Web design e HTML avançado*. 2000.

A plataforma web é um meio virtual composto por servidores HTTP que são máquinas que mantêm os sites, clientes HTTP que podem ser representados pelos navegadores e o protocolo HTTP que é a linguagem entre servidor e cliente. Ele é implementado em dois programas, um cliente e outro servidor. Os dois programas são executados em sistemas finais distintos e conseguem se comunicar através da troca de mensagens HTTP. Esse protocolo define a estrutura das mensagens e o modo como serão trocadas.

### 2.3.1 Como o protocolo HTTP trabalha

- A maioria das páginas web possui hiperlinks, que são palavras formatadas que possibilitam ao usuário ter acesso a outras páginas. Ele contém toda a informação necessária para que a página seja solicitada a outro computador.
- Quando o hiperlink é acionado a máquina do usuário envia uma mensagem chamada de *HTTP request* para outra máquina, ou seja, o servidor.
- O servidor recebe essa solicitação e faz a procura em seus arquivos. Ao ser encontrada, a página é enviada à máquina do usuário (cliente) e o navegador a exibe na tela. Caso não seja encontrada, é enviada uma mensagem de erro.



**Figura 2:** Requisição e Resposta HTTP  
*Fonte: HTML, XHTML, and CSS Bible. 2004.*

## 2.4 A URL E O SERVIÇO DE NOMES DNS

O Domain Name Service (DNS) é um dos serviços fundamentais da Internet. Cada computador possui um endereço IP, no entanto, localizar uma máquina pela Internet através de um número IP é algo inviável. Por isso cada endereço está associado a um nome, esse serviço é chamado de DNS e é oferecido por várias máquinas espalhadas pela Internet, que guardam tabelas onde associam o nome de uma máquina ou de uma rede a um endereço IP. Ao digitar o nome da máquina no navegador, por exemplo, “www.exemplo.com.br”, o navegador procura localizá-lo em outra máquina cujo IP ele já conhece e que oferece o serviço de nomes. Essa máquina consulta outros serviços de nomes na Internet e então devolve o endereço IP à solicitante. Caso haja falha no sistema de nomes, o navegador não terá o endereço IP solicitado e não poderá localizar a máquina correspondente, mesmo que ela não esteja fora do ar.

A função da Uniform Resource Locator (URL) é a identificação das páginas na Internet. A cada página é atribuída uma URL, que representa o nome universal dessa página. Elas são compostas por três partes: o protocolo, o nome DNS da máquina onde a página se encontra e um nome local que indica a página específica. Como pode ser identificado em seguida:

<http://www.clubedohardware.com.br/artigos/1351>

O protocolo é o (HTTP), o nome DNS do host é ([www.clubedohardware.com.br](http://www.clubedohardware.com.br)) e o nome do arquivo é (artigos/1351), que é um caminho relativo ao diretório da web padrão em

(clubedohardware.com.br). Alguns sites possuem atalhos internos, por exemplo, quando o nome do arquivo for nulo ele é redirecionado para a *homepage* (página principal) da organização referente, que pode ter sido definida como padrão.

### 3 A LINGUAGEM DE MARCAÇÃO HTML

A utilização da HyperText Markup Language possibilita o uso da função de hipertexto na Web. O termo “marcação” é relacionado à impressão de páginas. Os editores marcavam páginas manuscritas com símbolos para informar à impressora como as páginas deveriam ser impressas. (PFAFFENBERGER; SCHAFER; WHITE; KAROW, 2004).

A HTML é uma linguagem de marcação de hipertexto, simples e composta de marcações de formatação e diagramação de hipertexto/hipermídia, e por isso é conhecida como a linguagem da Internet. Ela é definida como “a linguagem universal da Web” (Rocha, 2000, p. 12), e permite que a informação contida nas páginas web possa ser acessada em máquinas de arquiteturas e sistemas operacionais distintos. “Linguagens de marcação são linguagens que “marcam” um texto simples, então ele é formatado e exibido em um navegador web de modo interessante e útil.” (Ford, 2009, p. 27).

Os termos HTML e HTTP foram criados por Tim Berners-Lee, que trabalhava no Instituto Suíço de Pesquisa como especialista em computação e redes. Ele tinha o objetivo de dar às pesquisas do instituto uma linguagem de marcação para que fosse possível compartilhá-las pela Internet. Logo, Berners-Lee baseou a linguagem HTML num padrão internacional de marcação – Standard Generalized Markup Language (SGML). A idéia básica desse padrão é a de manter separadas estrutura e apresentação de um documento. Essa divisão é benéfica, porque assim programadores dedicam sua atenção a estrutura do código, enquanto designers se preocupam com a apresentação do documento, a parte relativa à apresentação pode ser alterada rapidamente sendo mais simples e fácil de manter.

HTML é uma linguagem derivada e completamente especificada a partir da linguagem SGML. No entanto, não é preciso sabê-la para usar HTML. Ela não é uma linguagem de programação como C, Python, Java, dentre outras que permite a criação de algoritmos, no entanto, é uma linguagem declarativa que tem por função organizar informações em um arquivo de texto, para posteriormente ser visualizado por um browser. A linguagem HTML serve para definir a aparência (formatação) dos documentos na web. Apesar de haverem outras linguagens usadas de forma concorrente, ela é ainda a base para as páginas web e interpretada por todos os navegadores disponíveis.

Na construção de uma página Web é usado um arquivo de texto simples. Mesmo que essa página exiba vários recursos, há por trás uma página de texto e outros arquivos separados que foram montados pelo navegador. O código é texto, mas não somente texto. Ele é todo marcado com HTML, que é responsável por definir toda a estrutura da página para que o



navegador possa formatá-la e exibi-la de modo correto. O arquivo HTML tem seus elementos definidos usando “etiquetas de markup”, ele é repleto de marcadores que se destacam através dos caracteres especiais: “ < ” e “ > ”. Essas etiquetas são responsáveis por dar ao browser as instruções necessárias sobre a estrutura do documento e sobre a forma como a página deve ser apresentada graficamente.

Para o desenvolvimento de um código em HTML, faz-se necessário somente dois tipos de software básicos: um editor de textos e um navegador. Um documento HTML é um arquivo de texto comum, escrito com o editor de textos e salvo com a extensão .htm ou .html. Existem variados editores de textos que podem ser utilizados de acordo com a preferência do programador. Para usuários do sistema Windows, é aconselhável o uso do Bloco de Notas (Notepad).

Com a utilização de HTML, publicam-se documentos estruturados na rede, recupera-se informação por vínculos de hipertexto, projeta-se uma interface interativa com formulários para acesso a serviços remotos, como buscas e comércio eletrônico, e também podem ser incluídas imagens, vídeos, sons, animações e outras aplicações interativas dentro de documentos visíveis no browser.

### 3.1 HISTÓRICO

O HTML surgiu no início dos anos 90 e serviu durante muito tempo como a linguagem de marcação padrão para a Internet. Sua popularidade cresceu por consequência do surgimento de vários browsers que a utilizavam. Dessa forma, grupos de trabalho foram sendo criados com a intenção de padronizar especificações para o HTML. O World Wide Web Consortium (W3C), que é composto por representantes de várias empresas de tecnologia com o objetivo de desenvolver regras para a criação e interpretação de conteúdos para a Web. Todos os desenvolvedores devem seguir os padrões de acessibilidade do W3C, para que não sejam criadas barreiras tecnológicas e as páginas possam continuar a serem devidamente acessadas.

Em 1995, surgiu a versão 2.0 do HTML, a primeira a ser recomendada pelo Internet Engineering Task (IETF), se tornando o padrão da Internet. Era uma linguagem simples que dizia como o browser deveria estruturar a página, contudo, não definia como as listas e parágrafos deveriam aparecer graficamente.

Devido à expansão da Web na época, o desenvolvimento do HTML 3.0 não foi aprovado, pois não acompanhava as tendências do mercado, que exigiam maiores recursos de apresentação gráfica.

Em 1997, foi aprovada uma versão HTML 3.2, que introduzia recursos de apresentação gráfica. No entanto, a maioria desses recursos gráficos eram incorporações de extensões proprietárias das empresas Netscape e Microsoft, o que contradizia a filosofia do HTML de garantir a compatibilidade da linguagem mesmo em diferentes plataformas. Essa situação acabava por atrasar a criação de ferramentas de desenvolvimento eficientes, pois não era possível validar a linguagem HTML para plataformas que não suportavam alguns recursos gráficos mais sofisticados.

A filosofia que cercava o desenvolvimento do HTML era para que qualquer dispositivo pudesse ter acesso às informações da web. Incluindo computadores com monitores gráficos de diversas resoluções, telefones celulares, dispositivos geradores de voz, e outros. O HTML 3.2 prejudicava essa filosofia, então, após muitas discussões as empresas entraram em acordo e desenvolveram uma versão 4, que incluía mais recursos visando um acesso universal à informação da web, por exemplo, recursos de acessibilidade para pessoas com deficiência, suporte a convenções internacionais (outras línguas ou alfabetos), separação da estrutura, conteúdo e apresentação, recursos interativos do lado do cliente e otimização em tabelas e formulários.

Vários elementos do HTML 3.2 foram sendo considerados fora de uso pelo HTML 4. Notadamente, o HTML nunca realizou muito bem o trabalho de formatação gráfica de uma página, pois ele foi criado para estruturar o conteúdo dos documentos. Dessa maneira, a solução encontrada foi deixar a cargo da linguagem HTML a marcação do texto e deixar a uma nova linguagem — Cascading Style Sheets (CSS) — o tratamento da estrutura de apresentação, ou seja, a aparência do que será exibido na tela. A linguagem CSS permite a criação de folhas de estilo, que são aplicáveis a várias páginas de um site.

### 3.2 ELEMENTOS BÁSICOS DE UMA PÁGINA HTML

Quando o navegador lê um documento HTML, ele procura interpretar as seqüências de caracteres entre os símbolos “<” e “>”. Dessa forma, o navegador as considera como descritores ou simplesmente *tags* e não são exibidas na tela. Essas tags servem para estruturar a página a ser apresentada, obedecendo alguma regra previamente definida.

A maioria dos elementos possui um descritor inicial e um final, que cerca o texto marcado por eles, esse texto passa a ter uma função.

`<elemento>texto marcado</elemento>`

Há blocos de textos marcados que podem conter outros blocos. Nesse caso, o bloco mais interno deverá ser fechado antes do descritor de fechamento do bloco mais externo.

```
<h1>Texto <b> muito <i> importante</b></i> para todos</h1>
```

Não são todos os elementos HTML que podem conter outros descritores, e os que podem possuem regras para definir quais elementos são permitidos. Há casos de elementos que não precisam ter descritores de fechamento, como é o caso do <p> e </p>, ele pode ser omitido.

```
<p> Parágrafo ou <p> Parágrafo </p>
```

Há elementos que não podem conter texto ou outros descritores, devendo ser vazios e sem descritor final. Nesse caso, a quebra de linha <br> marca a posição onde a linha deverá ser quebrada.

```
A frase continua <br> na outra linha
```

### 3.2.1 Atributos

Alguns dos elementos podem ter um ou vários atributos, que são ou não opcionais. Eles têm a função de modificar alguma propriedade do texto marcado ou acrescentar uma informação necessária.

```
propriedade = "valor"
```

Os atributos aparecem somente no descritor inicial e logo após o nome do elemento:

```
<h1 align = "center" >Texto centralizado</h1>
```

A ordem na qual esses atributos aparecem não tem importância, contanto que estejam no descritor inicial. Em alguns casos, o uso de atributos é obrigatório, como os atributos HREF e SRC, que criam vínculos de hipertexto e imagens, respectivamente.

```
<a href = "http://www.google.com.br" >Clique aqui</a>
```

```
<img src = "imagem.jpg" width = "150" height = "96" border =
      "1">
```

### 3.2.2 Caracteres de escape

Os caracteres "<" e ">" por definirem os descritores, não são impressos na tela pelo navegador. Quando é necessário exibi-los, deve-se fazer uso de uma *seqüência de escape*. Essa seqüência é iniciada pelo símbolo "&" seguido de uma abreviação e um ponto e vírgula, que indica o final. Como o "&" também é um caractere especial, há uma seqüência para produzi-lo. As principais seqüências de escape para produzir "<", ">", "&" e aspas são:

Caractere	Seqüência de escape
<	&lt;
>	&gt;
&	&amp;
"	&quot;

**Tabela 1:** Seqüências de escape

Para produzir a linha,  $144 < 25 + x < 36 + y$ , no navegador, é preciso digitar o seguinte código HTML no editor de textos:

```
144 &lt; 25 + x &lt; 36 + y
```

### 3.2.3 Estrutura do documento

Há uma estrutura básica para uma página em HTML, definida de acordo com a especificação padrão e deve ser respeitada para garantir compatibilidade com o maior número de navegadores.

```
<!DOCTYPE HTML Public "-//IETF//DTD HTML 4.0//EN" -->
<HTML>
<HEAD>
<TITLE> Descrição do documento </TITLE>
```

Aqui ficam blocos de controle, folha de estilo, funções de scripts, informações de indexação, atributos que afetam todo o documento, etc.

```
</HEAD>
```

```
<BODY>
```

Toda a informação visível da página vem aqui.

```
</BODY>
```

```
</HTML>
```

A linha `<!DOCTYPE HTML Public "-//IETF//DTD HTML 4.0//EN" -->` é um descritor SGML, que informa ao navegador que ele deve interpretar o documento de acordo com a definição do HTML versão 4.0.

Os elementos `<HTML>` e `</HTML>` marcam o início e o final do documento e devem conter duas subestruturas: o cabeçalho, delimitado por `<HEAD>` e `</HEAD>`, e o corpo do documento, entre os descritores `<BODY>` e `</BODY>`.

O bloco do cabeçalho, marcado por `<HEAD>` e `</HEAD>` pode conter informações sobre o conteúdo do documento, mas não contém a informação que será exibida na página. O `<TITLE>` é o único elemento obrigatório do bloco do cabeçalho. Deve conter o título do documento que aparece fora da página, na barra de título do navegador. Ele deve conter informações que descrevam o documento.

```
<TITLE>Introdução</TITLE>
```

O bloco marcado por `<BODY>` e `</BODY>` contém a parte do documento onde será colocada a informação que efetivamente será mostrada e formatada na tela pelo navegador.

Elementos HTML podem ser de vários tipos e têm regras específicas sobre o que podem conter e onde podem ser usados. Quanto à estrutura que assumem na página, podem ser classificados como:

- Elementos de *bloco* (contém texto ou agrupam outros elementos de bloco)
- Elementos *inline* (em linha – usados dentro do texto)
- Elementos de *lista* (usados para construir listas)
- Elementos de *tabela* (usados para construir tabelas)
- Elementos de *formulário* (usados para construir formulários)
- Elementos para embutir *objetos* (usados para incluir imagens, vídeos, etc.)

Todos os elementos referentes à formatação do texto da página devem estar presentes no bloco `<BODY>`.

Tags	Descrição
<code>&lt;p&gt; texto &lt;/p&gt;</code>	Cria um parágrafo
<code>&lt;center&gt; texto &lt;/center&gt;</code>	Centraliza o texto
<code>&lt;b&gt; texto &lt;/b&gt;</code>	Texto em negrito
<code>&lt;i&gt; texto &lt;/i&gt;</code>	Texto em itálico
<code>&lt;u&gt; texto &lt;/u&gt;</code>	Texto sublinhado
<code>&lt;tt&gt; texto &lt;/tt&gt;</code>	Textos no estilo da máquina de escrever
<code>&lt;sub&gt; texto &lt;/sub&gt;</code>	Texto subscripto
<code>&lt;sup&gt; texto &lt;/sup&gt;</code>	Texto sobrescrito
texto <code>&lt;br&gt;</code>	Quebra de linha
<code>&lt;!-- texto --&gt;</code>	Tag para usar comentários no código e é ignorada pelo navegador
<code>&lt;dd&gt; texto</code>	Cria parágrafos (espaço para primeira linha)

**Tabela 2:** Principais tags de formatação de texto

### 3.2.4 As cores em HTML

Em HTML, as cores podem ser definidas através de um nome ou um número em hexadecimal, representando o código Red, Green, Blue (RGB) da cor correspondente. Contudo, ainda podem ser especificadas por três números decimais, que também representam o código RGB.

O código RGB informa a quantidade de luz vermelha, azul e verde que compõem a cor. Cada cor pode ter dezesseis níveis de intensidade, que varia de 0 a 256 (00 a FF, em hexadecimal). A exibição correta das cores depende da capacidade do vídeo onde serão visualizadas. Poucos sistemas são capazes de exibir mais de 65.536 cores simultâneas, a maioria exibe somente 256.

A tabela apresenta dezesseis nomes de cores padrão, suportados por todos os browsers que exibem cores, com seus respectivos códigos RGB em decimal e hexadecimal.

Cor	Nome	Cód. Decimal	Cód. Hexa	Cor	Nome	Cód. Decimal	Cód. Hexa
	red	255, 0, 0	ff0000		maroon	128, 0, 0	800000
	lime	0, 255, 0	00ff00		green	0, 128, 0	008000
	blue	0, 0, 255	0000ff		navy	0, 0, 128	000080
	yellow	255, 255, 0	ffff00		olive	128, 128, 0	808000
	aqua	0, 255, 255	00ffff		teal	0, 128, 128	008080
	fuchsia	255, 0, 255	ff00ff		purple	128, 0, 128	800080
	white	255, 255, 255	ffffff		silver	192, 192, 192	c0c0c0
	black	0, 0, 0	000000		gray	0, 0, 0	808080

Figura 3: Tabela de cores

Fonte: *Web design e HTML avançado*. 2000.

### 3.3 CRIAÇÃO DE UM DOCUMENTO HTML

Existem muitos editores especializados na criação de códigos em HTML, essas ferramentas apresentam um ambiente gráfico para a criação de páginas e possibilitam ao usuário utilizá-las mesmo sem o conhecimento da linguagem. Contudo, os códigos gerados são deficientes, pois apresentam erros e imperfeições que para serem corrigidos, necessitam que o usuário tenha alguma noção de HTML. Esses tipos de ferramentas são úteis para acelerar o desenvolvimento de páginas, logo, a melhor maneira de usá-las é ter o conhecimento básico da linguagem, para que, quando necessário, sejam feitas as otimizações no código. Muitos usuários utilizam o editor de texto Notepad para o desenvolvimento de páginas web, pois ele está presente em todas as máquinas que usam o sistema Windows. Ele pode ser considerado um editor fraco por não possuir algumas funcionalidades apresentadas por outros, como *highlighting* e formatações pré-definidas. Boas opções podem ser PHP Editor ou Notepad ++.

```

1 <html>
2 <head>
3 <title>exemplo</title>
4 </head>
5 <body>
6     Primeira página web!
7 </body>
8 </html>

```

82 chars 96 bytes 8 lines Ln: 1 Col: 1 Sel: 0 (0 bytes) in 0 rãnges Dos\Windows ANSI INS

Figura 4: Código HTML

Fonte: Primária.

