UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ELETRÔNICA

SISTEMA DE GERENCIAMENTO BIBLIOTECÁRIO UTILIZANDO INTERFACE WEB/SMS

Autor: _____
Rafael Oliveira Monteiro

Orientador: _____
Prof. Antônio Cláudio Gómez de Sousa

Examinador: _____
Prof. Marcelo Luiz Drumond Lanza

Examinador: _____
Prof. Ricardo Rhomberg Martins

Examinador: _____
Prof. Sergio Palma da Justa Medeiros

DEL
Novembro de 2007

Dedicatória

Dedico este trabalho a todos que me ajudaram durante esses cinco anos de luta para minha formação de graduação. Nisto incluo a minha família, meus professores e os amigos da faculdade e de fora dela.

Agradecimentos

Agradeço primeiramente à minha família pela paciência e ajuda prestada em toda trajetória de formação de caráter e valores que possuo.

Agradeço também ao meu amigo Frederico Andrade de Homobono Balieiro com quem comecei o desenvolvimento do projeto.

Agradeço ao professor Antônio Claudio Gomes de Souza pela orientação desse projeto e aos professores Marcello Luiz Rodrigues de Campos, Sérgio Palma da Justa Medeiros, Ricardo Rhomberg Martins e Marcelo Luiz Drumond Lanza, cujas disciplinas tiveram ligação direta com meu projeto.

Resumo

O trabalho aqui apresentado é um projeto de final de curso na graduação Engenharia Eletrônica e de Computação da Universidade Federal do Rio de Janeiro.

A área de trabalho escolhida para o desenvolvimento do projeto foi a computação e o objetivo principal foi o desenvolvimento de uma aplicação *web* capaz de gerenciar uma biblioteca, tendo como funcionalidade adicional o envio de mensagens para os usuários finais. Mensagens estas que podem ser enviadas via *email* ou *sms*.

O sistema utiliza um servidor *web (apache)* rodando em uma máquina com sistema operacional *linux*. O sistema funciona através de páginas desenvolvidas na linguagem *php*, integradas com um bando de dados *mysql*. Além disso, existem trechos do sistema que funcionam em cima de *shell scripts* e programas em C.

Como um dos tipos de mensagem é o *sms*, utilizamos também um modem *gsm*, com um chip, que será o remetente de todos os alertas enviados. Da mesma forma, para os alertas através de *email*, foi criada uma conta no servidor *yahoo*. O sistema é capaz de se conectar nesse servidor para enviar as mensagens.

A aplicação desenvolvida permite ao usuário final as funções básicas de busca e reserva de livros no sistema. Além destas, permite ao mesmo selecionar livros como favoritos e alterar seus dados pessoais.

A aplicação permite que o administrador do sistema cadastre, modifique e exclua usuários, autores, editoras e livros. Além disso, o mesmo poderá agendar o envio das mensagens.

Essas mensagens são geradas automaticamente pelo sistema nos casos pré-definidos, como avisar o cliente que ele deve devolver um livro, ou que um livro reservado está disponível, porém, podem ser manualmente programadas pelo administrador do sistema, contendo qualquer informação desejada.

Palavras-chave

Aplicação *web*, gerenciamento bibliotecário, *sms*, *email* e programação estruturada.

Índice

1. Introdução

Neste capítulo iremos tratar das considerações iniciais do projeto. Iremos discutir o problema e as motivações que nos fizeram escolher o assunto e estudar as ferramentas necessárias ao desenvolvimento do mesmo.

Para fazer isso, dividimos o capítulo em três subcapítulos. Primeiro apresentamos o problema, depois as principais motivações que nos levaram a escolher esse problema e as motivações relacionadas às tecnologias que deveriam ser empregadas.

1.1. Apresentação do problema

O foco desse projeto foi desenvolver um sistema de gerenciamento bibliotecário capaz de realizar as operações básicas de uma biblioteca de forma remota.

Esse sistema teria que ter as seguintes características básicas: Informatizar qualquer processo existente em uma biblioteca que seja realizado de forma manuscrita, tais como o cadastro de livros e os empréstimos aos seus usuários; permitir acesso remoto de qualquer computador com acesso a *internet*; prover a funcionalidade adicional de envio de mensagens e alertas aos usuários da biblioteca.

A primeira característica seria suprida através de programas de computadores capazes de realizar as operações necessárias em uma biblioteca.

A segunda característica seria suprida utilizando-se um servidor *web* capaz de fornecer acesso aos programas acima citados à usuários em qualquer lugar munido de acesso à *internet*.

Por fim, a terceira característica seria implementada através de alguma tecnologia de comunicação. As tecnologias escolhidas foram as de mensagens curtas via celular (*sms*) e mensagens eletrônicas (*email*).

1.2. Motivação ao problema

Esse é um projeto desenvolvido para conclusão do curso Engenharia Eletrônica e de Computação da Universidade Federal do Rio de Janeiro.

Nesse curso, os alunos têm acesso a algumas bibliotecas. A maioria delas, incluindo a principal biblioteca desse curso, possuem formas manuscritas de controle.

Os livros devem ser procurados de forma manual em gaveteiros com as fichas dos mesmos. Os empréstimos são cadastrados em papéis, assim como suas devoluções.

Por isso, surgiu a idéia de desenvolver um sistema computacional capaz de tornar mais fácil os processos de uma biblioteca, assim como o acesso à mesma, uma vez que o sistema seria implementado via *web*.

Além disso, vale ressaltar que em muitas bibliotecas, principalmente nas de universidades públicas, outros tipos de problemas são encontrados. Um deles é o atraso na devolução dos livros. Para isso, foi proposto o envio de alertas automatizados, capazes de informar os usuários sobre as datas de devolução dos livros. Vale ressaltar que esse é apenas uma das possíveis utilizações dos alertas. Entre outros, podemos citar o aviso de disponibilidade de um livro desejado, a chegada de novos livros e os alertas personalizados que os administradores queiram enviar aos usuários.

1.3. Motivação à tecnologia

Nesse sistema, tínhamos o desejo de utilizar tecnologias já vistas no curso, baseadas em programação e modelagem de banco de dados, assim como a utilização de novas tecnologias não abordadas. Para isso, optamos por escolher o envio de *sms's* como um dos alertas.

Como estagiário de uma empresa de telecomunicações, vi nesse projeto de curso uma grande oportunidade para aprender uma nova tecnologia que me seria útil tanto no trabalho, como no aprendizado acadêmico.

Além dessa tecnologia de envio de mensagens para aparelhos celulares, desejávamos utilizar os conceitos de modelagem de banco de dados, engenharia de software e programação nesse projeto. Essas foram as principais tecnologias usadas no desenvolvimento do sistema.

1.4. Sumário

Nesse primeiro capítulo abordamos os assuntos relevantes à introdução ao problema. Nos próximos capítulos iremos abordar os assuntos relacionados ao desenvolvimento do mesmo, desde as tecnologias utilizadas (capítulo 2) até as conclusões (capítulo 8).

Primeiramente, iremos abordar todas as tecnologias que serviram como motivação e que foram usadas no sistema. São estas o *sms* e o *email*, ambas servindo como meio de comunicação com o cliente. Assuntos esses que se encontram no segundo capítulo.

Uma vez tratadas as tecnologias usadas no projeto, iremos discursar sobre a metodologia utilizada no desenvolvimento do projeto.

Por esse se tratar de uma aplicação computacional, mais especificamente um *software*, iremos abordar o método de Engenharia de *Software* utilizado para desenvolver o sistema. Dentre as metodologias possíveis, escolhemos a baseada na análise estruturada. Essa metodologia será tratada no terceiro capítulo. Além disso, tratamos também do ambiente e das ferramentas usadas no desenvolvimento do projeto.

Uma vez descrita a metodologia que será utilizada no projeto, podemos fazer uma análise do problema. No capítulo quatro fazemos essa análise, além de referenciar versões mais completas nos apêndices da mesma.

Com o projeto analisado, estamos aptos a escrever sobre como implementamos o mesmo. No capítulo cinco abordamos o projeto e sua implementação.

Já no capítulo seis, abordamos todo o processo de desenvolvimento do projeto, onde detalhamos todas as fases do trabalho, os problemas encontrados e as soluções aplicadas a esses problemas.

Por fim, nos dois últimos capítulos, mostramos os resultados e concluímos o trabalho, sendo o primeiro mostrado no capítulo sete e o segundo no capítulo oito. Nesses capítulos, descrevemos como o sistema final foi terminado, fazemos uma avaliação das tecnologias, ferramentas e metodologias utilizadas, assim como apontamos os pontos de grande aprendizado. Além disso, propomos melhorias ao sistema.

2. Tecnologias

Neste capítulo iremos discutir as duas tecnologias utilizadas na implementação do sistema, incluindo a teoria necessária ao entendimento do sistema.

Para fazer isso, dividimos o capítulo em dois subcapítulos. Em cada um desses subcapítulos tratamos cada uma das tecnologias presentes nesse sistema. São elas: *Email* e *sms*.

Ambas as tecnologias foram usadas para fazer uma interface de comunicação com o cliente da biblioteca. Elas são usadas como meio de envio dos alertas e informações da biblioteca.

Tanto o *sms* como o *email* são meios de comunicação acessíveis a quase todos e, por isso, acreditamos que usando os dois, atingimos todo o público alvo de uma biblioteca.

Como o uso de *emails* hoje em dia é muito elevado, muitos deles passam despercebidos em nossas caixas postais eletrônicas. Por isso, o *sms* torna-se ainda mais interessante nesse projeto.

2.1. *Sms*

Nosso sistema terá um modem capaz de enviar mensagens para outros celulares. No caso, teremos uma aplicação interagindo com o modem, que por sua vez se comunicará com a operadora, enviando as mensagens para a mesma, como se fosse um aparelho celular. Por isso, facilitamos o nosso projeto, visto que a idéia original era criar uma conexão através da internet com as operadoras (VPN's).

Nos próximos tópicos iremos abordar o funcionamento básico de envio de uma mensagem para celular, o modem utilizado no projeto e as restrições das mensagens a serem enviadas pelo sistema, assim como o mecanismo de agendamento.