#### Exemplo de código em HTML:

```

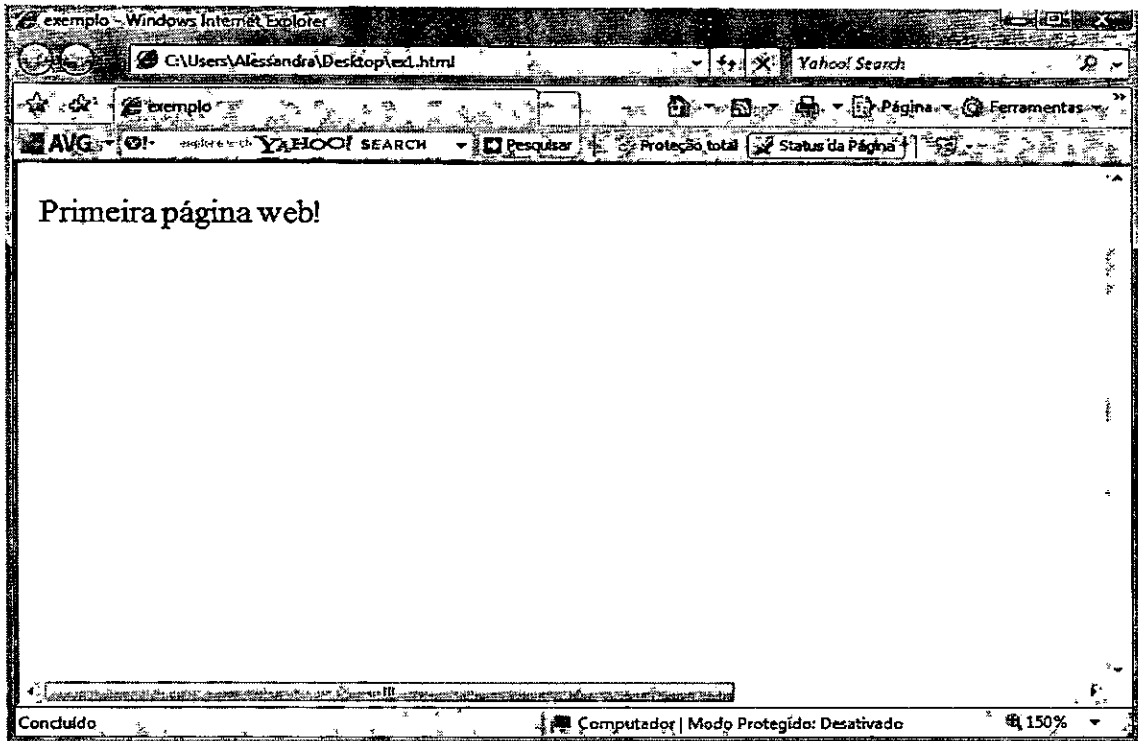
<html>
<head>
<title>exemplo</title>
</head>
<body>
    Primeira página web!
</body>
</html>

```

Os arquivos escritos em HTML devem ter a extensão .html. No entanto, a extensão .htm já foi bastante utilizada. Ela foi uma herança do Microsoft Disk Operating System (MS DOS) e do Windows 3.11 de 16 bits, cujos nomes dos arquivos não podiam exceder oito caracteres e suas extensões tinham, no máximo, três caracteres.

Para executar esse arquivo, não é necessário a utilização de um servidor. Tendo sido salvo com a extensão .html, ele pode ser aberto pelo navegador padrão do sistema, que exibe a página. O servidor Web mais popular é o Apache, ele é disponível para várias arquiteturas e de uso livre. O resultado obtido com a execução desse código é:





**Figura 5:** Apresentação de documento HTML no navegador

*Fonte: Primária.*

### 3.3.1 Sobre o código

A primeira etiqueta é `<html>`: sua função é definir o elemento raiz do documento que contém toda a definição da página. Ela indica ao navegador para iniciar um novo documento HTML cujo conteúdo está entre essa etiqueta de início e a de finalização `</html>`, que marca o fim do documento (página).

As etiquetas `<head></head>` define o cabeçalho do documento. Não há representação gráfica, mas indica sobre aquilo que a página contém e sobre a forma como ela deve ser apresentada graficamente.

As etiquetas `<title></title>` definem o título do documento. Ele fica representado na janela do navegador.

Entre as etiquetas `<body></body>` fica representado todo o corpo da página, e definem tudo que o navegador deve apresentar graficamente.

Os nomes das etiquetas podem ser escritos com letra maiúscula ou minúscula. A nova geração do HTML é uma aplicação da Extensible Markup Language (XML) e designa-se por Extensible HyperText Markup Language (XHTML). O XHTML é a melhor linguagem para criar páginas para a Web, mas é mais restritiva do que o HTML. Isso significa que algumas etiquetas em XML possuem diferença quando escritas em letra maiúscula ou minúscula, logo,

em XHTML adotou-se a convenção de escrevê-las com letra minúscula. Por essa razão é preferível manter esse padrão mesmo na escrita de páginas com código HTML, pois possíveis erros serão evitados se for preciso convertê-lo para XHTML.

## 4 A LINGUAGEM DE PROGRAMAÇÃO PYTHON

“A linguagem Python é destacada entre as linguagens dinâmicas como uma das mais populares e poderosas, com uma grande comunidade de usuários espalhados pelo mundo.” (Borges, 2010, p. 11).

Python possibilita facilidade no entendimento dos seus conceitos fundamentais e por isso é considerada simples, no entanto, possui uma base teórica e técnica muito complexa. Ela se torna uma linguagem produtiva por possuir sintaxe clara e concisa, o que favorece a legibilidade do código-fonte.

De acordo com Lutz (2007, p. 11): “Python é uma linguagem orientada a objetos utilizada em uma variedade de domínios, tanto para programas independentes como para aplicações de script. Ela é gratuita, portátil, poderosa e fácil de usar.”

Antigamente, as linguagens dinâmicas eram vistas como linguagens *script*, mas com o passar do tempo e contribuição de fatores como a internet, o software de código aberto e as metodologias ágeis de desenvolvimento de software, foi possível o crescimento dessas linguagens no mercado. A internet promove de forma intensa o compartilhamento de informações, isso possibilitou o crescimento do software de código aberto e em decorrência disso, nasceram as metodologias ágeis de desenvolvimento. As linguagens dinâmicas são normalmente de código aberto, e compartilham as mesmas funcionalidades. Por sua produtividade e expressividade essas linguagens se adequam perfeitamente às metodologias ágeis.

Python é uma linguagem interpretada, significando que não é preciso a compilação do seu código-fonte, pois ele é primeiramente executado por um interpretador que em seguida é executado pelo sistema operacional ou processador. Apesar de nem toda linguagem interpretada ser uma linguagem de script, Python também pode ser considerada uma linguagem de script, mas para afirmar esse conceito é necessário que sejam consideradas três diferentes definições: ferramenta de Shell, linguagem de controle e facilidade de uso.

As ferramentas de Shell são utilizadas para desenvolver scripts orientados a sistemas operacionais. Os programas são executados a partir de linhas de comando no console. O Python serve a essa função, mas ela é somente uma de suas muitas aplicações.

Por ser simples, a linguagem se torna uma ferramenta de controle naturalmente flexível. Por isso os programas em Python normalmente são distribuídos no contexto de um aplicativo maior. Esses podem executar trechos de código em Python para realizar determinadas tarefas.

A sua facilidade de uso é a melhor maneira para admitir Python como uma linguagem de script. Por ser simples, fácil e flexível, a linguagem permite que o desenvolvimento de programas seja mais rápido do que em linguagens compiladas. Isso não significa que Python se enquadra somente para a resolução de tarefas simples, pois o que ele possui é a capacidade de simplificar as tarefas para melhorar o processo de desenvolvimento. Os recursos oferecidos pelo Python podem tornar os programas mais sofisticados, de acordo com a necessidade do programador.

A escolha da melhor ferramenta para desenvolvimento costuma ser baseada na preferência pessoal ou em limitações específicas da própria linguagem de programação. Em relação ao uso do Python, é possível destacar algumas características vantajosas, como: qualidade do software, produtividade, portabilidade, bibliotecas de suporte e integração de componentes.

#### 4.1 HISTÓRICO

A linguagem de programação Python foi criada em 1990, por Guido van Rossum, no Instituto Nacional de Pesquisa para Matemática e Ciência da Computação da Holanda, e tinha seu foco original em usuários como físicos e engenheiros. O nome Python escolhido para a linguagem foi uma referência tirada de um programa da TV Britânica, o grupo de comédia favorito de seu criador, *Monty Python's Flying Circus*.

Guido van Rossum foi programador do grupo ABC, uma linguagem de programação da época, após alguns anos o projeto ABC foi encerrado devido à falta de sucesso e Rossum passou a integrar o grupo Amoeba, um sistema distribuído baseado em microkernel. Sua motivação para o desenvolvimento de Python foi a necessidade da existência de uma linguagem de alto nível no projeto Amoeba.

Em 1991, Guido publicou o código (versão 0.9.0) no grupo de discussão *alt.sources*. Nessa versão já estavam presentes classes com herança, tratamento de exceções, funções e os tipos de dado nativos *list*, *dict*, *str*. Também estava presente nessa versão um sistema de módulos emprestado do Modula-3. O modelo de exceções também lembrava muito o do Modula-3, com a adição da opção *else clause*.

Em 1994 foi formado o principal fórum de discussão do Python, o *comp.lang.python*, um marco para o crescimento da base de usuários da linguagem. Lançada em janeiro de 1994, a versão 1.0 incluía ferramentas para programação funcional. A versão Python 1.2 foi criada enquanto Guido estava na CWI. Na versão 1.4 a linguagem ganhou parâmetros nomeados,

que é a capacidade de passar parâmetros pelo nome e não pela posição na lista de parâmetros, e suporte nativo a números complexos, assim como uma forma de encapsulamento.

A comunidade de usuários criou expressões para serem referidas a assuntos da linguagem. Algumas delas como *Pythonic* indica que algo é compatível com as premissas do projeto Python, logo, *Unpythonic* significa o oposto. O próprio usuário é definido como *Pythonist*. As metas do projeto foram definidas por Tim Peters em um texto nomeado *Zen of Python*, disponível dentro do código Python e pode ser acessado com o uso de um comando através do console do sistema.

Esse texto enfatiza a postura pragmática de Guido, conhecido pela comunidade como Benevolent Dictator for Life (BDFL). As propostas para a melhoria da linguagem são conhecidas como Python Enhancement Proposals (PEPs) e servem de referência para novos recursos a serem implementados na linguagem.

Atualmente existem várias versões, todas disponíveis no site do desenvolvedor, as versões atuais são a Python 2.7 e a 3.1.2, lançada em 21 de março de 2010.

## 4.2 CARACTERÍSTICAS DA LINGUAGEM

A sintaxe da linguagem é clara e fácil de aprender e a qualidade do software é alta, pois o código é projetado para ser legível, logo, é fácil de ser mantido. A linguagem também oferece suporte ao mecanismo de reutilização de código, com a Programação Orientada a Objetos (POO). Python é uma linguagem completamente orientada a objeto. Todos os tipos básicos em python são objetos e seu modelo de classes suporta noções como o polimorfismo, sobrecarga de operadores e herança múltipla. A linguagem inclui estruturas de alto nível, como listas, dicionários, tuplas, strings, arquivos, classes, entre outros, e também possui módulos de bibliotecas padrão, que é um conjunto de funções pré-compiladas e portáveis. A sua utilização aumenta a produtividade do desenvolvedor, pois na maioria dos casos, um programa em Python será mais curto que seu correspondente em outra linguagem. Isso aumenta a velocidade de desenvolvimento e reduz o potencial de defeitos, visto que com menos código há menores chances de cometer erros.

Ela é deliberadamente otimizada para velocidade de desenvolvimento – sua sintaxe simples, de tipagem dinâmica, ausência de etapas de compilação e seu conjunto de ferramentas incorporado, permitem que os programadores desenvolvam programas em uma fração do tempo de desenvolvimento necessário para algumas outras ferramentas. O resultado é que o Python aumenta a produtividade do desenvolvedor

muitas vezes além do que se consegue com as linguagens tradicionais. (LUTZ, 2007, p. 35).

Python é um software de código aberto, logo, pode ser obtido de forma gratuita na Internet. Sua licença é compatível com a General Public License (GPL), porém não há restrições para cópias ou incorporações em produtos proprietários. Possui grande suporte feito on-line pela comunidade Python, que responde as dúvidas dos usuários e também é responsável pelo desenvolvimento da linguagem. Ela é composta pelo criador da linguagem Guido van Rossum e outras milhares de pessoas e grande parte de suas atividades são coordenadas via Internet. A especificação da linguagem é mantida pela Python Software Foundation (PSF).

Os programas escritos em Python executam em praticamente todas as principais plataformas, porque a sua implementação padrão é definida pela American National Standards Institute (ANSI) escrita em linguagem C portátil, portanto, os programas podem ser executados desde Personal Digital Assistant (PDAs) até em supercomputadores. Python está disponível em sistemas como: Unix, Linux, MS-DOS, MS Windows, Macintosh, entre outros. Além do interpretador, o conjunto de módulos de bibliotecas padrão que vêm com o Python, são implementados para serem portáveis entre as mais diversas plataformas, e os programas são compilados em código de byte portátil, permitindo que sejam executados em qualquer plataforma com uma versão compatível de Python instalada.

A linguagem é poderosa por unir características das linguagens de script, como simplicidade e facilidade de uso e as ferramentas avançadas de engenharia de software encontradas em linguagens compiladas. Nessas ferramentas, são encontrados alguns itens essenciais como:

- **Tipagem dinâmica:** a declaração de tipos dos objetos não é necessária. O tipo de uma variável é definido com a simples atribuição de um valor, e ele permanece o mesmo até que a variável seja recriada por uma nova atribuição. O valor associado pode inclusive mudar durante a execução do programa.
- **Gerenciamento de memória automático:** Python aloca e recupera de forma automática os objetos que não estão sendo mais usados através do “coletor de lixo” (garbage collector).
- **Suporte para programação em grande escala:** Python inclui módulos, classes e exceções para auxiliar na construção de sistemas maiores.

- **Tipos de objetos incorporados:** são oferecidas estruturas de dados como listas, dicionários e strings.
- **Ferramentas incorporadas:** há um conjunto de operações padrão para processar os diversos tipos de objetos, como a concatenação, fracionamento, ordenação, entre outras.
- **Bibliotecas:** existe o suporte a uma grande diversidade de bibliotecas. Pode ser feito em Python programas de vários tipos, utilizando gráficos, funções matemáticas complexas ou uma base de dados.
- **Utilitários de outros fornecedores:** por ser software livre, Python estimula os desenvolvedores com a criação de ferramentas que suportem tarefas além das que já são da própria linguagem.

Os códigos escritos em Python podem ser incorporados a outros códigos escritos em uma linguagem diferente. A Application Programming Interface (API) C do Python permite que programas em C referenciem e sejam referenciados por programas em Python. Além de outras integrações, Python possui uma com a plataforma Java através do projeto Jython, que oferece total acesso as classes Java e permite que o código Python seja compilado nessas classes. Essa flexibilidade possibilita o uso de programas em Python em outros ambientes ou sistemas.

Quando comparada a outras linguagens de programação, Python é considerada de fácil aprendizado. A sua execução é direta, bastando digitar o código e executá-lo, proporcionando interatividade e retorno rápido. Com sua sintaxe simples e, no entanto, poderosa, até mesmo programas sofisticados podem ser feitos em um curto tempo de desenvolvimento.

O Python possui um grande número de usuários e uma comunidade de desenvolvedores ativa. Por já estar no mercado há um tempo considerável e ser bastante utilizado, a linguagem se torna cada vez mais estável e robusta. Python se torna ideal para desenvolver scripts para a automação de determinadas tarefas; scripts para Internet; sistemas cliente-servidor; aplicações desktop; sistemas científicos até programação gráfica e de jogos. A sua versatilidade permite o desenvolvimento de aplicações para diferentes tipos de arquiteturas, podendo ser utilizado de *mainframes* a celulares. Além dos usuários individuais, Python é utilizado por importantes empresas como Google e Yahoo! em serviços de Internet, a IBM o usa para testes de hardware, a Industrial Light and Magic o usa para produção de animação em filmes, entre outras.

“Sua natureza de propósito geral o torna aplicável em quase todos os campos e não apenas em um.” (LUTZ, 2007, p. 37).

### 4.3 COMPILAÇÃO E INTERPRETAÇÃO

Quando o pacote Python é instalado no computador, são gerados vários componentes, mas essencialmente o interpretador e uma biblioteca de suporte. O interpretador pode assumir a forma de um programa executável ou de um conjunto de bibliotecas vinculadas a outros programas. Independente da forma assumida, todo o código escrito deve ser executado por ele.

“Depois de escritos, os programas são executados pelo interpretador Python, que faz a leitura e a execução das instruções do código, sendo o próprio interpretador um tipo de programa, uma camada lógica entre o código e a máquina.” (LUTZ, 2007, p. 45).

Um programa escrito em código Python pode ser criado meramente com o uso de algumas instruções da linguagem em um arquivo texto salvo com a extensão *.py*. Quando o Python é instruído a executar o código ou o *script*, algumas etapas são criadas até que ele seja realmente executado. Antes, ele é compilado para o código de byte e depois, passa para a chamada máquina virtual.

#### 4.3.1 O bytecode

O bytecode ou código de byte é simplesmente a tradução do código-fonte do programa em um conjunto de instruções em código de byte, uma representação de nível mais baixo que independe da plataforma. Dessa maneira a transformação garante portabilidade e velocidade aos programas, pois o código de byte é executado de forma mais rápida que o código-fonte original.

Depois da execução do programa, o Python pode armazenar em disco o respectivo código de byte em arquivos com a extensão *.pyc*, que significa arquivo *.py* compilado. A vantagem em armazenar esses códigos está na otimização da velocidade de inicialização. Quando o programa for executado novamente, caso o arquivo tenha sido salvo e inalterado, o Python carregará o arquivo *.pyc*, pulando a etapa de compilação. Caso tenham sido feitas alterações no código-fonte, o Python verifica a data desse arquivo original com a do arquivo de código de byte, para que possa recompilar.



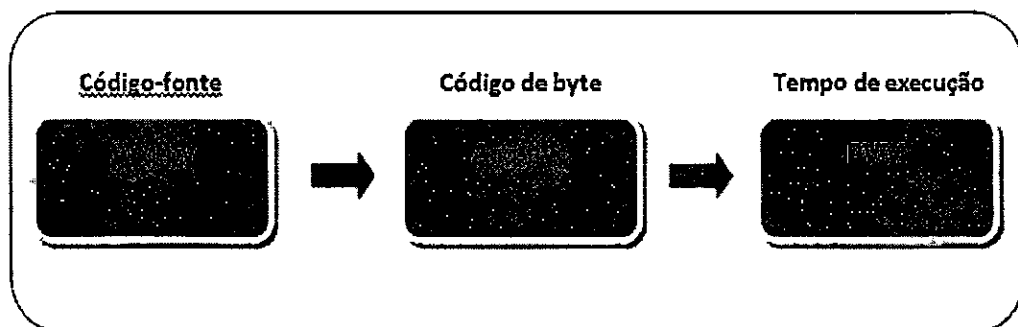
Em relação à portabilidade, os arquivos `.pyc` permitem que os programas Python sejam distribuídos entre diferentes plataformas, pois eles são executados mesmo com os arquivos de código-fonte `.py` ausentes.