2.1.1. Funcionamento do envio de uma mensagem curta

As mensagens são enviadas para uma Central de Serviço de Mensagem Curta (SMSC), que armazena a mensagem para transmissão futura quando não é possível enviá-la imediatamente (algumas centrais também provêem a opção de fazer apenas uma tentativa). Não há garantia de que uma mensagem realmente seja entregue, e

comumente há perda ou atraso na entrega (podem-se requerer confirmações, que, no entanto, não são confiáveis no caso de falhas).

Na operadora são feitos alguns processamentos. Dentre eles, citamos a localização do aparelho de destino, sua disponibilidade para o recebimento, o pedido de confirmação de entrega da mensagem e a checagem de créditos do remetente.

No caso do nosso sistema, a única consideração a ser feita é relativa a checagem de crédito, visto que usaremos um cliente pré-pago de uma operadora que pode eventualmente ficar sem créditos, impossibilitando o envio das mensagens. Nos demais casos, não haverá preocupação, visto que os alertas serão apenas informativos e não é fundamental que haja uma garantia de sua entrega.

O *chip* a ser utilizado é variável. Por isso não citaremos nenhum deles. Vale ressaltar, que o remetente da mensagem estará sempre associado a um desses números dos *chips*.

2.1.2. *Modem gsm*

Para a comunicação com a operadora de telefonia móvel será utilizado um *modem gsm*. Para o funcionamento do mesmo precisamos ter um *chip* com créditos e posicioná-lo em uma área com um bom nível de sinal.

A comunicação entre o sistema e o *modem* se dá através de alguns comandos AT. No entanto, como existem bibliotecas que já integram esses comandos, em nosso trabalho nos concentraremos apenas na utilização das mesmas.

2.1.3. Tamanho da mensagem

Um outro ponto relevante no planejamento do projeto é o tamanho da mensagem. Essas são restringidas ao tamanho máximo de 160 caracteres. Desse modo, mensagens maiores que essas não poderão ser cadastradas para envio.

2.1.4. Agendamento da mensagem

Nesse subcapítulo explicaremos como as mensagens serão agendadas.

Esse agendamento será feito utilizando arquivos textos para cada mensagem a ser enviada. Cada arquivo conterá: o número de destino, na primeira linha; a mensagem, na segunda linha; e, por fim, o código de identificação do alerta, na última linha. O nome do arquivo texto será formatado da seguinte maneira:

SMS_DATA_HORARIO_ID.dat, onde DATA, HORARIO e ID serão substituídos, respectivamente, pelos valores de data do envio, horário do envio e código do alerta.

Os arquivos ficarão armazenados em uma pasta e um script os monitorará. Sempre que for detectado que o alerta deve ser enviado (comparando-se com o horário e data do sistema), o mesmo será enviado, chamando-se o programa de envio de *sms*.

2.2. *Email*

A estratégia a ser utilizada para o envio dos *emails* será criar um *link* entre o sistema e um provedor de *emails*. Para que isso possa ser realizado será necessário utilizar o protocolo de comunicação *smtp*, um programa desenvolvido em cima da estrutura de comunicação *sockets* e o servidor propriamente dito. São esses os próximos tópicos abordados.

2.2.1. Protocolo *smtp*

Esse é o protocolo usado para envio de *emails*. O Protocolo SMTP é baseado em comandos (enviados pelo cliente) e respostas (enviadas pelo servidor). Todos os comandos SMTP geram uma resposta, e é considerada uma violação do protocolo SMTP o envio de um comando antes que o servidor dê a resposta do comando anterior.

Podemos resumir o mesmo nos seguintes comandos que terão que ser enviados por nossa aplicação:

- Conectar-se no provedor de *email;*
- Troca de mensagens de reconhecimento (HELLO e autenticação);
- Troca de mensagens informando o cabeçalho da mensagem (remetente, destinatário, assunto, data, etc);
- Conteúdo da mensagem;
- Sinalização de termino da comunicação;
- Término da conexão.

2.2.2. *Sockets*

O *sockets* será utilizado para fazer a comunicação entre o nosso sistema e as máquinas do servidor de *email*. O mesmo será desenvolvido respeitando-se as regras definidas pelo protocolo do tópico anterior.

2.2.3. Servidor *yahoo*

Foi criada uma conta nesse provedor. A conta é biblioteca_ufrj@yahoo.com.br. Todos os alertas enviados por *email* serão feitos por esse remetente.

2.2.4. Agendamento do email

Funcionará da mesma maneira que o agendamento de *sms*. No entanto, deverá ser incluída no arquivo uma linha para o título da mensagem, assim como o mesmo deverá ter o nome de EMAIL_DATA_HORARIO_ID.dat, seguindo o padrão já explicado. O programa a ser chamado será o de envio de *emails*.

3. Metodologia

Neste capítulo iremos discutir a metodologia usada no projeto. Essa metodologia foi aprendida e empregada durante uma das disciplinas do curso (Engenharia de Software) e, posteriormente, aproveitada nesse projeto final.

Para fazer isso, dividimos o capítulo em dois subcapítulos. No primeiro discutimos a metodologia estruturada para desenvolvimento de software, que foi a escolhida para esse projeto. No outro, abordamos o ambiente e as ferramentas usadas no mesmo. Sendo estas: Os ambientes Apache e MySQL; As linguagens de programação php, c e shell script; e, por fim, as ferramentas de auxílio erwin e dbdesigner.

3.1. Estruturada

A metodologia escolhida para o projeto foi a estruturada. Essa metodologia propõe uma distinção clara entre dados e funções. Desse modo, o sistema pode ser dividido em duas grandes fases: projeto de dados e projeto de programas.

Um aspecto fundamental da fase de projeto de dados consiste em estabelecer de que forma serão armazenados os mesmos do sistema. A plataforma de implementação para armazenamento de dados mais difundida é a dos bancos de dados relacionais e, portanto, neste projeto, abordaremos o projeto lógico de bancos de dados relacionais.

O projeto estruturado de programas, em sua dimensão de funções, envolve duas grandes etapas: o projeto da arquitetura do sistema e o projeto detalhado dos módulos. Paralelamente, devem ser feitos o projeto de dados e o projeto da interface com o usuário. Em nosso projeto planejamos os módulos e suas interligações.

Para cumprir com o exposto acima, dividimos a estrutura dessa metodologia em dois capítulos: um de análise e outro de projeto. No primeiro, descreveremos o produto e suas funções, assim como as características do usuário. Além disso, abordaremos requisitos específicos do sistema através de eventos, diagrama entidade relacionamento, diagrama de fluxo de dados e diagramas de transição de estado. Concluindo essa primeira etapa, vamos abordar as interfaces externas com usuários e os meios de comunicação. Faremos referência a apêndices.

No capítulo de projeto abordaremos a arquitetura do sistema, incluindo sua decomposição em módulos e a dependência entre os mesmos. Faremos também uma

descrição mais detalhada dos mesmos, fazendo referência a apêndices. Além disso, faremos a decomposição dos dados e como os mesmos estão interligados.

3.2. Ferramentas e ambiente

Nesse subcapítulo tratamos das ferramentas e do ambiente de desenvolvimento e de operação do sistema.

Para fazer isso, dividimos o capítulo em cinco subcapítulos. Em cada um desses subcapítulos tratamos cada uma das ferramentas ou ambiente. São elas: O servidor Apache; o banco de dados MySQL; as linguagens de programação php, C e Shell Script; os programas de modelagem de banco de dados ErWIN e DBDesigner; as configurações de rede.

Toda a hospedagem do sistema será feita em um computador pessoal, conectado à internet através de uma conexão banda larga. A máquina hospedeira roda o sistema operacional linux (Feedora 6).

A conexão na internet é feita através do serviço Velox da OI/Telemar e, por isso, temos um IP variável. Por isso, foi criada uma conta que associa uma url ao nosso IP, que é constantemente atualizado. Desse modo, acessando essa url, garantimos sempre que estamos acessando o servidor. A url é http://romserver.dyndns.org:9000/pf. No entanto, recomenda-se o acesso direto pelo IP do servidor.

3.2.1. Apache

O servidor escolhido para o projeto foi o Apache. O motivo principal é o fato de esse ser o servidor mais utilizado atualmente e o mais adequado para aplicações *web*, como no caso de uma biblioteca virtual.

Como escolheríamos o banco de dados MySQL, citado no próximo subcapítulo, usamos o programa XAMPP, que tem os dois integrados.

Uma vez instalado, o servidor está apto e funcionando corretamente, dependendo apenas de configurações da rede, que serão tratados no quinto subcapítulo (3.2.5).

3.2.2. MySQL

O banco de dados escolhido para o projeto foi o MySQL. A sua escolha é justificada pela facilidade de integração com a linguagem de programação php, que foi a linguagem mais usada no projeto.

Além disso, citamos o excelente desempenho e estabilidade apresentada por esse banco de dados para o projeto em questão, motivando ainda mais a sua utilização.

No entanto, o ponto crítico para a escolha foi o fato do mesmo ser um software livre. Assim como ele, poderíamos usar o PostgreSQL, mas pela facilidade de integração com o php já citada acima, optamos pelo MySQL.

Uma vez escolhido o servidor e o banco, usamos o programa XAMPP, que une os dois. Esse programa tem o Apache e o MySQL integrados e facilita a instalação dos mesmos.

Para fazer todas as operações com o banco, foi utilizada a linguagem SQL, sendo os *scripts* gerados manualmente ou por programas, como os citados no quarto subcapítulo (3.2.4). Todas as operações foram realizadas por linha de comando, seja executando cada *query* individualmente ou chamando arquivos com as queries já montadas. As *queries* de criação do banco estão no apêndice E.

3.2.3. Php, C e Shell Script

As linguagens de programação utilizadas no desenvolvimento do projeto foram: Php, C e Shell Script. A utilização e as motivações estão explicadas nos parágrafos seguintes.

Por se tratar de uma aplicação web, usamos o php, que é uma linguagem muito difundida e que vem crescendo nos últimos tempos. Além dessa característica, citamos o fato de o código não precisar ser compilado para ser usado, como um ponto fundamental para sua escolha.

Em nosso curso, a principal linguagem de computação aprendida é o C, C++. Dessa forma, por se assemelhar muito à essas duas linguagens, o php tornou-se ainda mais adequado no projeto.

A segunda linguagem utilizada no projeto foi o C. Sua usabilidade pode ser explicada pela necessidade do envio dos alertas por *email*. Essa funcionalidade poderia ser implentada com o php também, mas como em uma das disciplinas do curso (TCP-IP) usamos C para envio de *emails*, optamos por utilizar a mesma nesses casos.