#### 4.3.2 Python Virtual Machine (PVM)

Após a compilação do código-fonte em código de byte, o programa é executado pela Máquina Virtual Python. Ela é o mecanismo de tempo de execução do Python e é considerada a última etapa do interpretador.

“A PVM é definida como um grande loop que faz repetições nas instruções em código de byte, uma por uma, para executar suas operações.” (LUTZ, 2007, p. 48).

Todo o processo desde a transformação em código de byte até a execução pela PVM é oculto ao programador, cuja função é de codificar e executar os arquivos.



**Figura 6:** Transformação em código de byte  
*Fonte: Python para desenvolvedores. 2010.*

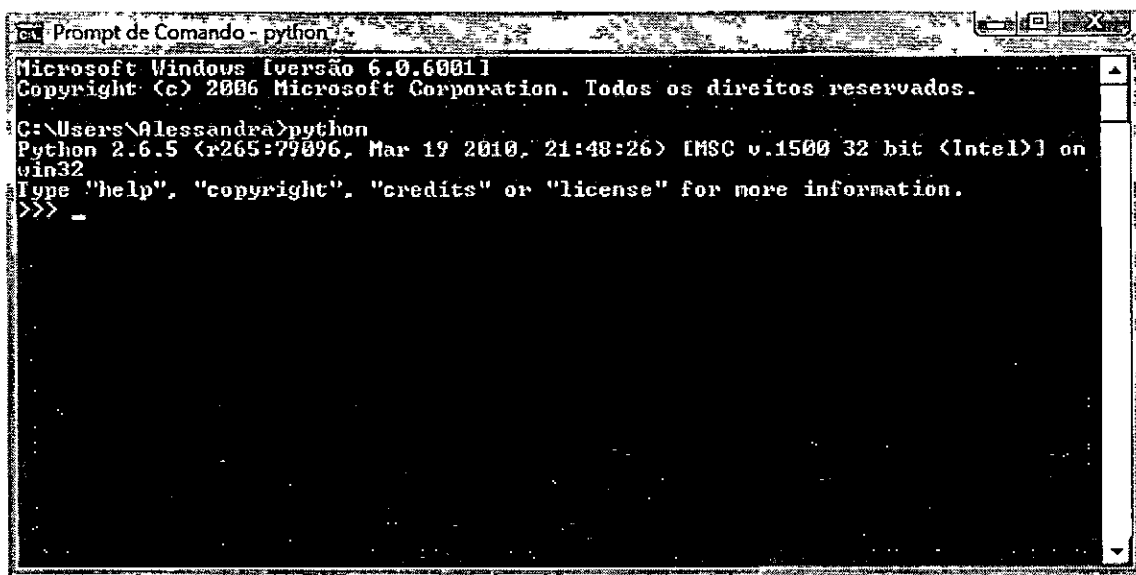
“O código de byte não é código de máquina binário, ele é uma representação específica da linguagem Python. Por isso, alguns programas Python não são executados com a mesma velocidade de outros escritos em linguagens compiladas.” (LUTZ, 2007, p. 49).

No entanto, essa diferença relativa à velocidade de execução dos programas não é um fator muito relevante para aqueles escritos em Python, pois o seu ganho em velocidade de desenvolvimento é considerado mais importante.

#### 4.3.3 Modos de execução

O interpretador Python pode ser usado de forma interativa, a partir do console de comandos do sistema, ou Shell para sistemas Unix. As linhas de código são escritas

diretamente, sem a necessidade de criar um arquivo.py separado. Para executar o interpretador, basta digitar o comando *python* e aparecerá na janela do console o símbolo `>>>`, chamado de *prompt*. Ele indica que o interpretador está esperando por alguma instrução.



```
Prompt de Comando - python
Microsoft Windows [versão 6.0.6001]
Copyright (c) 2006 Microsoft Corporation. Todos os direitos reservados.
C:\Users\Alessandra>python
Python 2.6.5 (r265:79096, Mar 19 2010, 21:48:26) [MSC v.1500 32 bit (Intel)] on
win32
Type "help", "copyright", "credits" or "license" for more information.
>>> _
```

Figura 7: O interpretador Python  
Fonte: Primária.

Esse modo interativo é uma característica da linguagem que a difere das demais, pois possibilita que trechos de códigos sejam testados e/ou modificados antes de ser feita sua inclusão nos programas originais.

Existem muitas ferramentas de desenvolvimento para Python, como Interface Development Environment (IDE), editores e shells:

Os IDEs são pacotes de software que integram várias ferramentas de desenvolvimento em um ambiente consistente, com a finalidade de aumentar a produtividade do programador. Eles costumam possuir recursos como *syntax highlight* (código-fonte colorizado de acordo com as regras da sintaxe da linguagem), *shell* integrado e *code completion* (o editor apresenta possíveis complementos para trechos do código que identifica).

Os editores são programas especializados em código de programação e possuem funções como coloração de sintaxe, exportação de código para outros formatos e conversão de codificação de texto.

Shell é um ambiente interativo para execução direta de comandos. O pacote de instalação do Python já inclui um Shell padrão.

## 5 O FRAMEWORK DJANGO

Django é um framework para desenvolvimento web, que fornece uma infra-estrutura de programação para as aplicações do usuário de forma que ele possa escrever um código limpo e sustentável, sem precisar reinventar recursos que o framework possui. Ele é construído de acordo com o padrão Model-View-Controller (MVC), que é um padrão de arquitetura de software, escrito na linguagem Python. Trata-se de um software livre construído para possibilitar o rápido desenvolvimento de web-sites dinâmicos, aplicações e serviços web.

Django ainda inclui um sistema de templates que permite separar a lógica de programação do design. Ele permite construções de sites dinâmicos em curto tempo, é projetado para que o programador possa se concentrar em aspectos mais interessantes do trabalho, aliviando as atividades repetitivas. Dessa forma, fornece abstrações de alto nível de padrões comuns de desenvolvimento web e uma visão mais clara de como resolver os problemas. Por isso a utilização desse framework se torna produtiva, por não ser necessário ao programador criar determinados recursos que já foram implementados. O seu uso também requer alguns conhecimentos básicos sobre Web, Programação Orientada a Objetos, Banco de Dados e Python.

### 5.1 HISTÓRICO

Django foi criado por um grupo de desenvolvedores Web em Lawrence, Kansas, USA. Ele foi criado em 2003, dentro do *Lawrence Journal-World*, por Adrian Holovaty e Simon Willison, quando eles começaram a usar código em Python para construir aplicações. A equipe era responsável pela produção e manutenção de vários sites locais de notícias, logo, Django prosperou em um ambiente de desenvolvimento ditado por prazos definidos por atividades jornalísticas. Esses sites requeriam que características fossem adicionadas e que aplicações inteiras fossem construídas em um intervalo de tempo muito curto, muitas vezes em dias ou somente em algumas horas. Dessa forma, Adrian e Simon desenvolveram um framework capaz de poupar-lhes tempo, pois essa era a única maneira de poderem construir aplicações que pudessem ser mantidas sob prazos extremos. O framework foi criado originalmente como sistema para gerenciar um site jornalístico, mas a equipe — que então incluía um novo membro — Jacob Kaplan-Moss, decidiu lançá-lo como um projeto de código

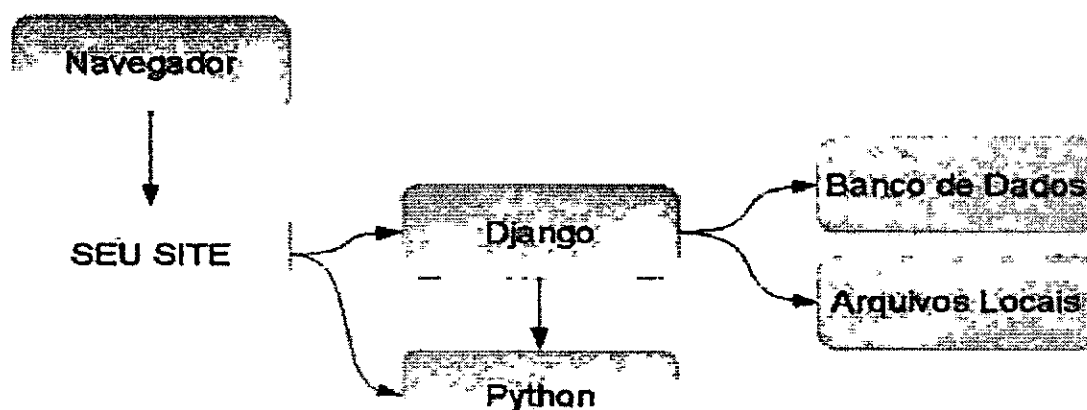
aberto, em julho de 2005. A idéia de atribuição do nome Django ao framework foi inspirada no músico de jazz Django Reinhardt.

Geralmente, o programador escreve linhas de códigos para uma determinada finalidade, e em seguida reescreve para outras, depois ele percebe que muito código pode ter sido repetido, causando desperdício de tempo e energia. Esse processo é uma das principais razões pelas quais o framework foi criado. Para isso, o Django usa uma metodologia de uso muito prática e objetiva, chamada Don't Repeat Yourself (DRY). A idéia básica é de que o programador faça algo apenas uma vez, de forma bem feita e sem ter que repetir. O código deve ser o mais reutilizável possível.

Anos depois de sua criação, Django está bem estabelecido como software de código aberto, com milhares de usuários espalhados pelo mundo. Dois dos desenvolvedores, Adrian e Jacob, ainda fornecem orientação para o crescimento do framework, contudo, a maior parte do esforço é advinda da equipe colaborativa.

## 5.2 VISÃO GERAL DO FUNCIONAMENTO

1. É necessário o uso de um navegador, como o Mozilla Firefox, que funciona nos sistemas Windows, Linux e Mac.
2. Inicia-se o navegador e digita-se o endereço do site criado.
3. O site é feito em Django, usando a linguagem de programação Python.
4. Através do Django, o site acessa ao banco de dados e em arquivos locais, retornando para o navegador uma página com várias funcionalidades.



**Figura 8:** Visão geral do funcionamento Django  
*Fonte: Aprendendo Django no Planeta Terra.2010.*

### 5.3 OS OBJETOS: HttpRequest e HttpResponse

O protocolo de hipertexto — HTTP — é a linguagem fundamental para que haja a comunicação através da Internet. Ele é falado por servidores e navegadores, juntamente com uma variedade de ferramentas especializadas para lidar com a Web.

Esse protocolo é essencial para o funcionamento de uma requisição (*request*) e uma resposta (*response*). O usuário emite uma requisição para o servidor, que retoma uma resposta com as informações solicitadas ou um erro indicando porque o pedido não pode ser cumprido. Enquanto as requisições e as respostas seguem uma detalhada especificação, Django fornece um par de objetos em Python que são projetados para tornar o protocolo mais simples de lidar em seu próprio código. A maioria dos detalhes é tratada “nos bastidores” pelo framework.

#### 5.3.1 HttpRequest

Como já foi mencionado, toda view recebe como seu primeiro argumento um objeto representando a requisição HTTP. Esse objeto é uma instância de uma classe, que encapsula uma série de detalhes a respeito da requisição, assim como alguns métodos úteis para a realização de algumas funções.

##### 5.3.1.1 Entendendo como Django lida com a requisição

Há várias informações em uma requisição (*HttpRequest*). Uma das informações é a URL, composta pelo protocolo, domínio, porta, caminho e parâmetros (nem sempre presentes).

**Http://localhost:8000/admin/?nome=Carl&idade=12**

- Protocolo: http
- Domínio: localhost
- Porta: 8000
- Caminho: /admin/
- Parâmetros: nome = Carl e idade = 12

Caso a porta não seja citada, deve-se entender que se trata da porta 80, padrão do protocolo Http.

### **1º passo: A URL é solicitada**

O usuário solicita uma página através da URL utilizando o navegador. O servidor web recebe a solicitação e a passa para o Python e o Django.

### **2º passo: O middleware é chamado**

Os middlewares são pequenos trechos de código que analisam a requisição na entrada e a resposta na saída. Um dos middlewares faz toda a segurança e autenticação do usuário. Outro adiciona a função de sessão, que é uma forma de memorizar o que o usuário está fazendo, a fim de ajudá-lo. Há outro middleware que memoriza as respostas no cache, pois caso a próxima requisição tenha o mesmo caminho e parâmetros, ele retorna a resposta que foi memorizada, evitando que seja processada novamente pela view.

### **3º passo: A URL é avaliada**

Depois de passar pelos middlewares, a requisição é analisada por um importante componente do Django, o URL Dispatcher. Ele faz a verificação do endereço e também do arquivo *urls.py* do projeto para apontar qual view será chamada para dar a resposta.

### **4º passo: O middleware é chamado novamente**

Se houverem funções dos middlewares para serem executadas depois da avaliação da URL e antes da chamada da view, então elas deverão ser chamadas neste momento.

### **5º passo: A view é chamada**

A view é uma função escrita em código Python. Ela recebe uma requisição (HttpRequest) e devolve uma resposta (HttpResponse). Nesse momento é realizado o trabalho do programador, que é analisar uma requisição, utilizar-se do banco de dados e então retornar uma página. O Django possui algumas views, prontas chamadas *generic views*. Há *generic views* úteis para blogs, cadastros em geral, para lembrar senhas, autenticar e sair do sistema, redirecionar, entre outras. As views escritas pelo programador devem estar num arquivo *views.py*.

### **6º passo: O template é carregado**

O *template object* é chamado e recebe o arquivo de template apropriado do sistema de arquivos, no local especificado pela view.

#### 7º passo: O template é renderizado

O texto dentro do template é renderizado. Os delimitadores são substituídos pelos respectivos dados. Nesse passo, o template é uma string Python.

#### 8º passo: O middleware é chamado novamente

Se houverem funções dos middlewares para serem executadas depois da geração da resposta, mas antes dela ser enviada ao usuário, então elas deverão ser chamadas neste momento.

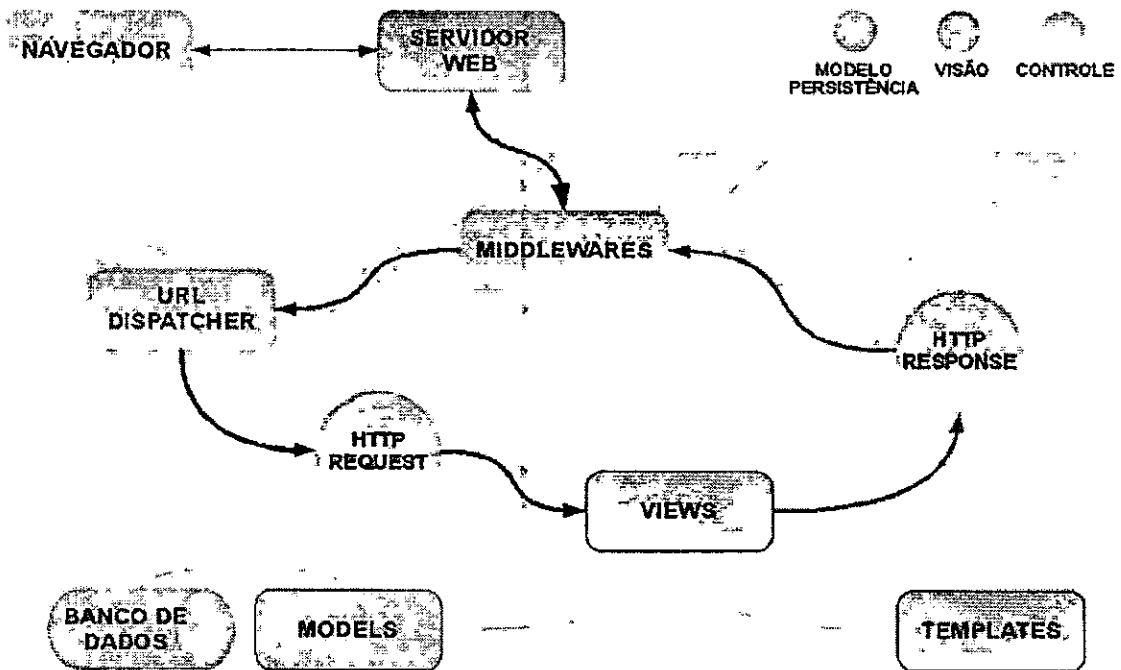
#### 9º passo: A resposta é enviada ao navegador

O template renderizado é empacotado com o formato necessário para que seja interpretado pelo navegador. A página de resposta enviada ao usuário pode estar no formato HTML ou outros como XML, JavaScript, CSV e PDF.

### 5.3.2 HttpResponse

Após uma requisição ser recebida e processada, cada view é responsável por retornar a instância de uma resposta. O objeto mapeia um real HttpResponse, incluindo cabeçalhos, essa é a única maneira de controlar o que é enviado de volta para o navegador. Vários atalhos, chamados de *shortcuts*, estão disponíveis para criar respostas mais facilmente. Ao contrário do que ocorre com as requisições, o programador ao criar suas views, tem o controle total sobre a forma como as respostas serão criadas. A classe padrão que é instanciada, aceita três argumentos para personalizar seu comportamento, contudo, nenhum deles é necessário. Digitando-se as linhas abaixo no prompt de comando, tem-se a seguinte saída:

```
>>>From django.http import HttpResponse
>>>Print HttpResponse()
Content-Type: text/html; charset=utf-8
```



**Figura 9:** HTTP request e HTTP response  
 Fonte: *Aprendendo Django no Planeta Terra.2010.*

#### 5.4 O PADRÃO MVC

O Django faz com que o desenvolvedor se preocupe menos com coisas redundantes, porque proporciona o desenvolvimento rápido de aplicações. Ele automatiza coisas repetitivas, como formulários, enquetes e qualquer outro tipo de aplicação que tenha que ser feita, podendo ser aproveitada em outros projetos, pois Django tem em sua filosofia que toda aplicação pode e deve ser plugável.

No trabalho com esse framework, é utilizada a construção em camadas, o que facilita o desenvolvimento e a manutenção dos sistemas. O Django utiliza o padrão MVC, onde o código que define e acessa os dados (models) fica separado da lógica (controller), que por sua vez está separado da interface do usuário (views), mas os seus idealizadores determinaram os nomes das camadas utilizadas como MTV: Models, Templates e Views. A principal vantagem dessa abordagem é que esses componentes são de baixo acoplamento, pois cada peça da aplicação tem uma finalidade essencial e pode ser alterada de forma independente, sem afetar as demais. Por exemplo, o desenvolvedor pode alterar uma URL de uma parte da aplicação sem afetar a implementação, assim como um designer pode alterar o código HTML de uma página sem precisar mexer no código Python que a renderiza. Da mesma maneira um administrador de banco de dados pode



renomear uma tabela do banco e especificar a alteração em um único local, ao invés de procurar inúmeros arquivos e fazer as substituições.

#### 5.4.1 Models

São os modelos ou layout dos dados. Com eles é feito o Mapeamento Objeto Relacional do banco de dados. Todas as classes criadas são transformadas em tabelas no banco, e durante o desenvolvimento não é preciso digitar praticamente nenhuma string Structure Query Language (SQL). Um model em Django é a descrição dos dados do banco de dados, representados em código Python. Django usa o model com a função de executar SQL de forma interna, para retornar estruturas de dados em Python representando as linhas das tabelas do banco. Os models existem de forma independente do resto do sistema e podem ser usados por qualquer aplicação que tenha acesso a eles. Esses dados podem até mesmo serem manipulados via interpretador sem precisar carregar um servidor web. Tendo os models armazenados como código e não no banco de dados, faz com que seja fácil mantê-los sob controle. Dessa forma, é mais simples acompanhar as mudanças de layout dos dados.

A desvantagem dessa abordagem, no entanto, é que o código em Python pode estar fora de sincronia com o que está realmente no banco de dados. Se forem feitas alterações em um model do Django, é necessário que o programador faça as mesmas alterações na base de dados para mantê-lo consistente com o model. Django inclui um utilitário que pode gerar modelos a partir de um banco de dados já existente.

Cada Model é uma classe escrita em código Python que herda ou estende do módulo `django.db.models.Model`. Um exemplo de código de um model:

```
class Book (models.Model):  
    name = models.CharField(max_length = 40)  
    pub_date = models.DateField()
```

#### 5.4.2 URLs e Views

##### 5.4.2.1 URLs

Django não oferece todos os recursos automaticamente para que sejam geradas as URLs de qualquer site. Para cada site, deve ser declarado explicitamente o esquema de

endereçamento mais apropriado, utilizando as configurações de URL. Esta pode ser vista como uma limitação do framework, no entanto essa é uma característica, pois permite ao programador definir os endereços do seu site da forma como gostaria. Todas as configurações relativas às URLs ficam armazenadas num arquivo *urls.py*. E essas configurações são compostas de uma lista de padrões que mapeia para cada URL uma view específica. Esses padrões possuem alguns componentes, e todos são especificados em conjunto, como argumentos para uma função.

```

From django.conf.urls.defaults import *

urlpatterns = patterns(' ',
    (r'^$', 'post_list'),
    (r'^(?P<id>\d+)/$', 'post_detail'),
    (r'^(?P<id>\d+)/comment/$', 'post_comment'),
)

```

Esses argumentos podem ser colocados em dois grupos:

- Um prefixo único de caminho de importação para cada view, que é especificada como uma string.
- Um número qualquer de padrões de URL.

A tupla de padrões definida nas configurações de URL, contém todas as informações necessárias para mapear uma solicitação de entrada para uma view. Os caminhos definidos pelas URLs serão checados com uma das expressões regulares, que foram previamente definidas. Se alguma for encontrada, a solicitação é passada para a view específica. Dessa forma, no exemplo, a linha do código (**r'^\$', 'post\_list'**), utiliza uma expressão regular que serve para fazer correspondência a uma determinada URL, se for encontrada uma combinação correta, a view em questão é carregada.

Símbolo	Corresponde a:
. (ponto)	Qualquer caractere.
\d	Qualquer dígito.
[A-Z]	Qualquer caractere entre as letras A e Z maiúsculas.

[a-z]	Qualquer caractere entre as letras A e Z minúsculas.
[A-Za-z]	Qualquer caractere entre as letras a e z.
+	“Um ou mais” da expressão anterior. Por exemplo, <code>\d+</code> significa um ou mais dígitos.
[^/]+	Um ou mais caracteres antes da barra.
?	“Nenhum ou um” da expressão anterior. Por exemplo, <code>\d?</code> significa nenhum ou um dígito.
*	“Nenhum ou mais” da expressão anterior. Por exemplo, <code>\d*</code> significa nenhum, um ou mais dígitos.
{1,3}	“Entre um e três” da expressão anterior. Por exemplo, <code>\d{1,3}</code> significa um, dois ou três dígitos.

**Tabela 3:** Símbolos mais comuns das expressões regulares

#### 5.4.2.2 Views

Elas funcionam como o "controller" da aplicação. Nessa parte é definida o que será passado para os templates apresentarem em suas páginas. Basicamente, views são definidas como funções escritas em código Python, que são chamadas quando o usuário aciona uma URL específica. Elas recebem de entrada um objeto *HTTP request* e retornam um objeto *HTTP response*. Independente de outros argumentos que possam ser passados, uma view sempre recebe como primeiro argumento um objeto, ou seja, uma view deve aceitar no mínimo esse objeto de argumento.

Uma view deve ser capaz de aceitar quaisquer argumentos que são passados, incluindo aqueles que são capturados a partir da URL, e se comportar de acordo com a lógica de interação das aplicações, retomando ao usuário uma tela adequada, para que seja possível acessar os dados representados pelos modelos. Em outras palavras, a view precisa saber como lidar com os argumentos passados, tratando-os dentro do seu código para que então possa dizer ao sistema de templates, qual deles deverá ser carregado. Elas podem acessar os dados definidos pelos models, recuperar e atualizar informações necessárias para realizar uma dada tarefa solicitada pelo usuário.

Um tema comum no desenvolvimento em Django é sobre reutilização de código, para que sejam evitadas repetições desnecessárias. Em relação às views, parece pouco provável que possam se adequar a esse conceito. Entretanto, as *generic views* que o Django possui, servem a esse propósito e podem ser utilizadas em diversos tipos de aplicações, e para isso é

preciso somente de algumas poucas configurações. Um simples exemplo de código em uma view:

```
from django.http import HttpResponse
def hello(request):
    return HttpResponse("Hello world!")
```

### 5.4.3 Templates

O template é um arquivo de texto que se destina a separar a apresentação de um documento dos seus dados, nele são definidos espaços reservados chamados *placeholders* e as *templates tags* que estruturam como o documento deverá ser mostrado. (HOLOVATY, 2009)

No Django, templates são arquivos auxiliares utilizados pelas views. São basicamente arquivos de texto com delimitadores (*placeholders*) e uma lógica simples dentro deles. Esses delimitadores são avaliados quando os templates são renderizados e o Django os substitui por determinados valores. Enquanto as views são responsáveis por apresentar os dados, os templates têm a função de definir a forma como esses dados serão mostrados na tela para o usuário, controlando todos os detalhes referentes à apresentação da página. Normalmente são produzidos documentos em HTML, mas os templates podem ter qualquer outro tipo de formato baseado em texto.

Os templates formam uma parte importante do desenvolvimento em Django, tanto que se trata de uma camada inteiramente separada das demais. Algumas plataformas de desenvolvimento não possuem essa separação entre os códigos da lógica e da apresentação, e isso dificulta a sua reusabilidade e manutenção, já que o desenvolvedor geralmente precisa fazer atualizações. Por outro lado, a plataforma Django garante que os designers não precisam entender sobre código Python para trabalhar com os templates, pois o código relativo à parte lógica fica separado daquele relativo à interface. Entretanto, esse mecanismo dos templates é só uma das ferramentas que as views podem utilizar para gerar alguma interface como saída para o usuário. Trabalhar com esse sistema de templates é uma maneira claramente melhor de desenvolver aplicações.

Quando os templates são carregados e renderizados, os marcadores presentes no seu código são substituídos pelos valores adequados, logo, os traços da linguagem de marcação do template são removidos e o navegador apresenta uma página resultante. As views são o

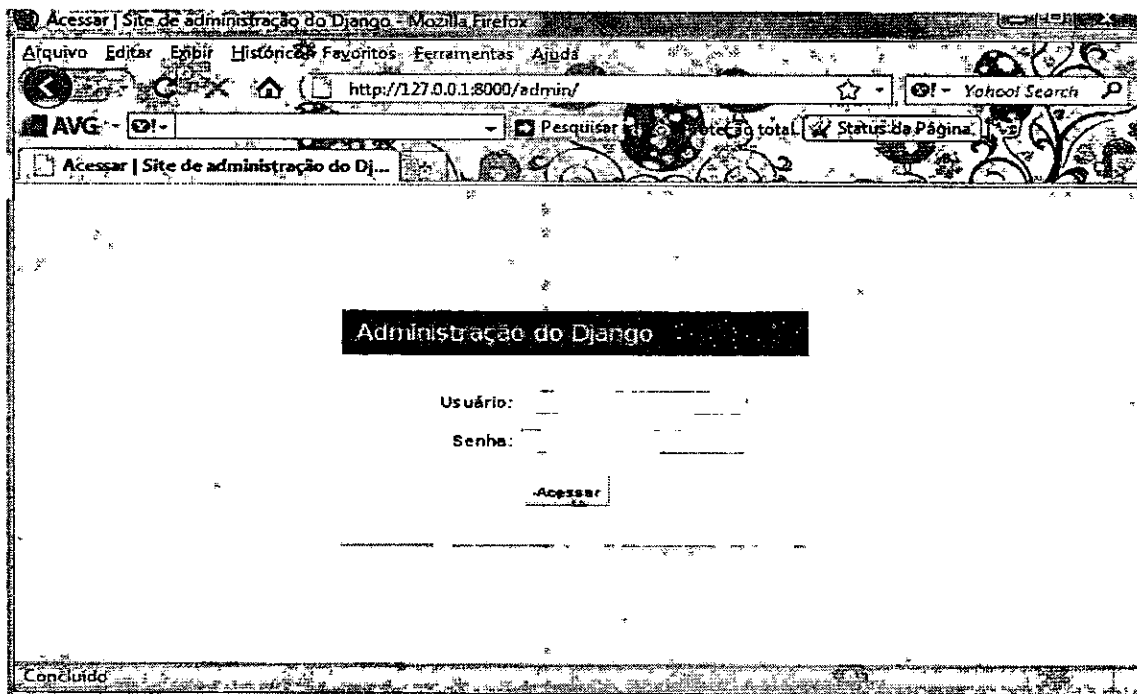
- O texto delimitado por um par de chaves e de sinais de percentagem, `{% if ordered_warranty %}` é chamado de *template tags*. Uma tag mostra ao sistema como realizar alguma função. A tag `{% if ordered_warranty %}` funciona através de uma condição, similar à função lógica que possui no código Python. Da mesma forma, a tag `{% for item in item_list %}` funciona permitindo que seja feito um *loop* para cada item na sequência.
- O texto `{{ship_date|date: "F j, Y"}}` é um exemplo de filtro, que é o modo mais conveniente de alterar o formato de uma variável. Nesse caso, uma variável `ship_date` é passada para o filtro `date`, para o qual são passados os argumentos: “F j, Y”. O filtro altera o formato da data de acordo com o especificado pelos argumentos. Os filtros são utilizados através da colocação de barra `|` após a variável.

## 5.5 O SITE DE ADMINISTRAÇÃO DO DJANGO

As interfaces de administração são comuns, repetitivas e cansativas de serem construídas, possuem sempre as mesmas funcionalidades de autenticação de usuário, exibir e manipular formulários e validar dados de entrada. Por isso, Django já oferece como um recurso uma interface de administração própria, contribuindo com o trabalho do desenvolvedor, facilitando e melhorando sua produtividade.

Para ativar o site da administração do Django, é necessário fazer poucas modificações nos arquivos de `urls.py` e `settings.py`. Essa administração já faz parte de uma larga suíte de funcionalidades do Django, chamada `django.contrib`. Esses recursos já estão embutidos no próprio código do framework e auxilia para que o usuário não precise reinventar o que já está pronto.

O site da administração é projetado para usuários não técnicos, logo, é bastante auto-explicativo.



**Figura 10:** Administração do Django – login

Fonte: <<http://127.0.0.1:8000/admin/>>.

Para logar, é necessário que o usuário tenha criado anteriormente uma conta de superusuário. Caso não tenha sido feito, basta digitar no console do sistema o seguinte código: *python manage.py create superuser*.

Após logar, a página exibida é a página principal (*home page*) da administração. Essa página lista todos os objetos disponíveis que podem ser editados pelo site. Outros tipos de objetos podem ser adicionados pelo usuário através de sua criação nos *models*, no entanto, inicialmente o site disponibiliza somente Grupos e Usuários. Cada um possui uma lista de mudanças (*change list*) e um formulário de edição (*edit form*). As listas de mudanças exibem todos os objetos disponíveis no banco de dados e os formulários permitem ao usuário adicionar, alterar ou apagar registros do banco.

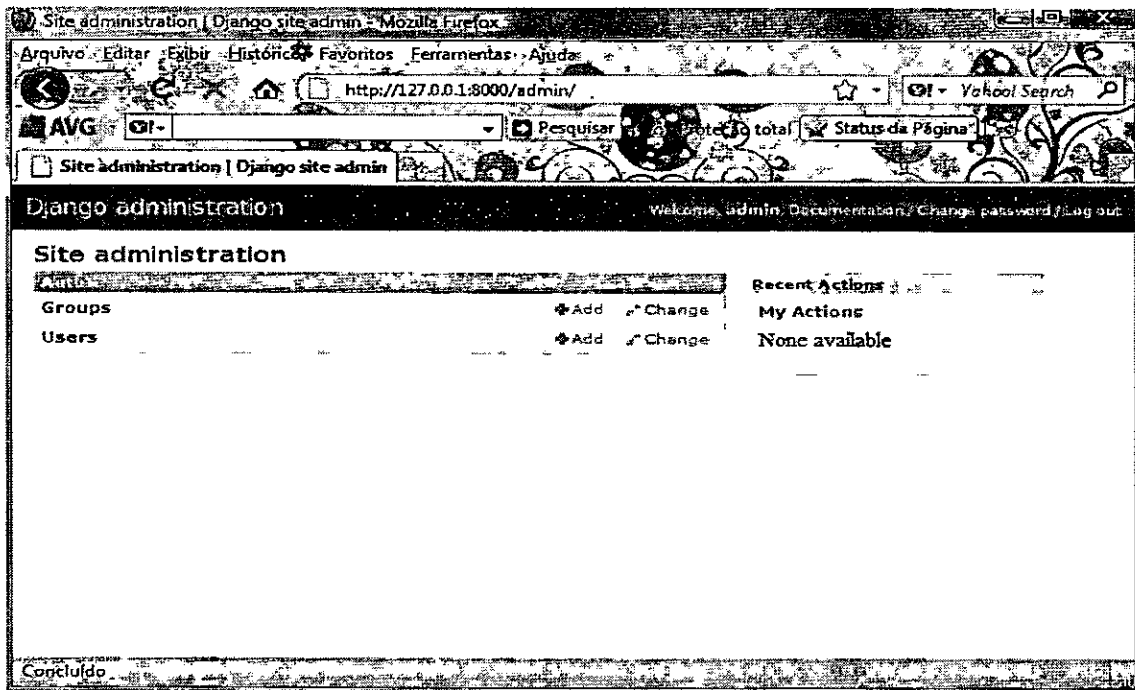


Figura 11: Administração do Django – página principal

Fonte: <<http://127.0.0.1:8000/admin/>>.

Ao clicar sobre a linha onde está o termo *Users*, uma página é carregada com uma lista dos usuários existentes e que estão armazenados na base de dados.

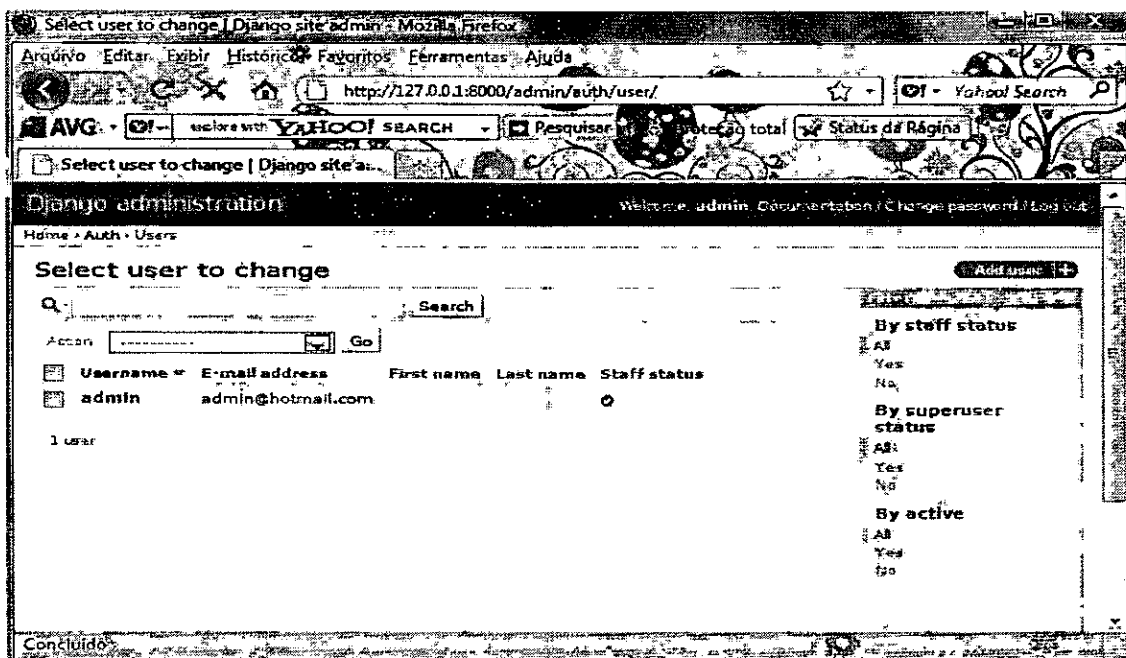
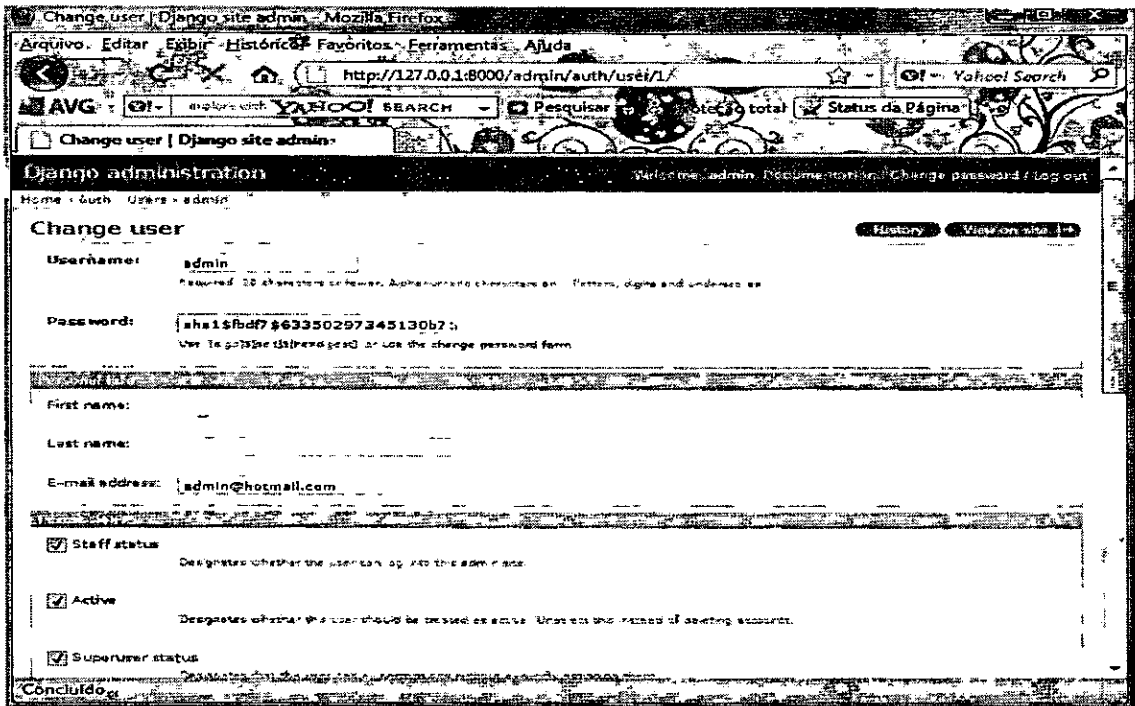


Figura 12: Administração do Django – lista de mudanças

Fonte: <<http://127.0.0.1:8000/admin/>>.