Por fim, a última linguagem usada foi o *Shell Script*. Como nosso sistema precisa realizar tarefas periódicas relacionadas aos alertas, que não necessariamente são enviados no momento de seu cadastro no sistema, precisamos criar scripts que monitorassem a necessidade de envio de um alerta em determinado horário. Para isso, utilizamos *scripts* feitos com *Shell Scrip*t rodando em *background* no sistema.

Como todo esse código desenvolvido é muito grande, apenas os trechos mais importantes serão abordados no capítulo de Implementação (cap. 5) e nos apêndices. O código completo pode ser encontrado na versão eletrônica desse relatório.

3.2.4. ErWIN e DBDesigner

Toda a modelagem do banco de dados do sistema foi feita utilizando-se esses dois programas. No inicio do projeto utilizávamos o *ErWIN* e, posteriormente, passamos a utilizar o *DBDesigner*.

O primeiro foi usado por ser a ferramenta recomendada na cadeira Banco de Dados do curso. Com a mesma, conseguimos modelar todo o banco de dados através de um modelo relacional.

A segunda foi utilizada pela facilidade de obtenção dos *scripts* SQL de criação do banco de dados modelado na ferramenta. Exportamos o modelo criado na primeira ferramenta para a segunda.

3.2.5. Configurações de rede

Como já citado anteriormente, a conexão com a internet é feita através do serviço Velox e com o recebimento de um IP dinâmico.

Um dos problemas associados ao IP dinâmico já teve sua solução descrita. Usou-se um servidor DNS, que fornece uma url fixa para o endereço da aplicação. E cada vez que o IP variar, o mesmo é atualizado nesse servidor DNS (www.dyndns.com.br).

O outro problema associado é o uso de uma rede interna. O uso da mesma faz com que a máquina servidora tenha um IP inválido para a internet. Desse modo, foi necessária a configuração do roteador para que apontasse o tráfego requisitado para a máquina servidora. Como o Velox bloqueia a porta padrão http (80), usou-se a porta 9000 para o servidor do sistema.

4. Análise do problema

Nesse capítulo iremos fazer a análise do problema, especificando os requisitos do sistema a ser desenvolvido.

4.1. Descrição geral

O projeto é uma aplicação *web* para gerenciar uma biblioteca. Nesse gerenciamento, além das funções básicas de uma biblioteca, incluiremos funcionalidades de alertas informativos aos clientes.

Dentre as funcionalidades, destacamos a reserva de livro, o banco de favoritos de um usuário e o sistema de busca de títulos.

No que diz respeito aos alertas, teremos o envio de mensagens curtas para celular e mensagens eletrônicas (*emails*).

4.2. Requisitos específicos

Nesse tópico abordaremos os requisitos específicos do sistema. Para tanto, o dividimos em listagem de eventos, diagrama entidade relacionamento, diagramas de fluxo de dados e diagramas de transição de estado.

4.2.1. Eventos

Os eventos do sistema serão:

- Fazer cadastro no sistema;
- Recuperar senha;
- Fazer autenticação;
- Sair do sistema;
- Gerar menu;
- Alterar senha;
- Alterar dados pessoais;
- Cadastrar/Editar/Remover autor;
- Cadastrar/Editar/Remover editora;
- Cadastrar/Editar/Remover livro;
- Cadastrar/Editar/Remover alerta;
- Enviar alerta;

- Procurar livro/usuário/alerta;
- Visualizar livro/usuário/alerta;
- Remover usuário;
- Atualizar reservas;
- Reservar livro;
- Colocar livro nos favoritos;
- Emprestar livro;
- Devolver livro;
- Visualizar reserva/favorito/empréstimo;

4.2.2. Diagrama entidade relacionamento

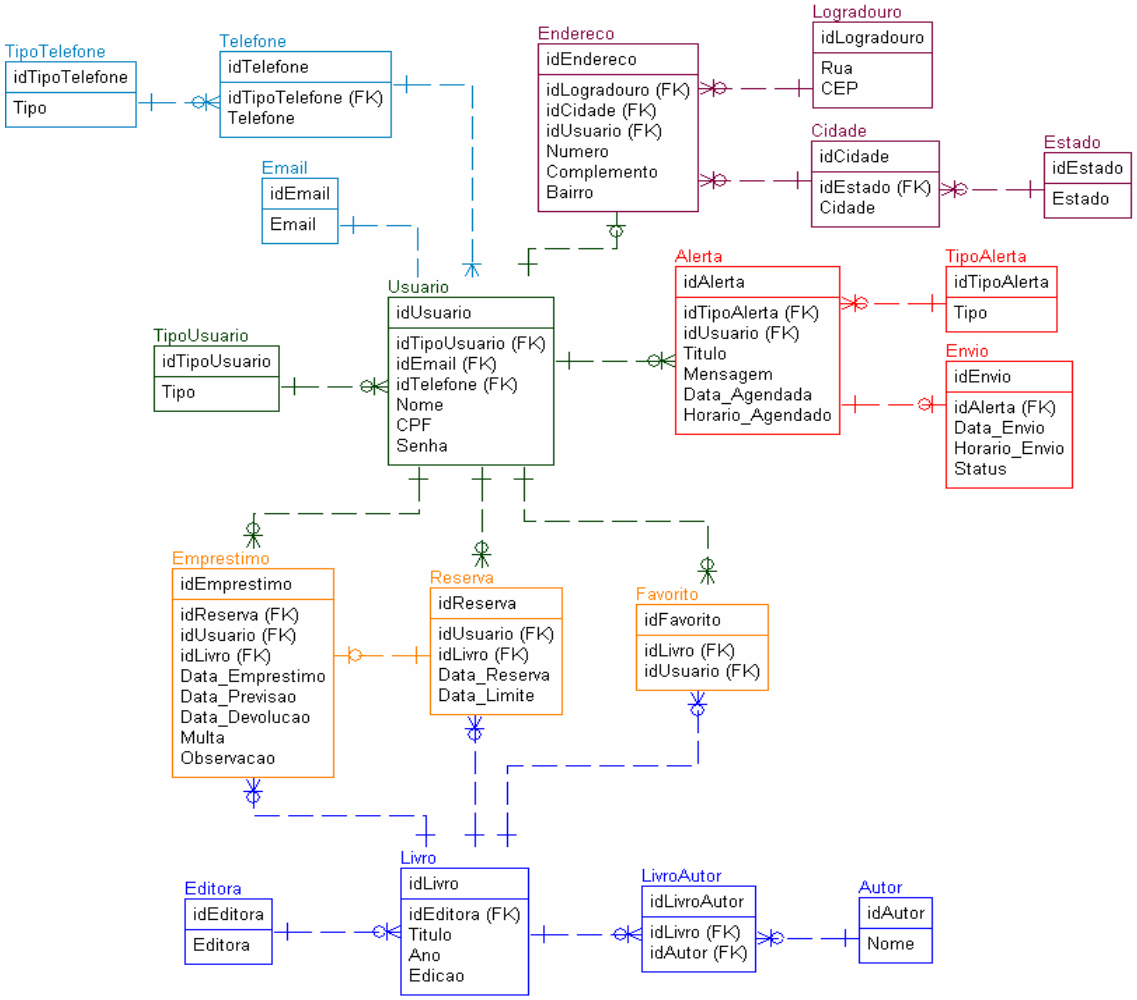

Figura 4.1 – Diagrama entidade relacionamento

É possível inclusive identificar grupos de dados relacionados pela utilização das cores. No próximo capítulo abordaremos os dados em maiores detalhes. No entanto, esse diagrama representa a modelagem essencial do sistema.

4.2.3. Diagrama de fluxo de dados

Por serem muitos, ilustraremos apenas os principais diagramas de fluxo de dados do sistema, sendo que desses a maioria faz uso dos alertas, motivo pelo qual os classificamos como principais. Diagramas que influem na segurança do sistema também são igualmente considerados importantes. Os demais podem ser encontrados no apêndice A.

Cadastrar alerta

O processo cadastrar alerta recebe os campos necessários preenchidos pelo usuário, verifica se estão todos corretos. Caso esteja, verifica a existência do usuário a receber o alerta e, existindo, cadastra-o no sistema.

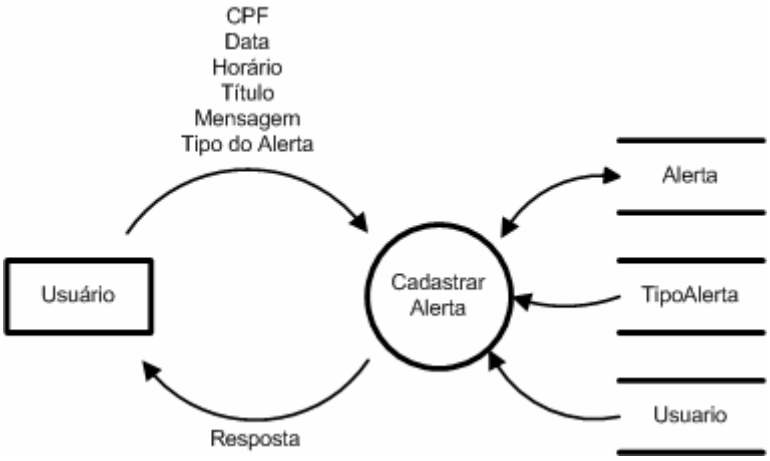

Figura 4.2 – DFD cadastrar alerta

Fazer autenticação

O processo autenticar recebe um cpf e uma senha digitados pelo usuário e procura por um registro no banco de dados. Caso exista, o usuário será autenticado. Caso contrário, o mesmo receberá uma mensagem de erro.

Figura 4.3 – DFD fazer autenticação

Reservar livro

O processo reservar livro recebe o livro que o usuário deseja reservar, verifica a existência de ambos e, existindo, cadastra-o no sistema. Se ocorrer algum erro, o usuário será informado. Além disso, é verificado se o livro já está reservado ou emprestado. Caso esteja, a reserva cadastrada não terá um limite e não será cadastrado um alerta. Se o livro não estiver reservado, nem emprestado, um alerta é cadastrado para informar o cliente no último dia de sua data limite para pegar o livro.