Clicando sobre um usuário, é exibido um formulário de edição que permite ao administrador, editar os atributos referentes ao usuário em questão. É possível tanto alterar

como adicionar ou apagar usuários e essas ações ficam registradas num histórico de links, que fica localizado no canto superior direito da janela. Toda mudança feita através da interface de administração fica registrada, para possíveis consultas futuras.



**Figura 13:** Administração do Django – formulário de edição

Fonte: `<http://127.0.0.1:8000/admin/>`.



## 6 IMPLEMENTAÇÃO

A pesquisa está voltada para a área de desenvolvimento web, destacando a utilização de uma ferramenta criada recentemente e específica para esse tipo de desenvolvimento, o framework Django. A grande maioria das publicações relacionadas à ferramenta está em livros escritos na língua inglesa e em grupos de estudo espalhados pela internet, denominados de comunidades web, dentre elas, a Django Brasil, que é um website da comunidade brasileira voltada para o uso desse framework. A escolha desse framework é bastante viável por todas as características citadas anteriormente no capítulo sobre Django. Como Django é um framework web Python, faz-se necessário a instalação de uma versão Python que pode ser a partir da 2.3 até a 2.6. Estando instalada alguma versão Python a partir da 2.5 ou posterior, é opcional configurar um servidor de banco de dados, pois o framework já inclui uma base de dados leve denominada SQLite. Django foi criado como um projeto de código aberto, publicado sob a licença de código aberto Berkeley Software Distribution (BSD) em 2005.

Toda a pesquisa foi feita em um período de seis meses, utilizando exclusivamente como fonte, sites e livros sobre assunto encontrados na internet. O objetivo geral desse trabalho é elaborar uma pesquisa teórica e prática a cerca da criação de um site dinâmico, utilizando para tal, o framework para desenvolvimento web, Django.

O projeto consistiu em duas etapas, a primeira foi realizada por meio de pesquisas bibliográficas feitas em sites da internet e através de análises em livros de Programação Web, Django, Python e HTML. A segunda etapa foi concluída a partir da utilização do framework Django, versão 1.1.1.

### 6.1 REQUISITOS

O levantamento dos requisitos foi feito a partir da obtenção de informações por meio de entrevistas informais e da observação direta.

#### Requisitos Funcionais:

- O sistema deve possibilitar o cadastro de alunos e professores, dos operadores e dos recursos;
- O sistema deve possibilitar o registro de reservas obedecendo a restrições;
- O sistema não deve permitir registros de reserva idênticos;

- O sistema deve possibilitar a visualização e controle dos registros dos recursos e das reservas;
- Os usuários devem poder obter informações sobre os recursos disponíveis e sobre os registros das reservas;

#### Requisitos Não-Funcionais:

- Somente um grupo de funcionários que trabalhe na instituição terá acesso ao sistema por meio de login e senha;
- Somente alunos e professores da IES poderão fazer reserva dos recursos;
- O software pode ser operacionalizado em qualquer sistema, independente da arquitetura;

## 6.2 METODOLOGIA

### 6.2.1 Criando um projeto

Para criar um projeto em Django, primeiramente é criada uma pasta que vai armazenar todos seus projetos criados. Iniciando o console do sistema, conforme já referenciado no capítulo sobre Python, é necessário utilizar-se de alguns comandos do Disk Operating System (DOS), um sistema operacional antigo cuja utilização é baseada em linhas de comandos digitados pelo usuário, para achar o caminho correto do diretório onde a pasta do projeto se encontra. Nesse caso, a pasta criada se chama “Projetos” e se encontra no caminho: C:\Projetos. A tabela abaixo apresenta alguns dos comandos básicos do MS-DOS (versão do DOS da Microsoft):

Comando	Utilização
DATE	Comando que atualiza a data do sistema operacional.
TIME	Semelhante ao comando <i>date</i> , só que <i>time</i> modifica a hora do sistema operacional em vez da data.
DIR	Comando que mostra a lista de arquivos de um diretório. Essa instrução pode conter alguns parâmetros, entre eles: /P - lista o diretório com pausas para quando a quantidade de arquivos é grande o suficiente para que não possa ser exibida de uma só vez na tela; /W - lista o diretório organizando a visualização na horizontal;

	/S - exibe não só o conteúdo do diretório atual como também o conteúdo das pastas deste; /? - use essa instrução para conhecer todos o parâmetros do comando <i>dir</i> .
<b>CLS</b>	Comando que "limpa" a tela, isto é, elimina as informações exibidas até então e deixa o cursor no canto superior esquerdo.
<b>MKDIR ou MD</b>	Comando que cria um diretório a partir da pasta corrente com o nome especificado.
<b>CHDIR ou CD</b>	Comando que muda o diretório corrente para outro a partir da pasta atual.
<b>RMDIR ou RD</b>	Comando que remove um diretório a partir da unidade corrente. O diretório somente será eliminado se não houver nenhum arquivo ou pasta em seu interior.
<b>CHKDSK</b>	Comando que checa a integridade e as especificações do disco mostrando informações sobre este na tela.
<b>MEM</b>	Exibe informações atuais sobre a memória do computador.
<b>RENAME ou REN</b>	Comando que permite ao usuário alterar o nome de um arquivo.
<b>MOVE</b>	Comando que tem duas funções: renomear diretórios ou mover arquivos de uma pasta para outra.

**Tabela 4:** Comandos do DOS

Após localizar a pasta, digita-se o comando abaixo (onde o termo *sgrr* é o nome do projeto):

```
python c:\Python26\Scripts\django-admin.py startproject sgrr
```

```

Prompt de Comando
Microsoft Windows [versão 6.0.6001]
Copyright (c) 2006 Microsoft Corporation. Todos os direitos reservados.
C:\Users\Alessandra>cd /
C:\>cd Projetos
C:\Projetos>python c:\Python26\Scripts\django-admin.py startproject sgrr
C:\Projetos>
  
```

**Figura 14:** Prompt de comando

Fonte: Primária.

Na pasta do projeto serão criados automaticamente quatro arquivos que são:

- **\_\_init\_\_.py**: Um arquivo vazio que diz ao Python que esse diretório deve ser considerado como um pacote Python. (pacote é um conjunto de módulos em Python).
- **manage.py**: Um utilitário de linha de comando que permite a você interagir com esse projeto Django de várias maneiras. É criado no diretório por pura conveniência.
- **settings.py**: Configurações para este projeto Django. *Django settings* irá revelar para você tudo sobre como o settings funciona.
- **urls.py**: As declarações de URLs para este projeto Django; um "índice" de seu site movido a Django. Você pode ler mais sobre URLs em *URL dispatcher*.

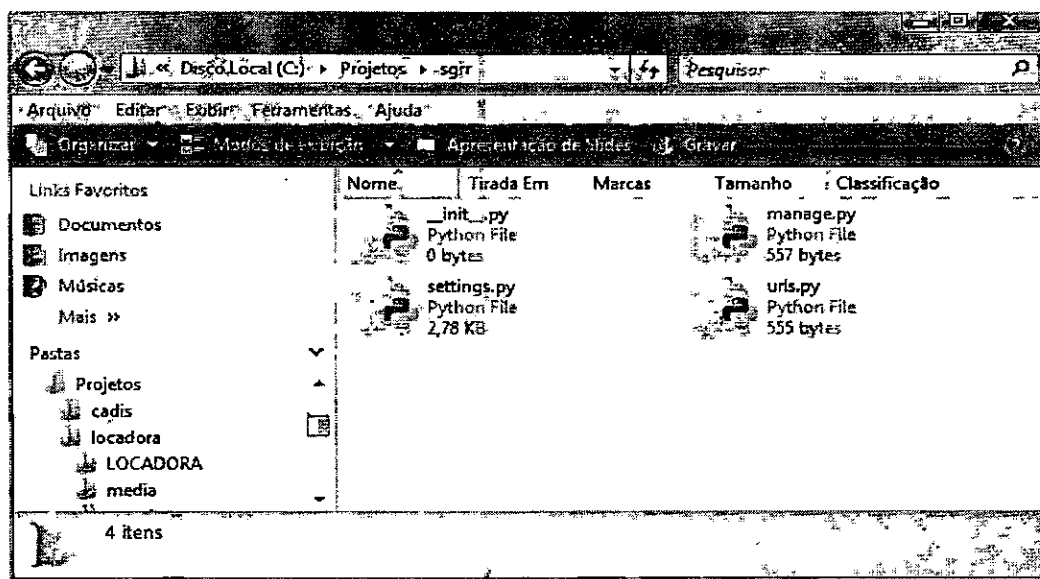


Figura 15: Diretório do projeto  
Fonte: Primária.

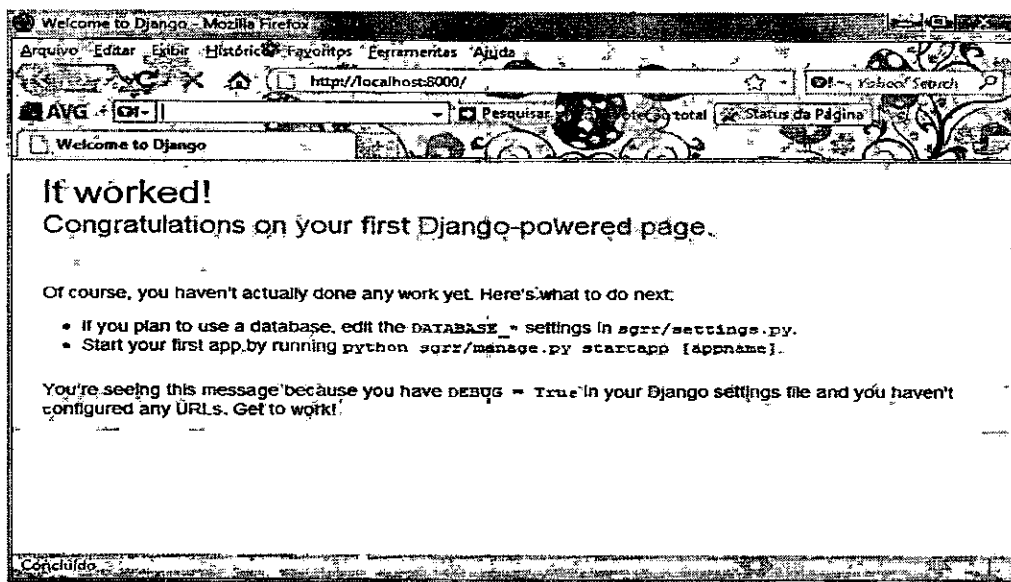
O Django vem com um pequeno servidor web embutido para auxiliar no desenvolvimento. O código a seguir tem a função de iniciar o servidor. Ainda dentro da pasta, será criado um novo arquivo utilizando o bloco de notas, nomeado como "executar.bat" e onde será escrito o código abaixo:

```
| python manage.py runserver
| pause
```

Após ser salvo, clica-se neste arquivo para executá-lo. A partir deste momento utiliza-se um navegador web. Esse arquivo pode ser utilizado sempre que necessário para iniciar o

servidor, no entanto sua criação é opcional, pois o mesmo código pode ser escrito e executado no console. Depois de o browser ter sido aberto e o endereço citado abaixo ser localizado, é exibida uma página de boas vindas do django.

**http://localhost:8000/**



**Figura 16:** Tela de boas-vindas do Django  
*Fonte: Primária.*

O conteúdo da página indica que a criação do projeto foi bem sucedida e explica que ainda não há nenhum trabalho realmente feito, para isso ainda é preciso configurar uma base de dados e iniciar uma aplicação.

### 6.2.2 Preparando uma base de dados

O banco de dados PostgreSQL foi escolhido para dar suporte ao projeto, pois esse software tem a vantagem de ser de código aberto, não é necessário pagar licença por ele, possui boa performance, é escalável e multi-plataforma.

Depois de o banco ter sido devidamente instalado e após ser iniciado, é preciso primeiramente criar uma nova role, preferencialmente com o mesmo nome do projeto e em seguida criar uma nova base de dados, onde o campo “owner” ou dono deve ser a role que foi criada e o campo referente à senha também tem o mesmo nome do projeto.

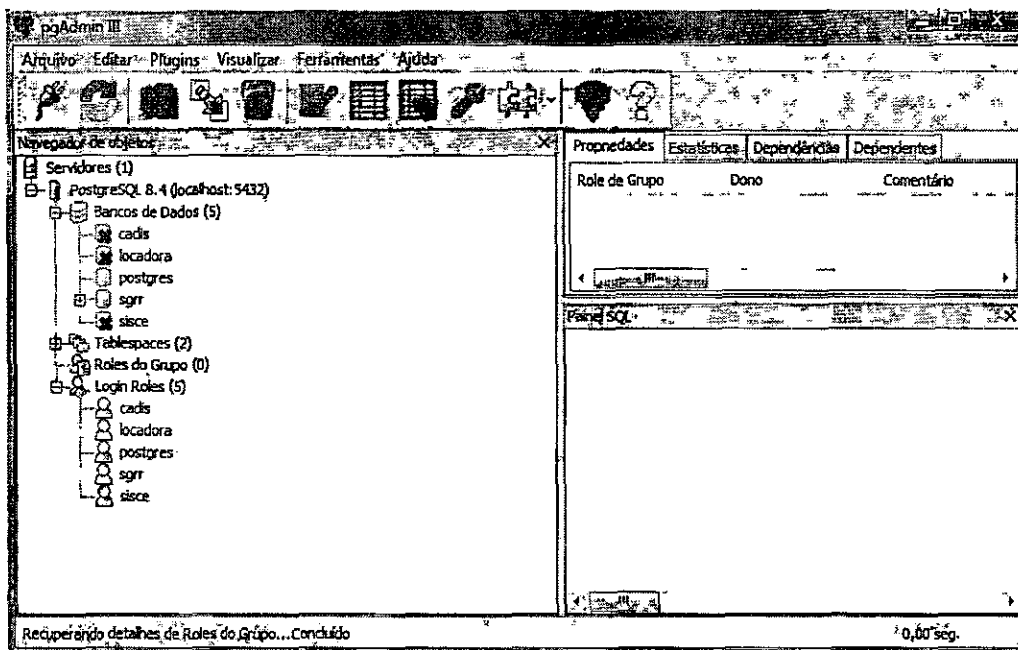


Figura 17: Interface do banco de dados

Fonte: Primária.

Depois de criar a base de dados, o próximo passo é editar o arquivo settings.py que se encontra na pasta do projeto. Inicialmente, os campos relativos à configuração do banco de dados se encontram vazios. É necessário que sejam configurados com as informações sobre o banco de dados escolhido.

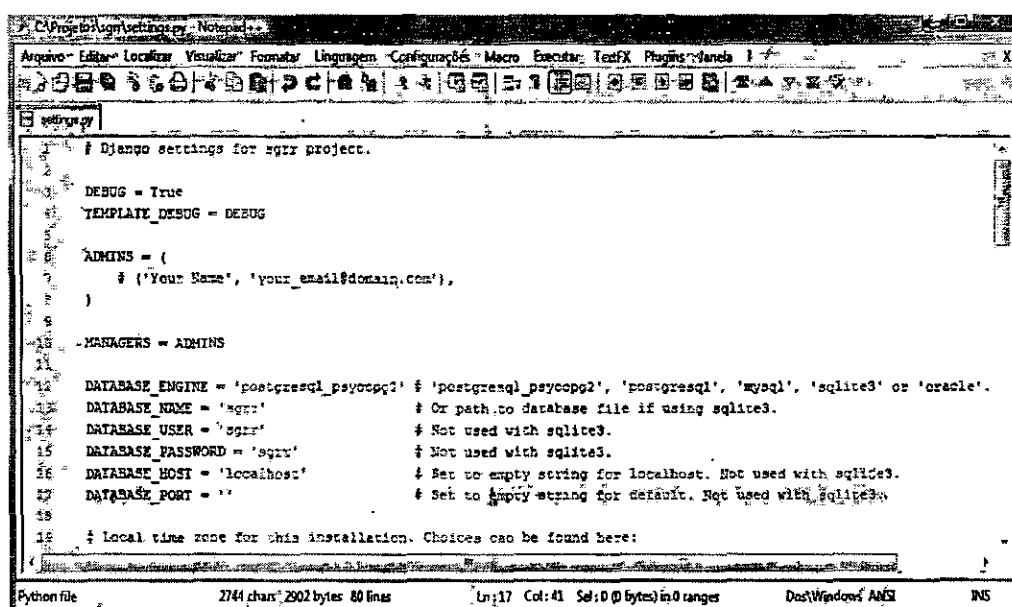


Figura 18: Visualização do arquivo settings.py no Notepad ++

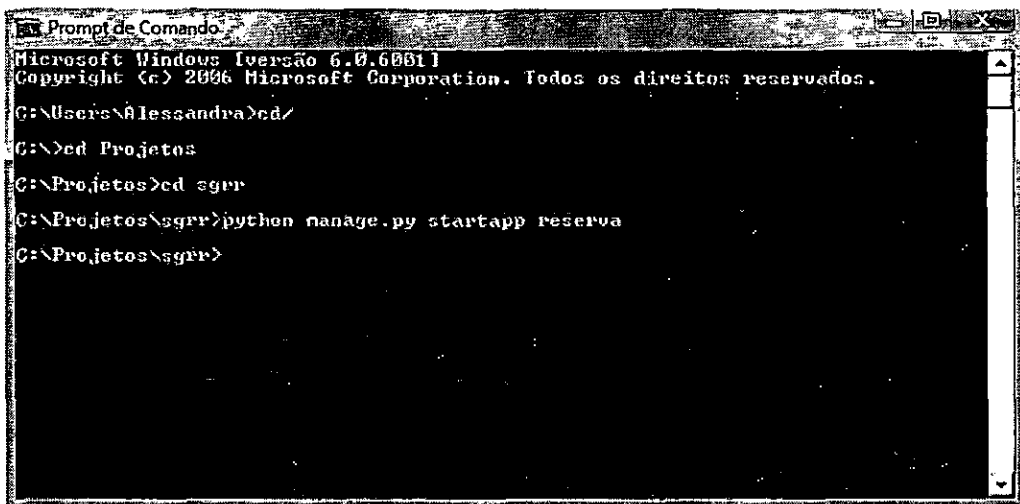
Fonte: Primária.

É importante criar na pasta do projeto outras duas pastas. Uma para armazenar os templates da aplicação que serão criados e outra para armazenar as imagens. Essas pastas podem ser nomeadas preferencialmente como templates e media respectivamente.

### 6.2.3 Criando a aplicação

Para criar a aplicação é preciso usar o console. Após localizar o diretório do projeto, o comando abaixo deve ser digitado (onde o termo *reserva* é o nome da aplicação):

```
python manage.py startapp reserva
```



```
Prompt de Comando
Microsoft Windows [versão 6.0.6001]
Copyright (c) 2006 Microsoft Corporation. Todos os direitos reservados.
C:\Users\Alessandra>cd /
C:\>cd Projetos
C:\Projetos>cd sgr
C:\Projetos\sgr>python manage.py startapp reserva
C:\Projetos\sgr>
```

Figura 19: Prompt de comando

Fonte: Primária.

A pasta da aplicação fica no diretório do projeto principal, com os demais arquivos criados anteriormente. Os arquivos que foram criados são: `__init__.py`, `models.py`, `tests.py` e `views.py`. A partir desse momento, é possível começar a desenvolver a aplicação, mas antes, é importante criar juntamente com a pasta da aplicação outras duas: uma para armazenar os arquivos de imagens (opcional) e outra para armazenar os templates.

### 6.2.4 Iniciando a aplicação

Dentro do diretório da aplicação *reserva*, o primeiro passo é editar o arquivo `models.py`. Para isso, basta abri-lo com o bloco de notas do sistema ou instalar outro software que tenha recursos de `highlighting`, como o Notepad ++, que é usado nessa implementação.

Nesse arquivo, são definidas todas as classes relativas às tabelas do banco de dados, é dessa forma que é feita a ligação da aplicação com a base de dados.

### 6.2.5 Utilizando a interface de administração do Django

Algumas dessas classes serão registradas diretamente na interface de administração do Django, para isso é necessário criar um arquivo `admin.py` dentro da pasta da aplicação. As classes que serão registradas serão: `curso`, `recurso`, `tipo_recurso` e `pessoa`.

Após criar e salvar os códigos nos arquivos `models.py` e `admin.py`, outro arquivo deverá ser criado, ainda na mesma pasta da aplicação: `forms.py`. Nesse arquivo ficam os formulários da aplicação. Nesse projeto foi necessário criar o formulário de reserva.

Após trabalhar nesses três arquivos, é preciso alterar o arquivo referente às configurações em `settings.py`, para que seja possível usar a administração do Django.

Até então, somente foi preparada a base para que a aplicação seja desenvolvida. O próximo passo é editar o arquivo `views.py`, `urls.py` e criar os templates que exibirão as páginas criadas no navegador. As views recebem os códigos escritos em linguagem Python, o arquivo das views é definido pelas funções, que concentram a maior parte do trabalho. No arquivo `urls.py` são definidas uma url para cada view. É necessário criá-las porque é a partir de uma url que uma view é chamada. Já os templates têm como função mostrar os resultados do processamento dos códigos das views. Cada view tem uma função específica e um template associado.

## 6.3 O SISTEMA DE GERENCIAMENTO DE RESERVA DE RECURSOS

Na tela inicial do sistema, existem três links que direcionam o usuário para outras páginas. O link “docente/discente” (voltado para alunos e professores da IES) e a imagem abaixo levam para uma página de login, enquanto o link “sobre” exibe somente uma página com informações básicas sobre o sistema.



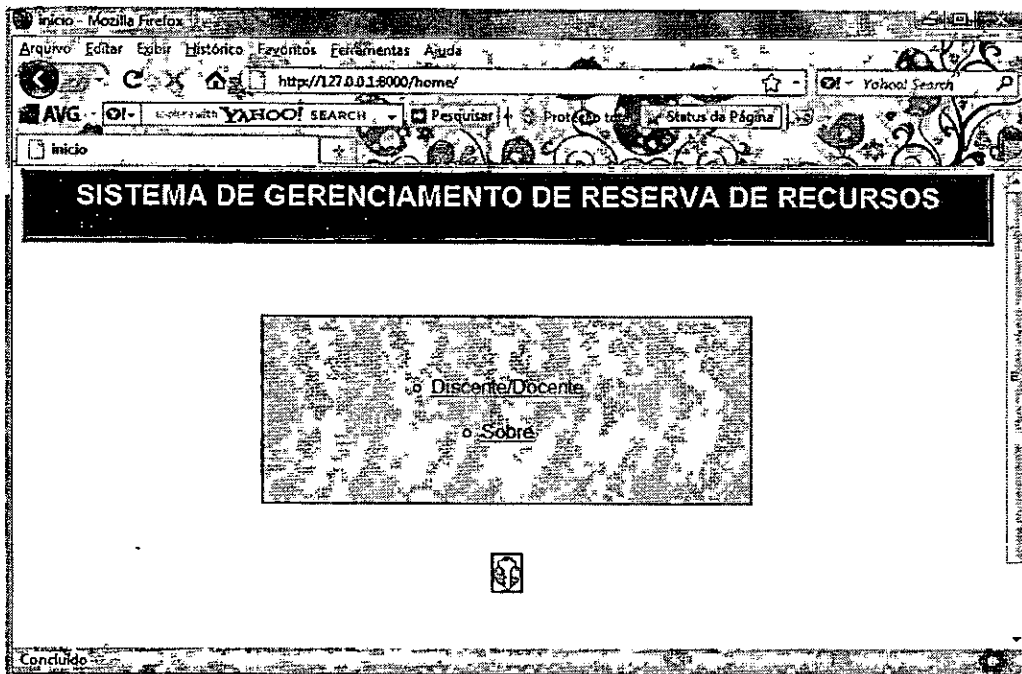


Figura 20: Tela inicial da aplicação

Fonte: Primária.

Assim como foi documentado no diagrama de caso de uso (vide anexos), o usuário deve fornecer dados de login válidos para ter acesso às opções dadas pelo sistema. Ao clicar sobre o link “docente/discente” aparecerá a seguinte tela de login:

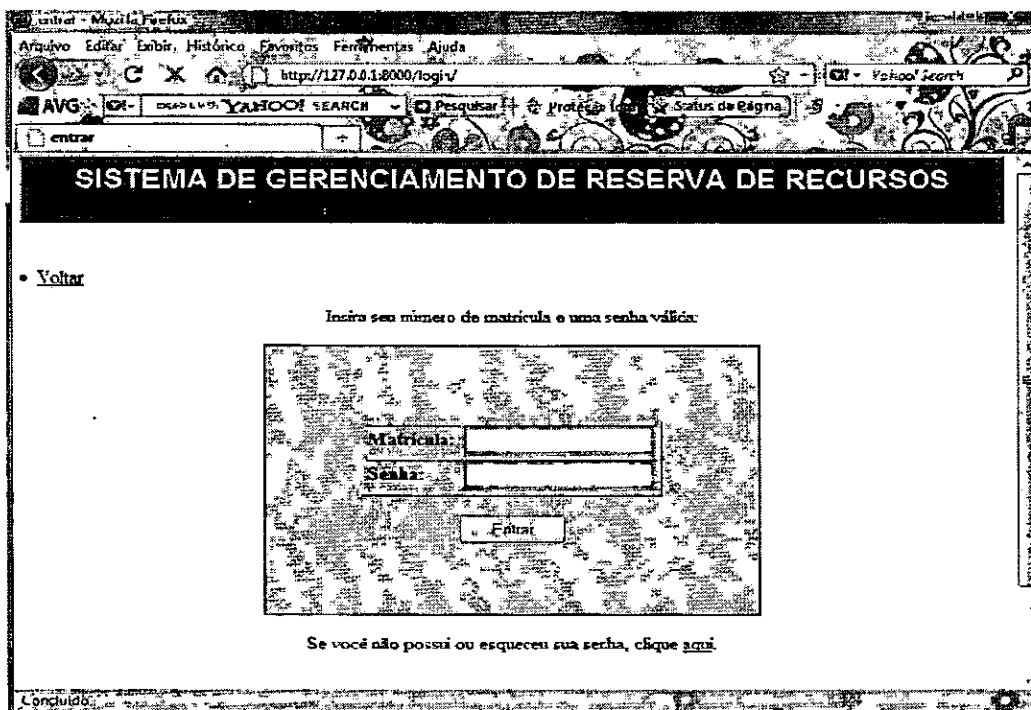


Figura 21: Tela de login para usuário

Fonte: Primária.

Caso ainda não tenha uma senha, o aluno tem a opção de clicar no link “aqui” e fazer um rápido cadastro da sua senha. Após ser fornecido um número de matrícula e senha válidos, o usuário vê na tela uma página com as opções do que pode fazer.

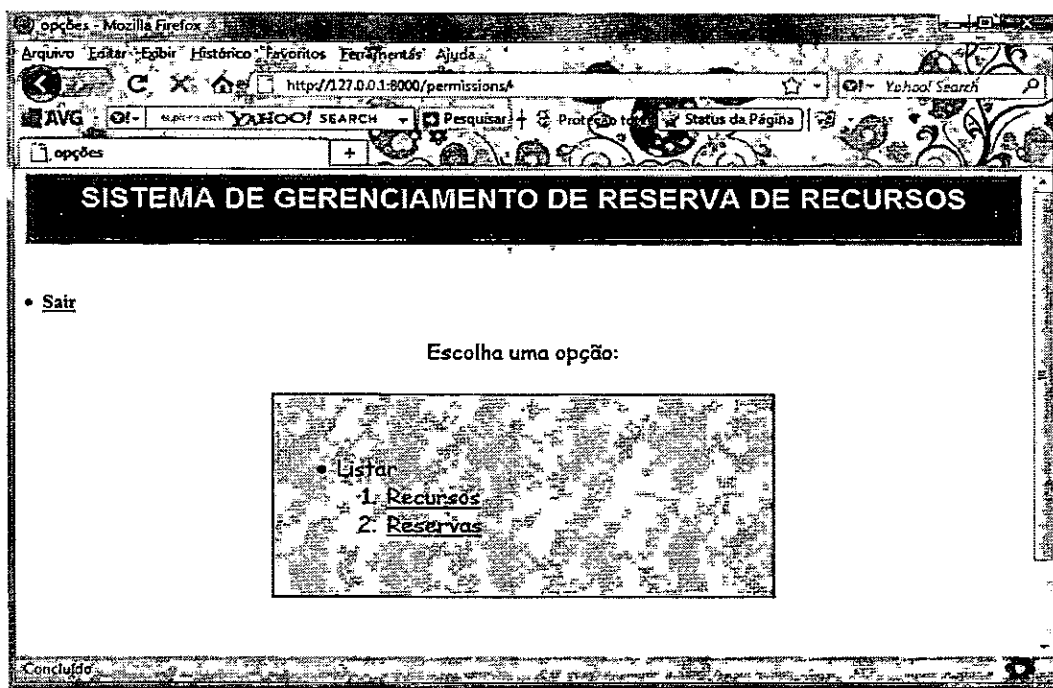


Figura 22: Tela de opções do usuário

Fonte: Primária.

Clicando no link “recursos”, a página exibida será:

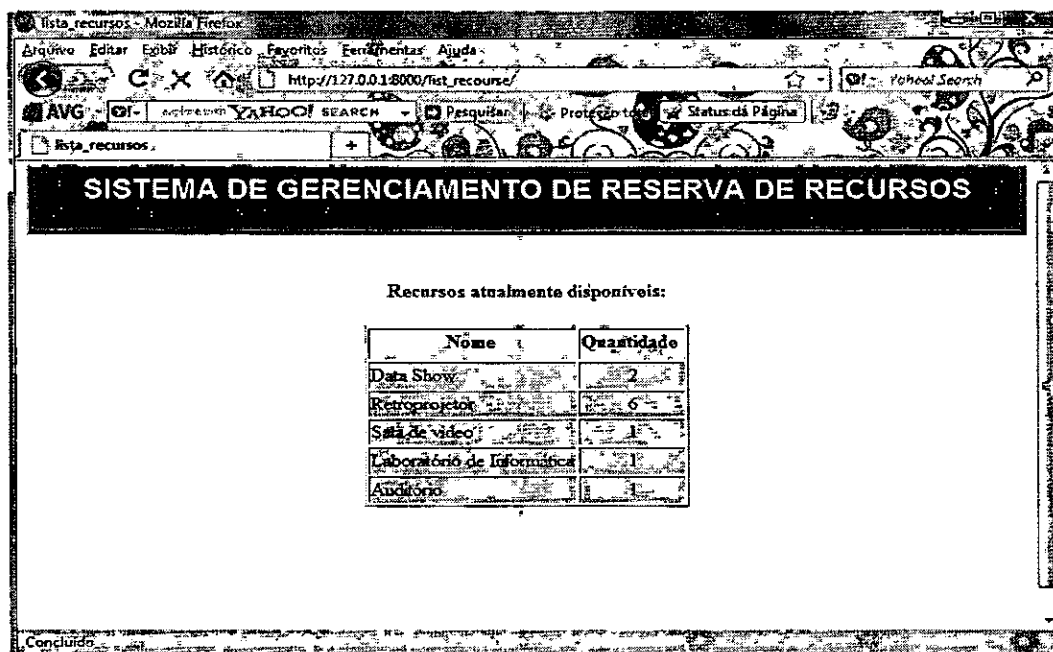
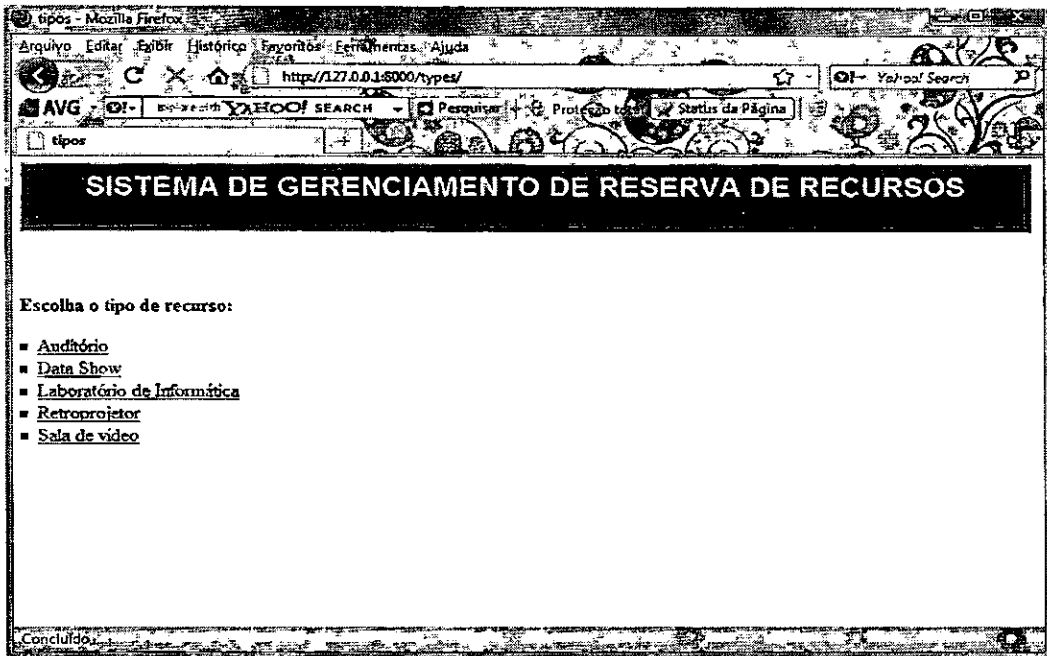


Figura 23: Tela exibe uma lista dos recursos

Fonte: Primária.

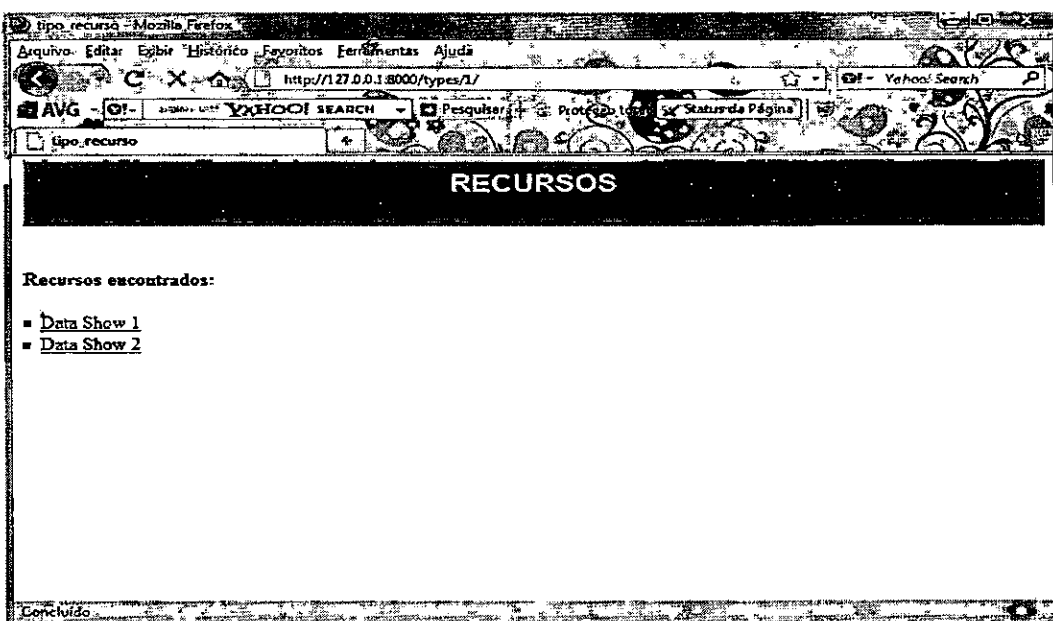
Essa página meramente apresenta os recursos da IES e a quantidade de cada um. Voltando para a página de opções e clicando no link “reservas”, a página exibida será:



**Figura 24:** Tela exhibe uma lista com os tipos de recursos

*Fonte: Primária.*

Primeiramente, são exibidos os tipos de recursos existentes. O usuário deve escolher qual o tipo do qual deseja obter a lista de reservas feitas. Por exemplo, escolhendo o tipo “data show”, leva a página seguinte:



**Figura 25:** Tela exhibe os recursos relativos ao tipo escolhido

*Fonte: Primária.*

Como existem dois objetos desse tipo de recurso, o usuário deve escolher nesse momento de qual recurso ele deseja checar se há ou não reservas feitas e se houverem, observar os horários e datas.