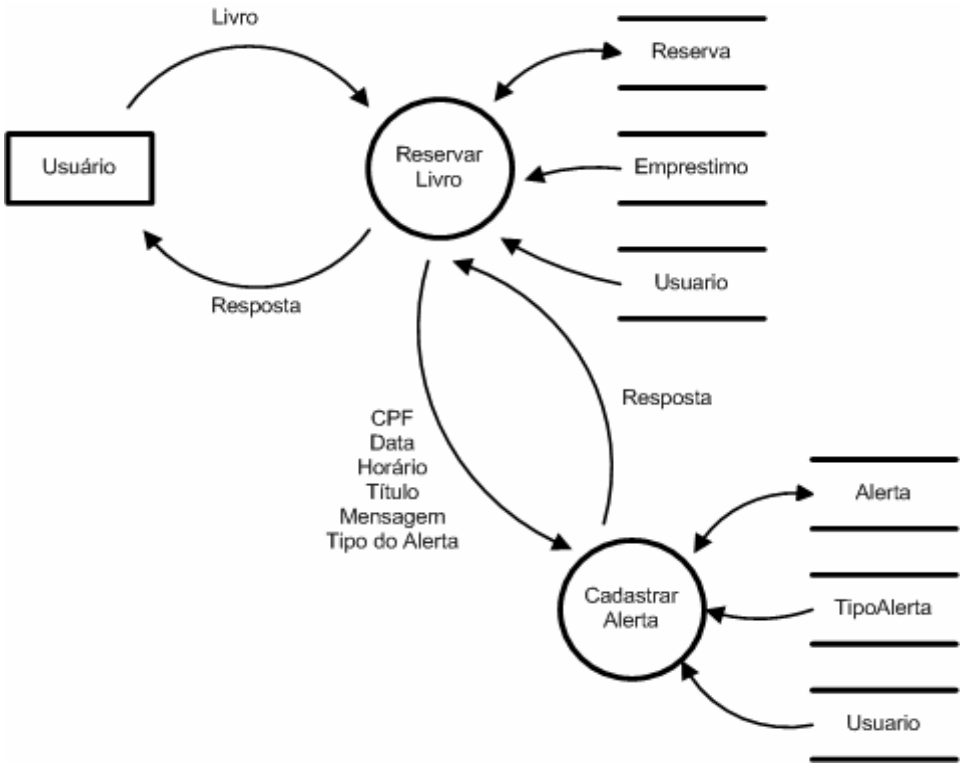

Figura 4.4 – DFD reservar livro

Gerar menu

O processo gerar menu verifica o tipo de usuário e exibe a página com as opções correspondentes ao mesmo.

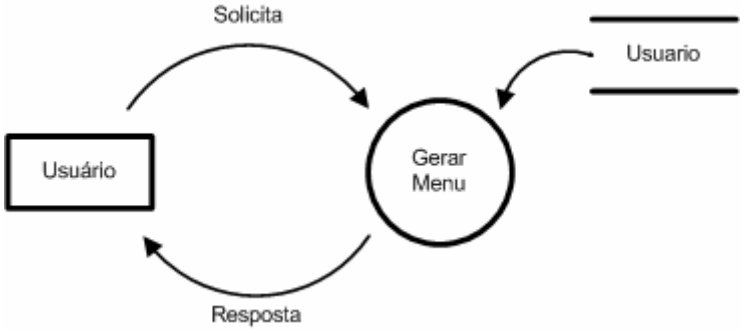

Figura 4.5 – DFD gerar menu

Emprestar livro

O processo emprestar livro recebe o livro que o usuário deseja pegar emprestado, confirma a existência de ambos e, existindo, verifica se o livro já está reservado ou emprestado. Caso esteja reservado, será verificado se a reserva é para o usuário em questão. Se o livro for emprestado com sucesso, um alerta é cadastrado para informar ao cliente, no último dia, a sua data limite de devolução do livro.

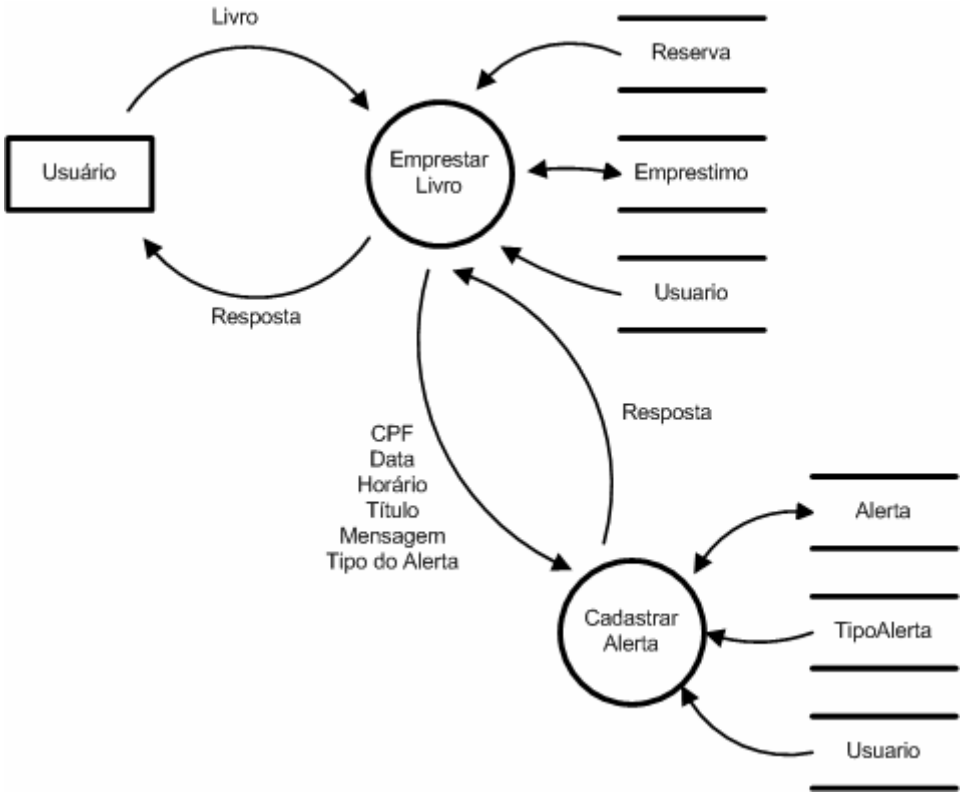

Figura 4.6 – DFD emprestar livro

Atualizar reservas

O processo atualizar reservas procura por reservas que estejam expiradas no sistema e as atualizam. Se alguma reserva nessa situação for encontrada e existirem outras reservas em aberto para o mesmo livro, o próximo cliente da fila de espera será alertado e terá uma data limite cadastrada para sua reserva. Da mesma forma, será cadastrado um alerta para o dia limite dessa reserva.

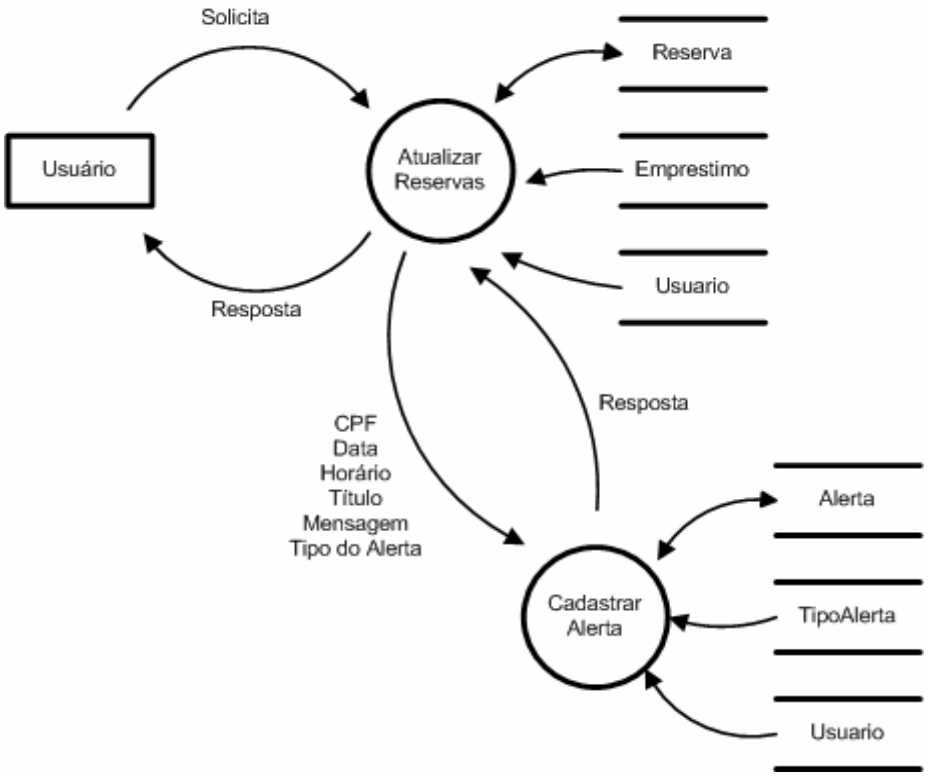

Figura 4.7 – DFD atualizar reserva

4.2.4. Diagrama de transição de estado

Abaixo ilustramos os possíveis diagramas de estados para usuários do tipo administrador e cliente. Ainda, mostramos o diagrama de estados para um usuário não autenticado no sistema.

Usuário não autenticado

No digrama exibido estão ilustrados os possíveis estados para um usuário que não esteja autenticado no sistema. São eles: Autenticar-se, recuperar senha e fazer o cadastro.