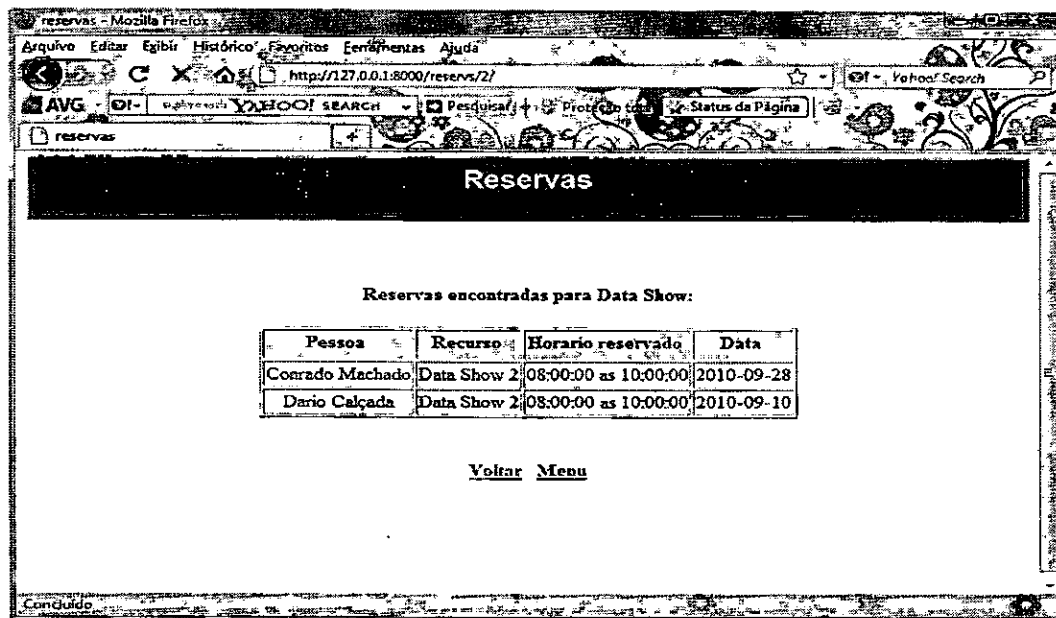


Figura 26: Tela exibe uma lista das reservas feitas para o recurso escolhido

Fonte: Primária.

Após checar as reservas existentes, o usuário poderá fazer a sua própria reserva. Essa etapa fica a cargo do operador do sistema, que é o único autorizado a fazer reservas. Dessa forma, volta-se à página inicial para observar como esse processo é dado.

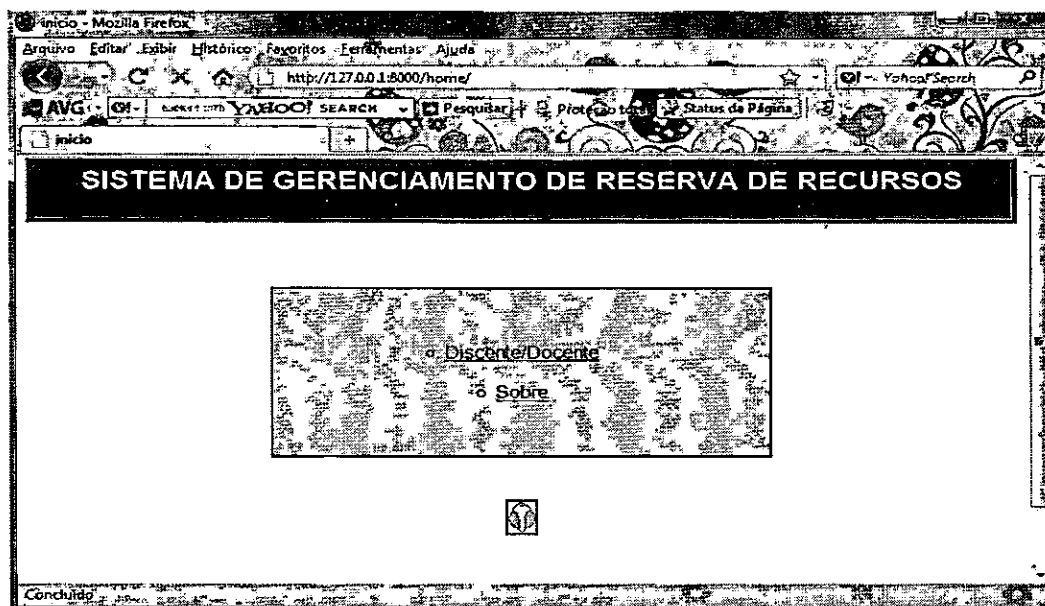


Figura 27: Tela inicial da aplicação

Fonte: Primária.

Clicando sobre o ícone da imagem, é exibida uma página onde o operador do sistema (já devidamente cadastrado), irá fornecer seu login e senha. Esse operador representa um funcionário ou um grupo de funcionários que estão autorizados a fazer as reservas dos recursos para os alunos/professores.



Figura 28: Tela de login do operador  
Fonte: Primária.

A partir dessa etapa, o operador se identifica e é direcionado para uma página que exibe suas possíveis ações:

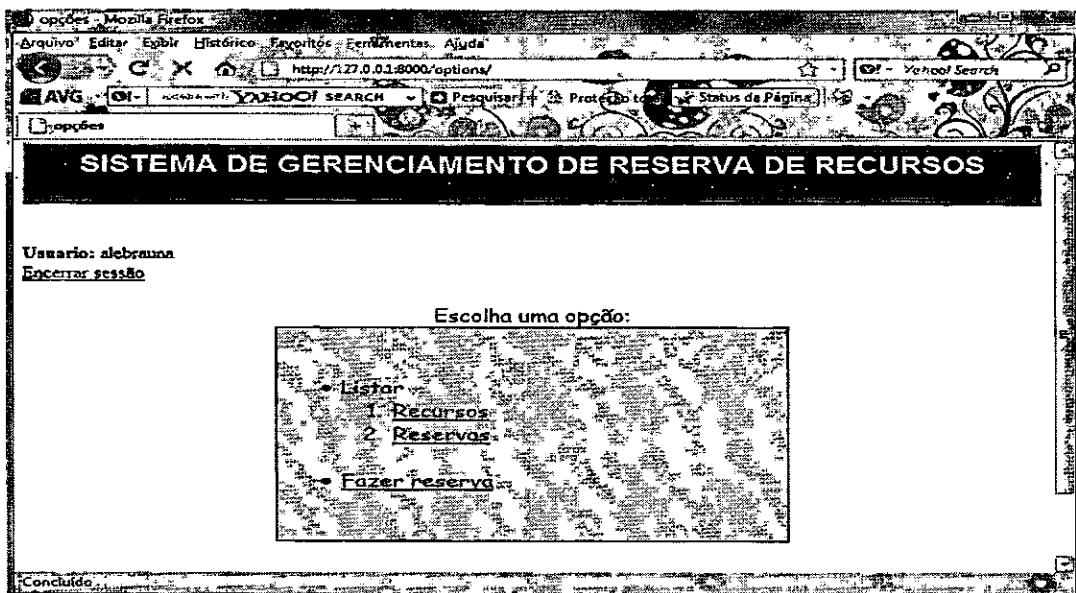


Figura 29: Tela de opções do operador  
Fonte: Primária.

O operador pode listar todos os recursos da IES clicando em “recursos”. Essa página é a mesma citada anteriormente.

A partir do clique no link “fazer reserva”, o operador abre uma página onde é feita uma pesquisa pelo número de matrícula da pessoa que solicita a reserva. Essa parte do processo pode ser descrita pelos diagramas de sequência e de estados encontrados em anexo.

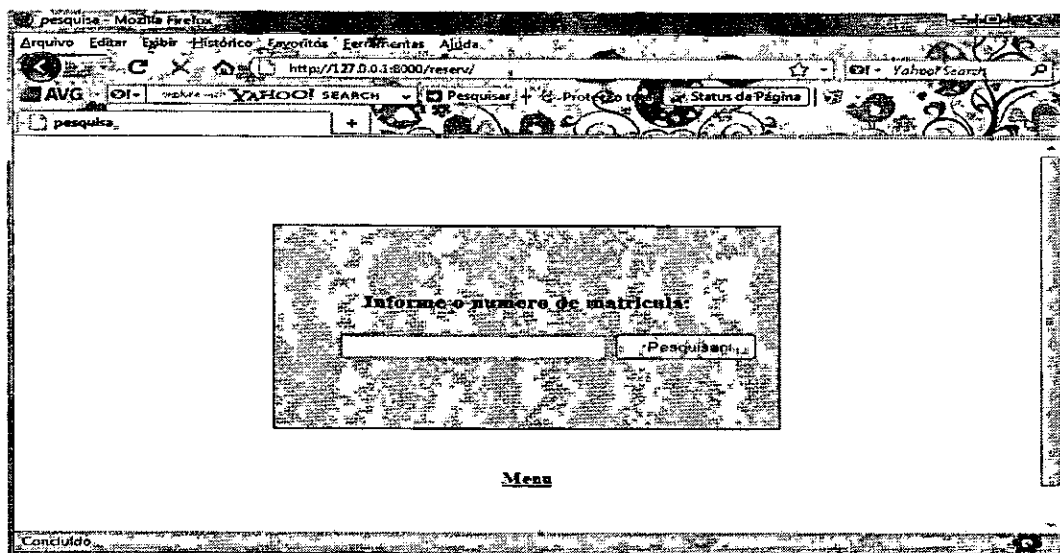


Figura 30: Tela de pesquisa

Fonte: Primária.

O número de matrícula é inserido e como retorno, tem-se uma página com um formulário pronto para efetuar a reserva:

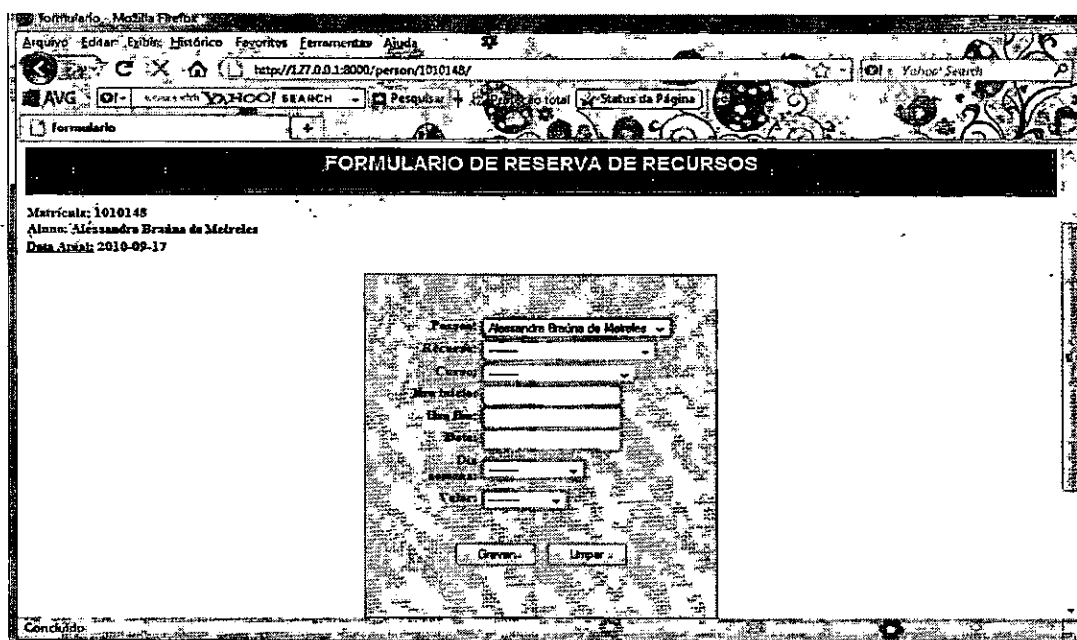
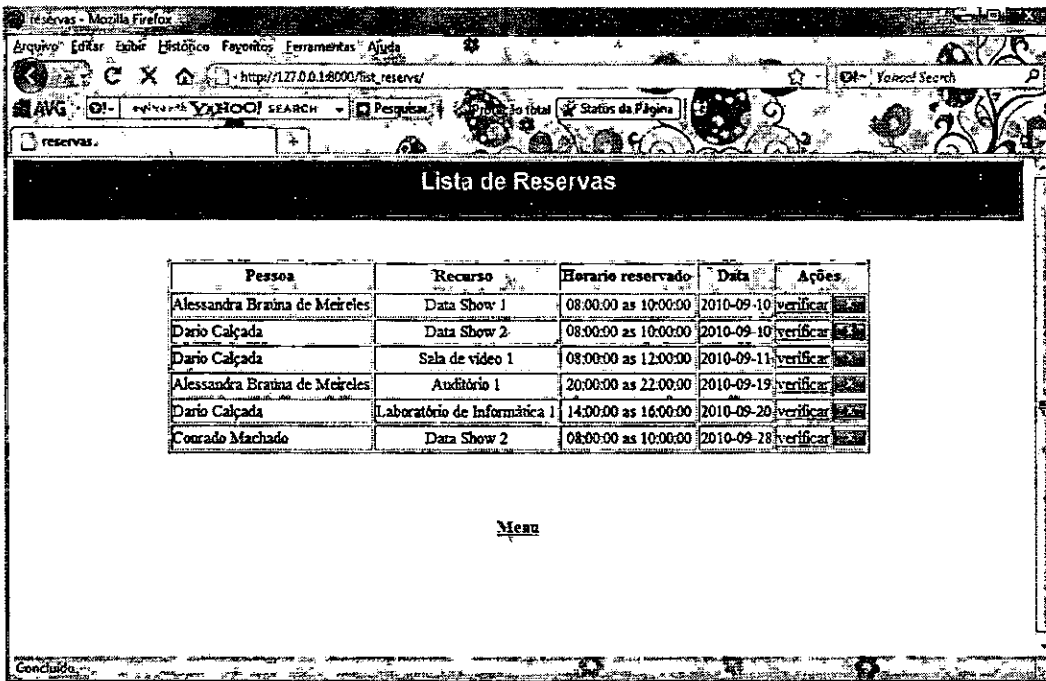


Figura 31: Tela exibe um formulário para registro de reservas

Fonte: Primária.

O operador preenche todos os campos do formulário de acordo com as informações fornecidas pela pessoa, caso aja alguma restrição, como uma reserva idêntica, o próprio formulário exibe os erros na página e não permite que o registro da reserva seja salvo na base de dados. Caso contrário, a reserva é devidamente efetuada e seu registro pode ser conferido na lista de reservas, a partir do link “reservas” encontrado no menu.

Ao clicar sobre o link de “reservas” o operador é direcionando para uma página com uma lista das reservas feitas, organizadas por ordem decrescente de data. A reservas são exibidas numa tabela, cada uma em uma linha, e para cada uma delas existe a opção de visualizar e deletar. A primeira de “visualizar” mostra as informações pertinentes à pessoa que fez a reserva e a segunda de “deletar” apaga o registro da reserva da base de dados.



The screenshot shows a web browser window with the title 'reservas - Mozilla Firefox'. The address bar shows 'http://127.0.0.1:8000/list\_reserva/'. The page content includes a search bar and a table titled 'Lista de Reservas'. Below the table is a 'Menu' link.

Pessoa	Recurso	Horario reservado	Data	Ações
Alessandra Brauna de Meireles	Data Show 1	08:00:00 as 10:00:00	2010-09-10	verificar
Dario Calçada	Data Show 2	08:00:00 as 10:00:00	2010-09-10	verificar
Dario Calçada	Sala de vídeo 1	08:00:00 as 12:00:00	2010-09-11	verificar
Alessandra Brauna de Meireles	Auditório 1	20:00:00 as 22:00:00	2010-09-19	verificar
Dario Calçada	Laboratório de Informática 1	14:00:00 as 16:00:00	2010-09-20	verificar
Conrado Machado	Data Show 2	08:00:00 as 10:00:00	2010-09-28	verificar

[Menu](#)

Figura 32: Tela exibe uma lista de todas as reservas feitas

Fonte: Primária.

Assim como descrito no diagrama de caso de uso, o administrador do sistema é responsável por todas as funções. Ele utiliza os recursos da administração do Django. Na página da administração existem as tabelas relativas à aplicação, que foram criadas no arquivo *models.py* e registradas no *admin.py*. Cada linha da tabela que tem o nome da aplicação se refere a uma model. E para cada uma há dois links associados: adicionar e modificar.

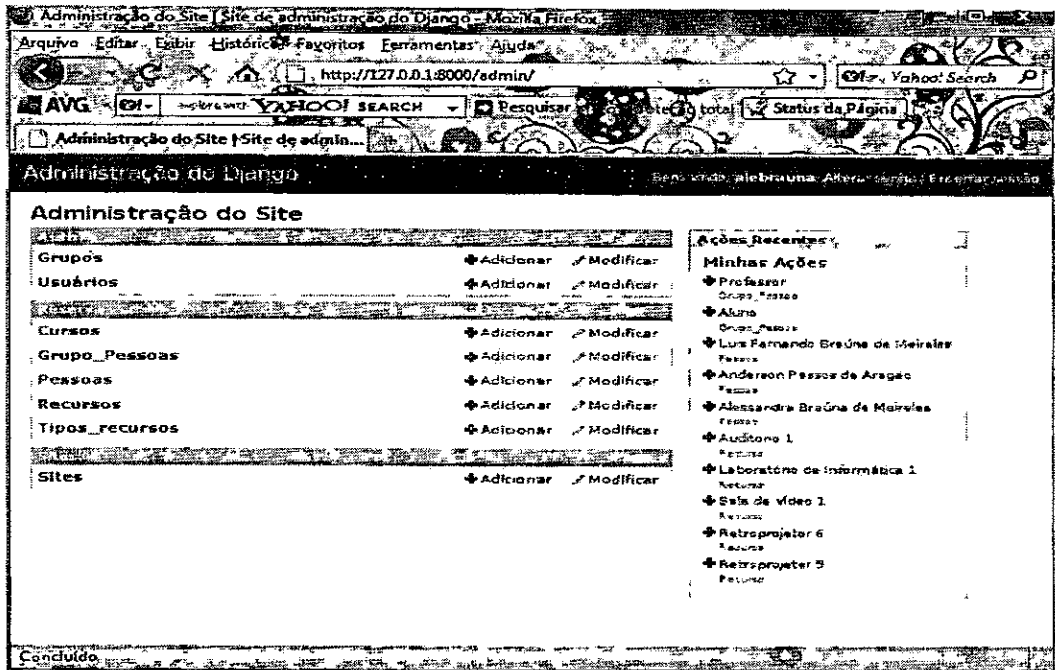


Figura 33: Tela da administração do Django para a aplicação

Fonte: <<http://127.0.0.1:8000/admin/>>.

Por exemplo, clicando sobre “adicionar” é exibida uma página com um formulário, com os campos que já tinham sido definidos nas models, para a criação de um novo curso. Para cada objeto são as mesmas opções, ou seja, funciona da mesma maneira.