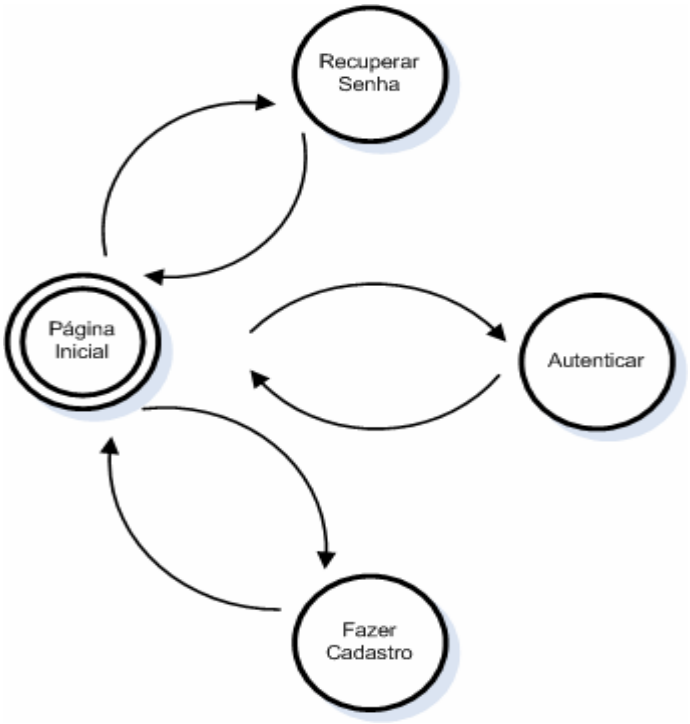

Figura 4.8 – DTE usuário não autenticado

Usuário autenticado

O diagrama abaixo ilustra os casos comuns tanto para administradores como clientes que estejam autenticados no sistema.

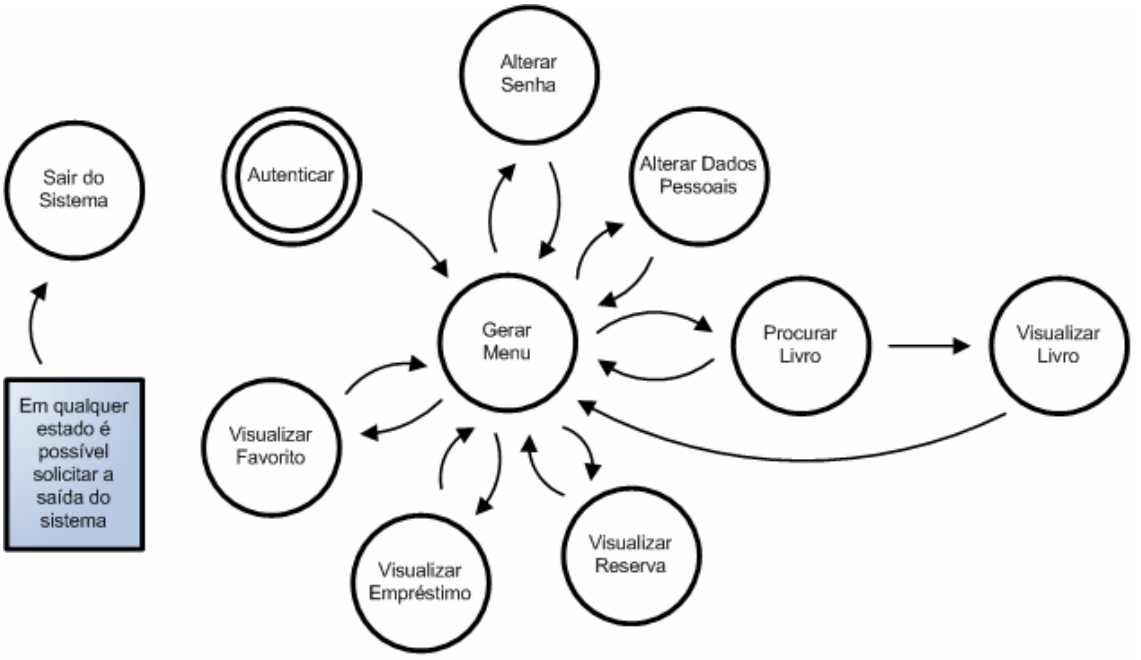

Figura 4.9 – DTE usuário autenticado

Cliente

No diagrama abaixo podemos observar os estados relativos às ações que o cliente pode ter em relação aos livros.

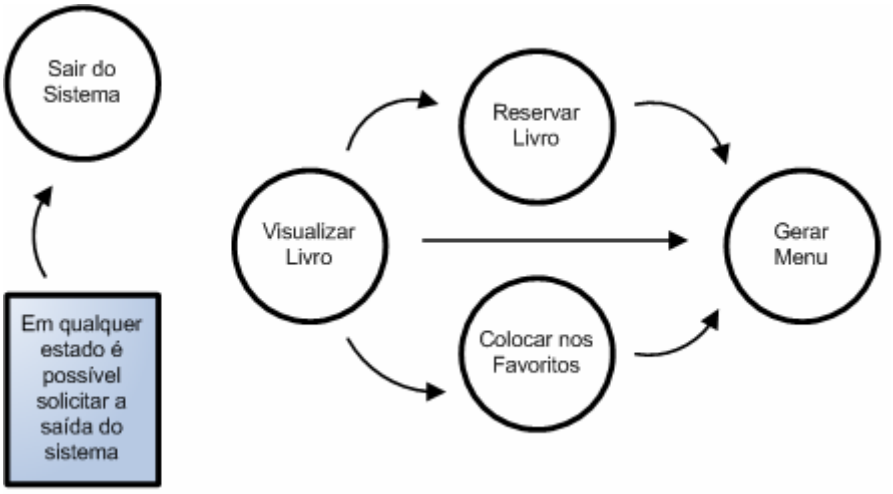

Figura 4.10 – DTE usuário cliente

Administrador

No diagrama abaixo podemos observar os estados relativos às ações que o administrador pode ter em relação aos livros.

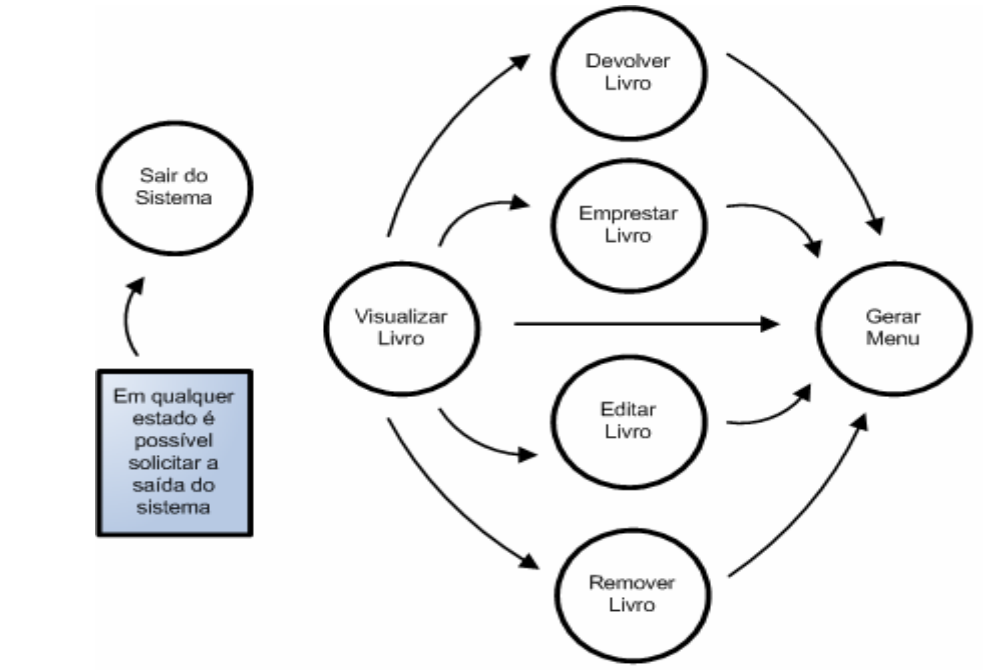

Figura 4.11 – DTE usuário administrador

No diagrama abaixo podemos observar os estados relativos às ações que o administrador pode ter além daquelas comuns ao cliente.

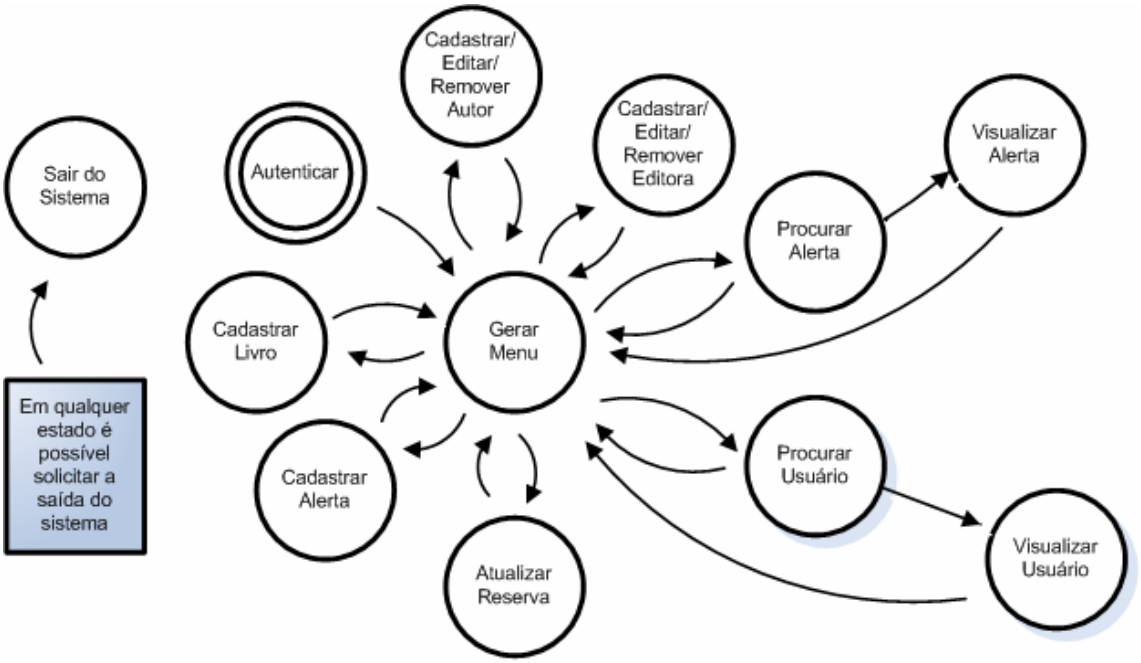

Figura 4.12 – DTE usuário administrador

No diagrama abaixo podemos observar os estados relativos às ações que o administrador pode ter em relação aos usuários.

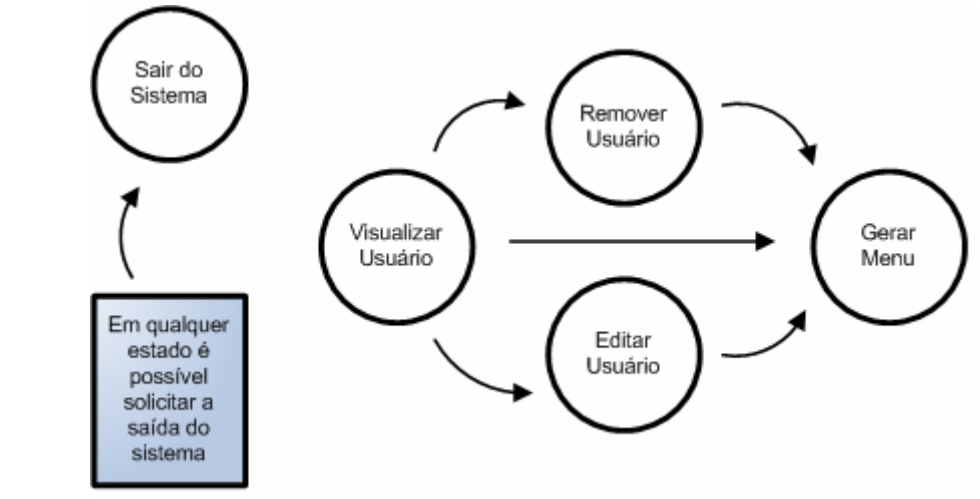

Figura 4.13 – DTE usuário administrador

E, por fim, no diagrama abaixo podemos observar os estados relativos às ações que o administrador pode ter em relação aos alertas.

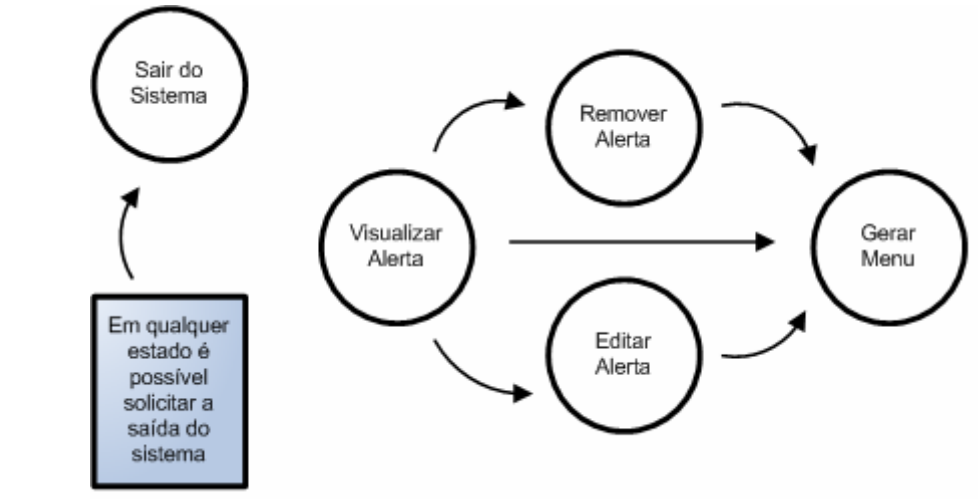

Figura 4.14 – DTE usuário administrador

4.3. Interfaces externas

A Interface com o usuário será feita através da *web*, onde tentaremos reproduzir um cenário amigável de uma biblioteca. Existirá uma página inicial de autenticação e em seguida serão exibidas páginas que facilitem a busca e reserva dos livros disponíveis.

Existirão, ainda, as interfaces via *email* e *sms*, para envio dos alertas informativos da biblioteca.

4.4. Atributos

Os seguintes atributos de qualidade podem ser conferidos ao projeto:

- Amigabilidade – Uma biblioteca *online*, para atrair clientes, precisa ter um ambiente agradável para o usuário, que o faça se sentir bem navegando. Além disso, é necessário que o ambiente seja de fácil utilização, uma vez que a grande maioria dos clientes não possui grandes conhecimentos técnicos.

- Confiabilidade – O sistema precisa ser confiável, uma vez que ele guardará informações cadastrais dos usuários. Além disso, o sistema permite que reservas sejam feitas, de forma que é necessário um bom sistema de autenticação.

- Portabilidade – Uma vez que o sistema não possui nenhum requisito que o obrigue a rodar apenas sobre Linux, ele pode ser compilado em qualquer plataforma fazendo pequenas adaptações.

- Manutenibilidade - O sistema que estamos projetando tem essa característica devido a estar sendo bem especificado, e a codificação será feita com muitos comentários, o que facilitará a correção de erros e a expansão. Essa última tem uma maior importância devido ao fato de que sempre é necessário atribuir novas funções para o melhor funcionamento de uma biblioteca via *web*.

5. Projeto

Nesse capítulo iremos mostrar a arquitetura final do sistema. Faremos uma decomposição do mesmo em blocos, depois relacionaremos os mesmos e suas dependências. Por fim, os abordaremos em maiores detalhes, fazendo referência a apêndices.

5.1. Decomposição

Aqui iremos decompor o sistema, quebrando em blocos menores. Iremos dividir em blocos que englobem funções semelhantes. Primeiro faremos essa decomposição para as funções do sistema. Após, decomporemos os dados.

5.1.1. Módulos

Subdividimos o sistema em blocos menores, porém não tão pequenos de forma a se obter uma melhor visualização do mesmo. Ilustraremos os processos de cada um deles e os dados envolvidos.

Recuperar Senha – Módulo responsável por cadastrar uma nova senha no sistema para o usuário e enviá-la através de um *email* ou *sms*.

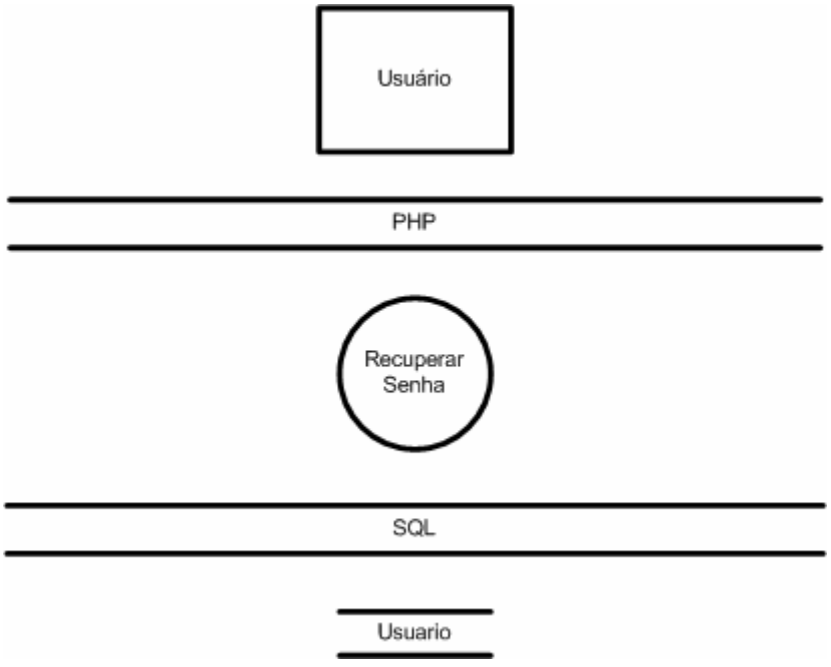

Figura 5.1 – Módulo recuperar senha

Autenticar Usuário – Módulo responsável por verificar a existência do usuário e a exatidão da senha informada.

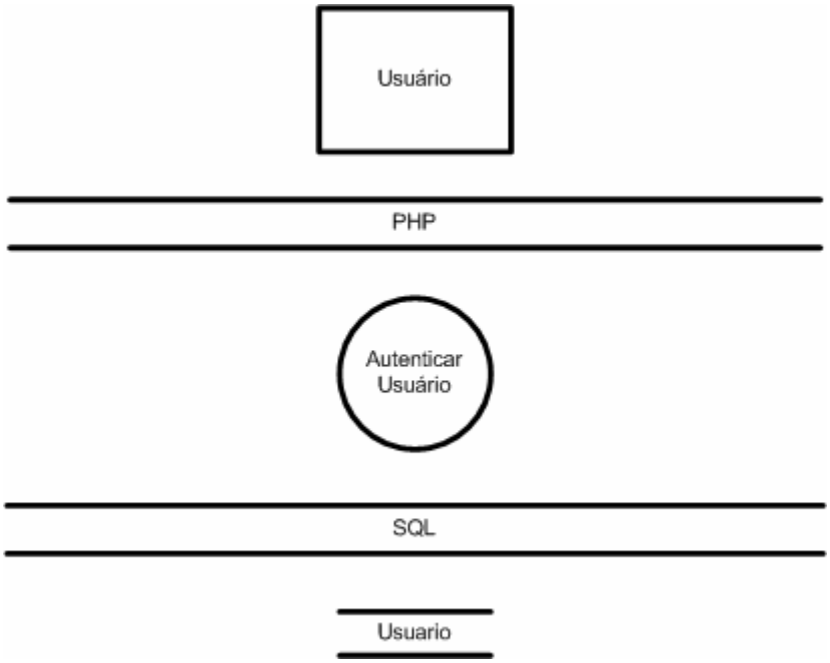

Figura 5.2 – Módulo autenticar usuário

Fazer Cadastro – Módulo responsável por cadastrar um novo usuário no sistema.