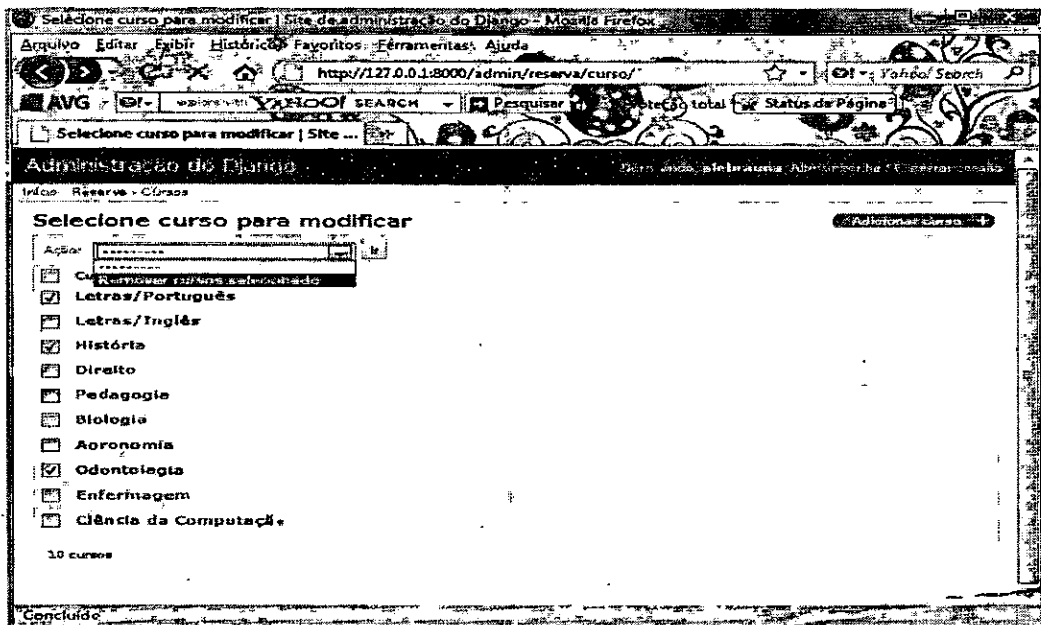
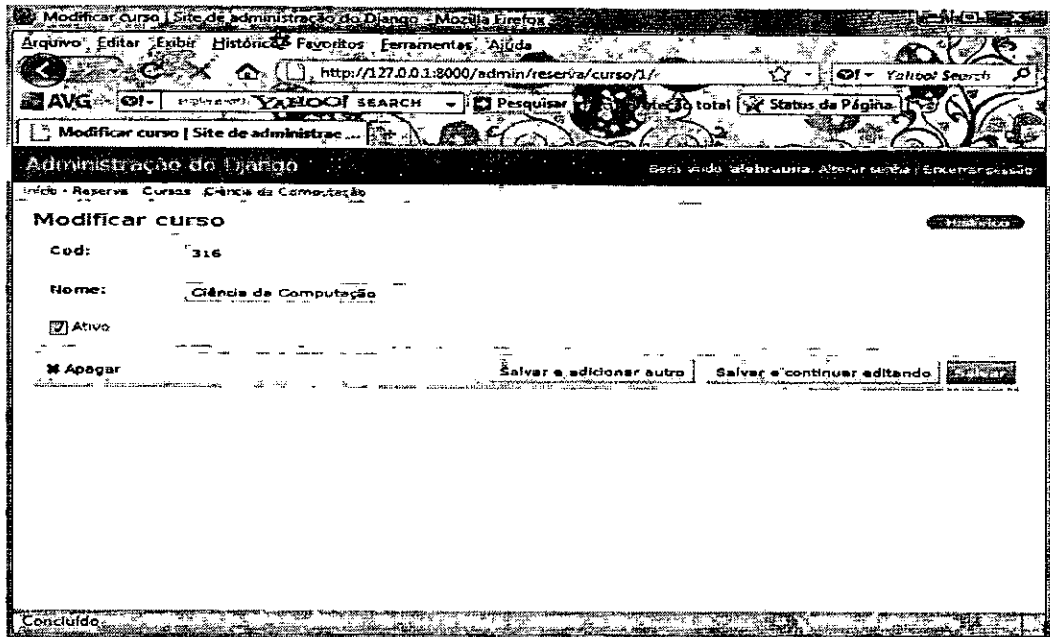


Figura 34: Tela exibe lista de mudanças para objetos da aplicação

Fonte: <<http://127.0.0.1:8000/admin/>>.



Ao clicar sobre o link “modificar”, surge uma página com a lista dos objetos criados. É possível marcar os objetos e selecionar a ação de remover ou clicar sobre um deles, nesse caso aparecerá uma página para edição do objeto.



**Figura 35:** Tela exibe formulário de edição para objeto da aplicação  
*Fonte:* <<http://127.0.0.1:8000/admin/>>.

## 7 CONSIDERAÇÕES FINAIS

Atualmente a maioria das plataformas de desenvolvimento web proporciona ao programador soluções eficazes relativas à segurança, design, interação e funcionalidades úteis. As ferramentas disponíveis nessas plataformas possibilitam uma manutenção fácil e a construção de códigos mais compreensíveis. Aos poucos, elas vêm eliminando o trabalhoso acesso ao banco de dados, dispensando o uso da SQL para manipulá-los.

Django é um proeminente membro de uma nova geração de frameworks web. Desde a sua criação, o framework vem se consolidando e ganhando cada vez mais adeptos, sendo adotado por várias empresas na criação de seus web sites. Essa tecnologia transforma o desenvolvimento de aplicações web em uma atividade menos repetitiva, economizando tempo e focando na construção de suas partes mais importantes.

Considerando que o processo de reserva de recursos utilizado dentro do Campus Prof. Alexandre Alves de Oliveira (UESPI-Parnaíba) é pouco eficiente, a implantação de um sistema web para o gerenciamento das reservas possibilitaria um maior controle dos empréstimos dos recursos aos alunos e professores da instituição, e ainda concentraria essa tarefa de reserva a um grupo específico de funcionários autorizados. Os alunos e professores também se beneficiariam, podendo fazer consultas por conta própria sobre os recursos disponíveis pela IES.

No entanto, para que seja possível a implantação do sistema, a UESPI necessitaria modificar alguns elementos de sua regulamentação referente à locação dos recursos, pois estes ainda são um pouco confusos e sem padronização, para que possam atender adequadamente aos quesitos de organização, segurança, agilidade e eficácia. Essas modificações aumentariam a viabilidade da utilização do software.

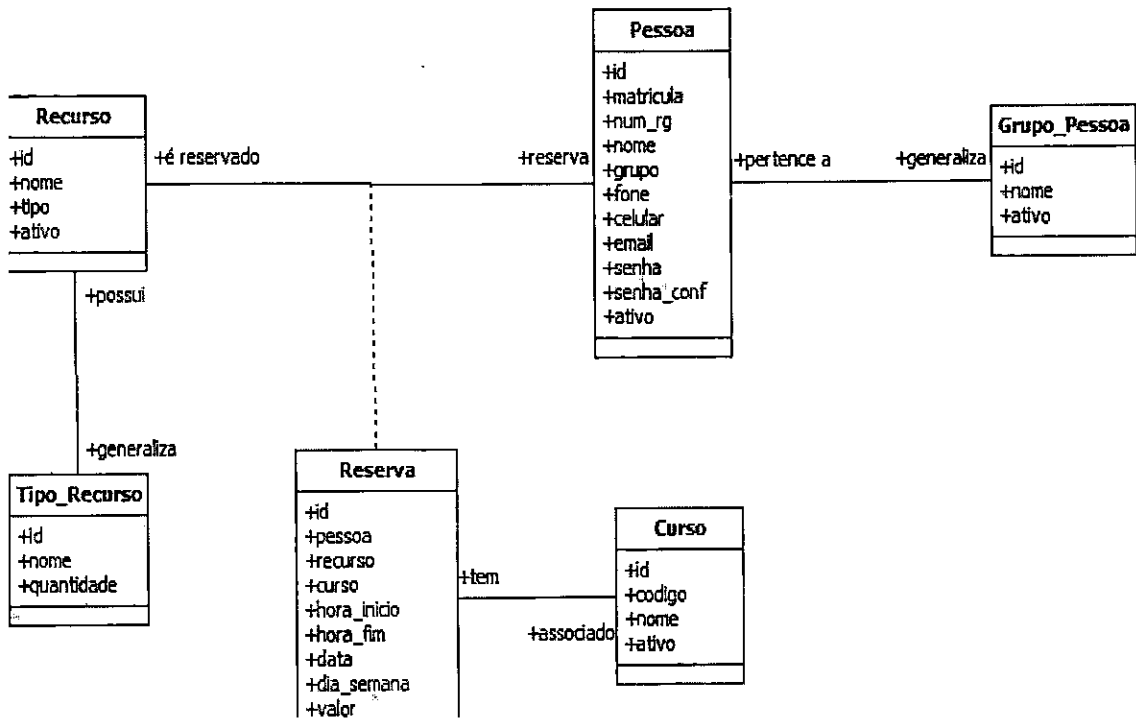
Reconhecendo as características e vantagens apresentadas a respeito do framework Django, a sua adoção para desenvolver o sistema web de gerenciamento de reservas se mostra perfeitamente cabível. Como sugestão de trabalhos futuros o software trataria exceções que não puderam ser consideradas durante o desenvolvimento, devido aos próprios regulamentos do processo de reserva da instituição.

## REFERÊNCIAS BIBLIOGRÁFICAS

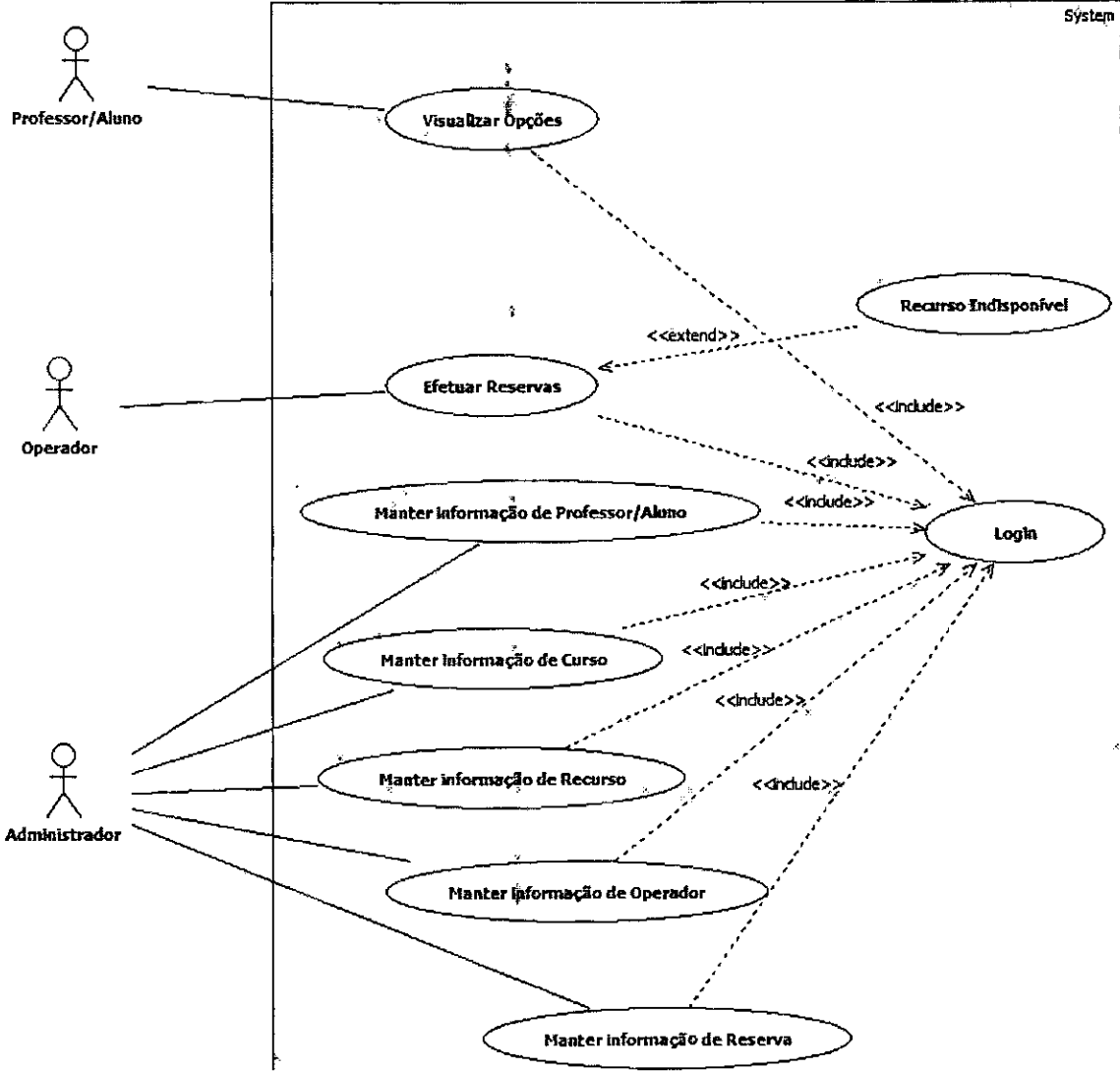
- ALCHIN, Marty. **Pro Django**. 1 ed. New York: Apress, 2009. 311 p.
- BEZERRA, Eduardo. **Princípios de Análise e Projetos de Sistemas com UML**. 3 ed. Rio de Janeiro: Campus, 2007. 290 p.
- BORGES, Luis Eduardo. **Python para desenvolvedores**. 2ed. Rio de Janeiro: O'Reilly, 2010. 360 p.
- BRANDÃO, José Mário Neiva. **Aprendendo Django no Planeta Terra**. Disponível em: <[www.aprendendodjango.com](http://www.aprendendodjango.com)>. Acesso em: 26 de maio. 2010.
- FORD, Jerry Lee Jr. **HTML, XHTML, and CSS for the Absolute Beginner**. 1 ed. Boston: Cengage, 2010. 433p.
- HOLOVATY, Adrian; KAPLAN-MOSS, Jacob. **The Definitive Guide to Django: Web Development Done Right**. 2 ed. New York: Apress, 2009. 538 p.
- \_\_\_\_\_. **The Django Book**. Disponível em: <<http://www.djangobook.com/>>. Acesso em: 15 de mai 2010.
- \_\_\_\_\_. **Django documentation**. Disponível em: <<http://www.djangobrasil.org/>>. Acesso em: 10 de jul. 2010.
- HOLZSCHLAG, Molly E. **250 HTML and Web Design Secrets**. 1 ed. Indianapolis: Wiley, 2004. 434 p.
- LUTZ Mark; ASCHER David. **Aprendendo Python**; Tradução João Tortello. 2.ed. Porto Alegre: Bookman, 2007. 568 p.
- NEWMAN, Scott. **Django 1.0 Template Development**. 1 ed. Birmingham: Packt Publishing, 2008. 271 p.
- PFAFFENBERGER, Brian; SCHAFER, Steven M.; WHITE, Charles; KAROW, Bill. **HTML, XHTML, and CSS Bible**. 3 ed. Indianapolis: Wiley, 2004. 843 p.
- ROCHA, Helder Lima Santos da. **Criação de Web Sites I – tecnologias de apresentação**. 1 ed. São Paulo: Ibpinet, 2000. 140 p.
- \_\_\_\_\_. **Web design e HTML avançado**. 4 ed. São Paulo: Ibpinet, 2000. 129 p.
- ROSSUM, Guido van. **The History of Python**. Disponível em: <<http://python-history.blogspot.com/2009/01/personal-history-part-1-cwi.html>>. Acesso em: 01 de mai. 2010.

## **APÊNDICES**

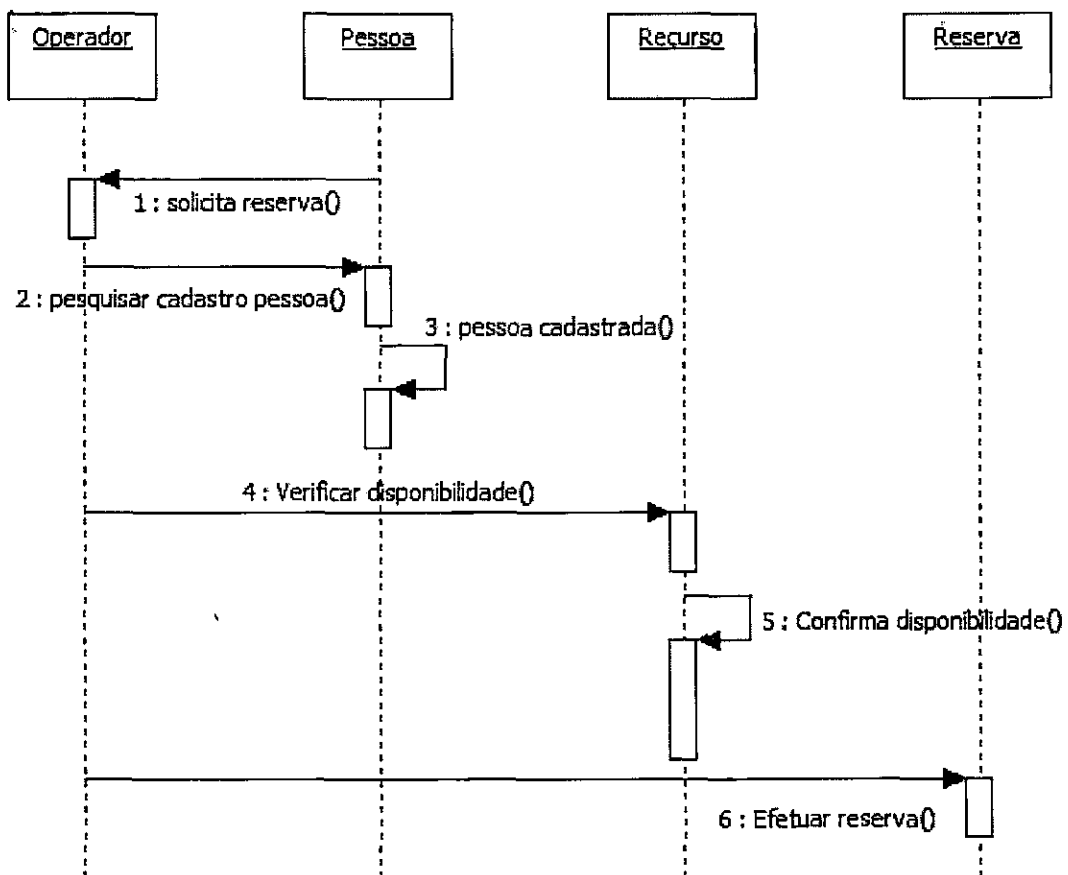
## APÊNDICE A: DIAGRAMA DE CLASSES



# APÊNDICE B: DIAGRAMA DE CASOS DE USO



## APÊNDICE C: DIAGRAMA DE SEQUÊNCIA



## APÊNDICE D: DIAGRAMA DE TRANSIÇÃO DE ESTADO