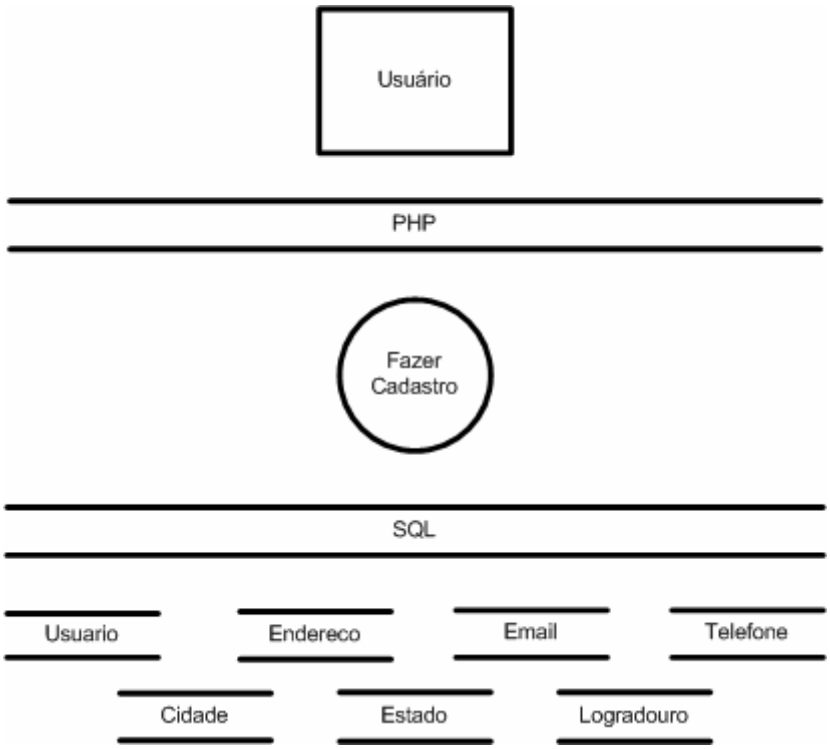

Figura 5.3 – Módulo fazer cadastro

Gerar Menu – Módulo responsável por identificar o tipo de usuário e mostrar o menu correspondente.

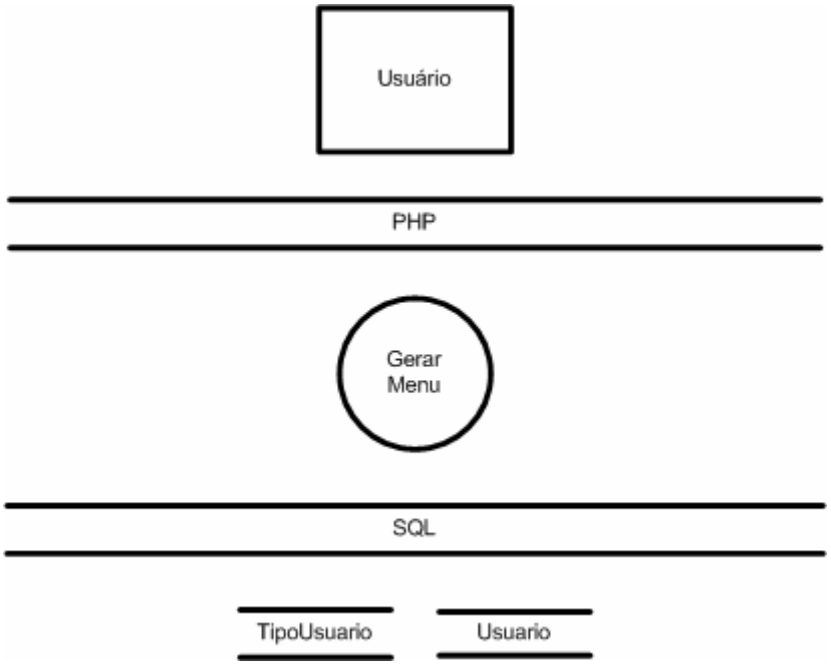

Figura 5.4 – Módulo gerar menu

Relações Cadastro – Módulo responsável por cadastrar autores, editoras, livros e alertas no sistema.

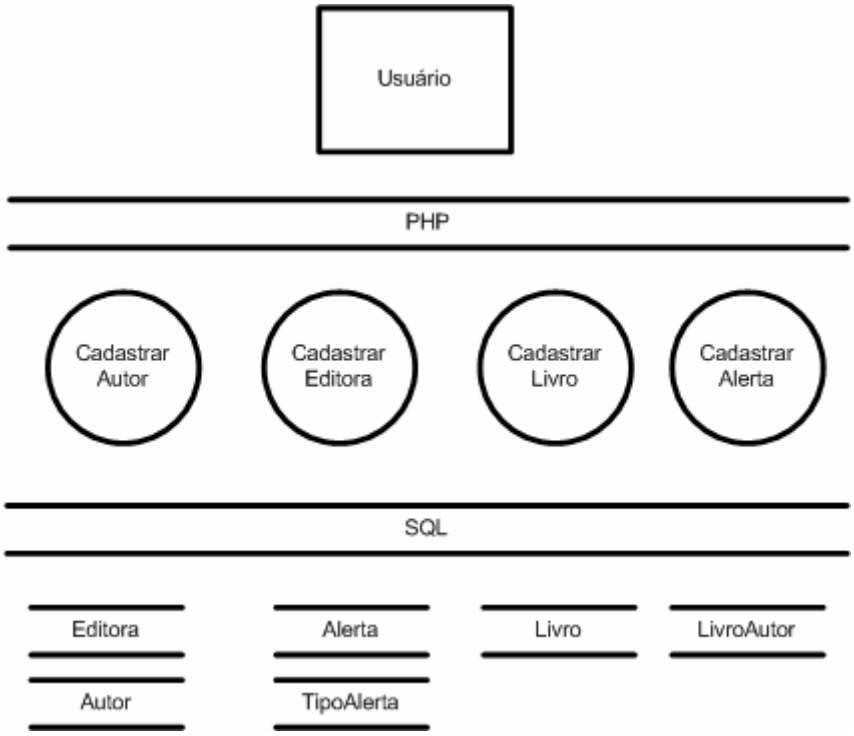

Figura 5.5 – Módulo relações de cadastro

Relações Edição – Módulo responsável por editar usuários, autores e editoras do sistema.

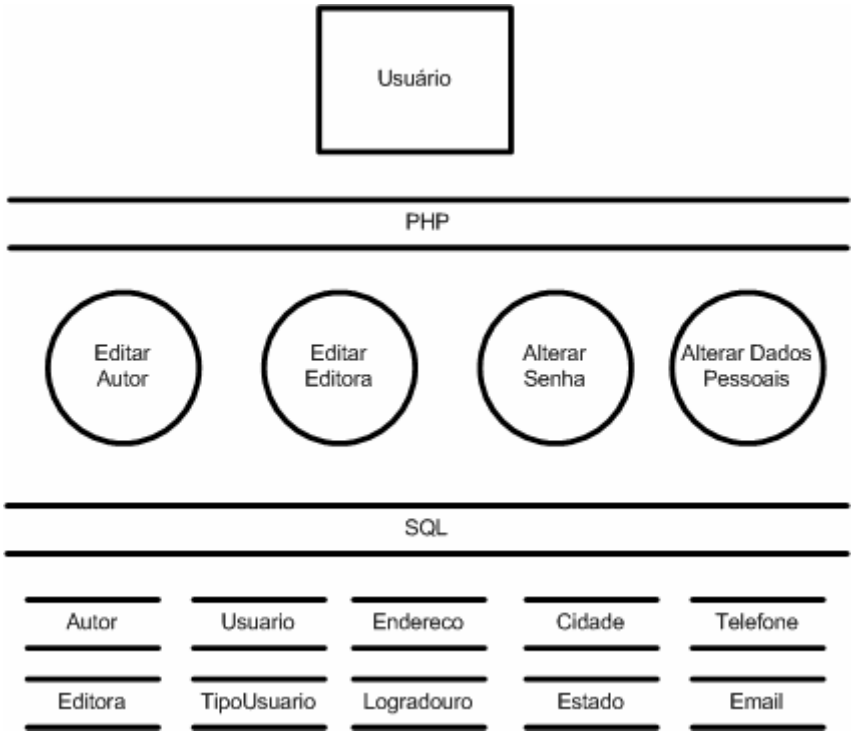

Figura 5.6 – Módulo relações de edição

Relações Remoção – Módulo responsável por remover autores e editoras do sistema.

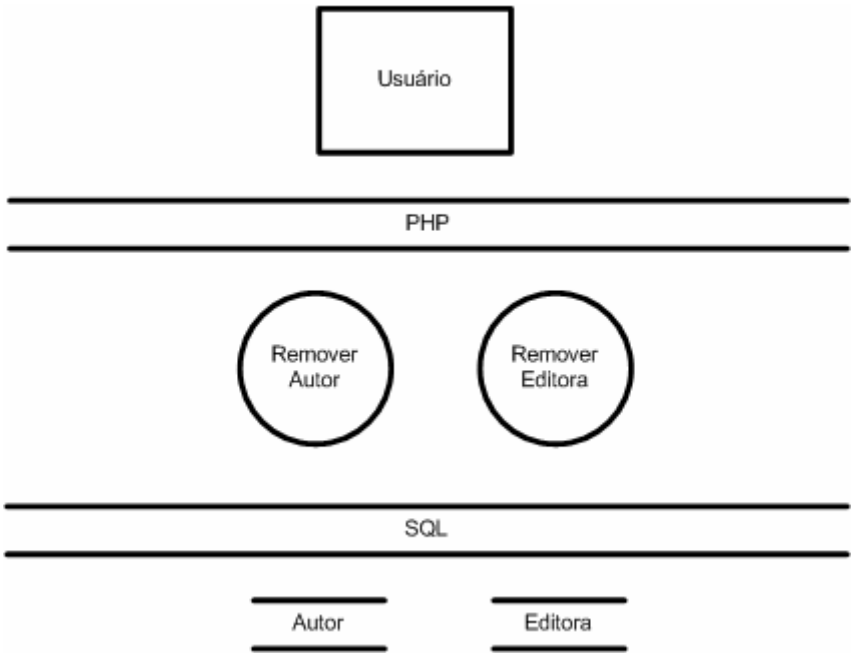

Figura 5.7 – Módulo relações de remoção

Relações Consulta – Módulo responsável por pesquisar livros, alertas e usuários do sistema.

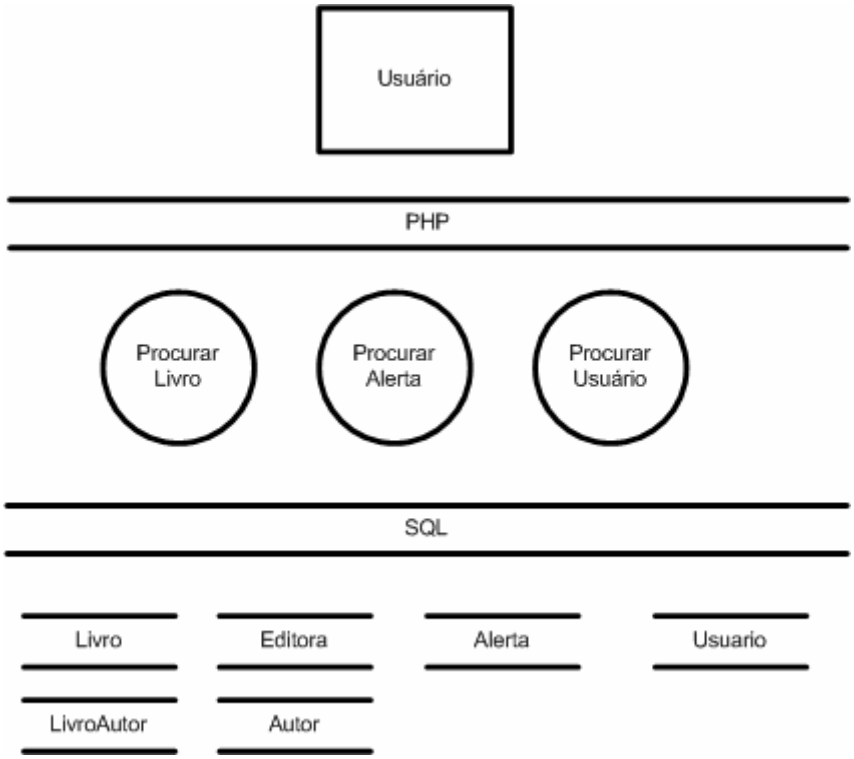

Figura 5.8 – Módulo relações de consulta

Relações Visualização – Módulo responsável pela visualização de livros, alertas e usuários.

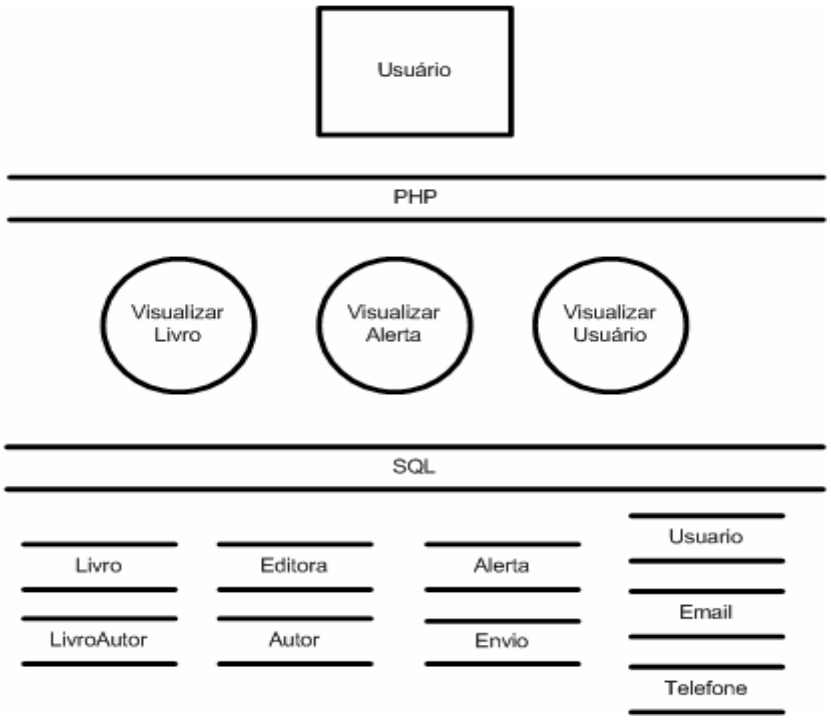

Figura 5.9 – Módulo relações de visualização

Relações Atualização – Módulo responsável por atualizar as reservas do sistema.

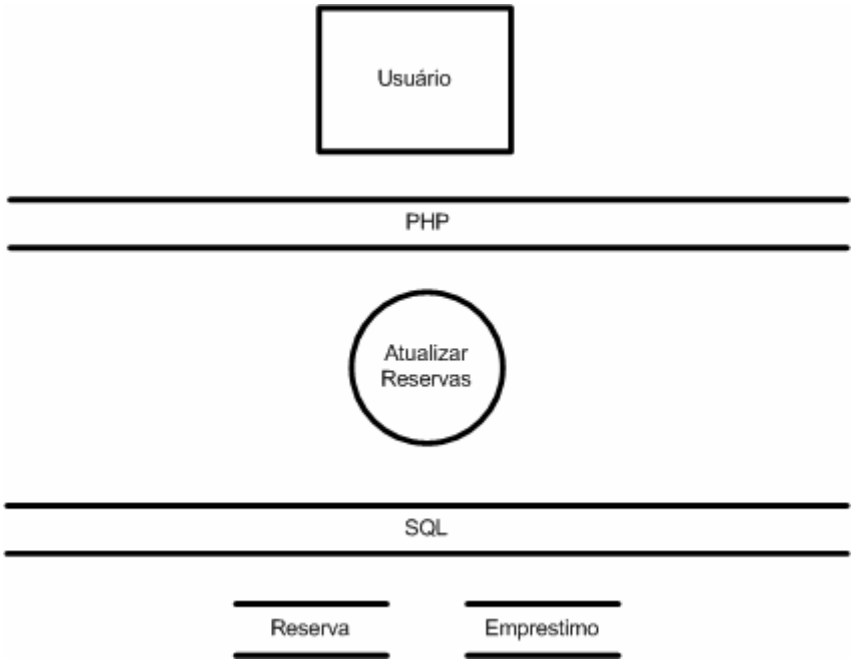

Figura 5.10 – Módulo relações de atualização

Relações Livro – Módulo responsável pelas ações que podem ser feitas sobre livros do sistema.

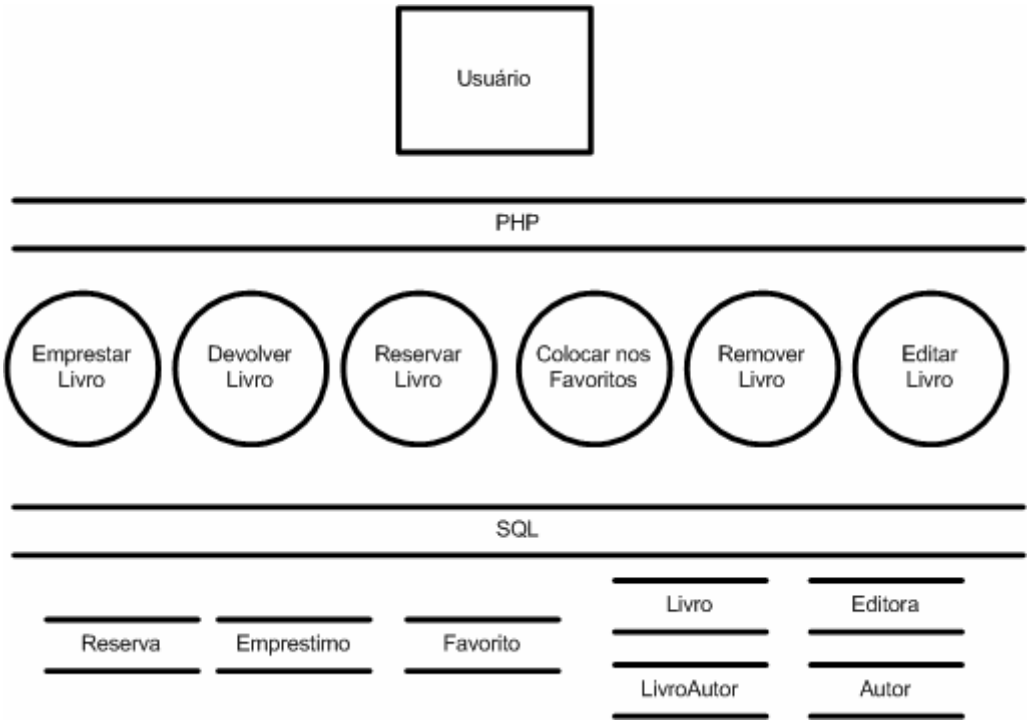

Figura 5.11 – Módulo relações de livro

Relações Usuário – Módulo responsável pela remoção de usuários.

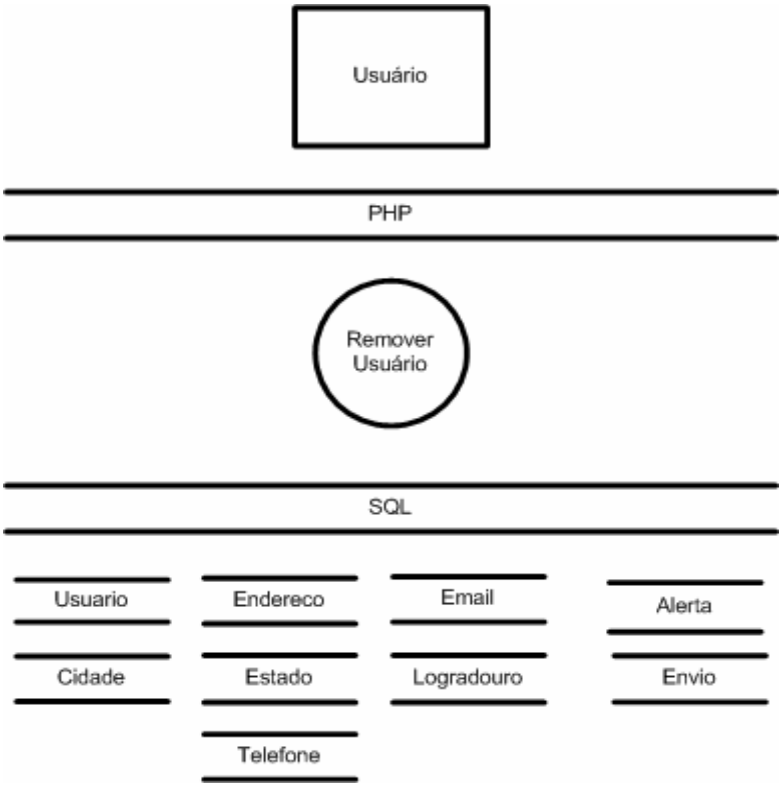

Figura 5.12 – Módulo relações de usuário

Relações Alerta – Módulo responsável pelas ações que podem ser feitas nos alertas do sistema.

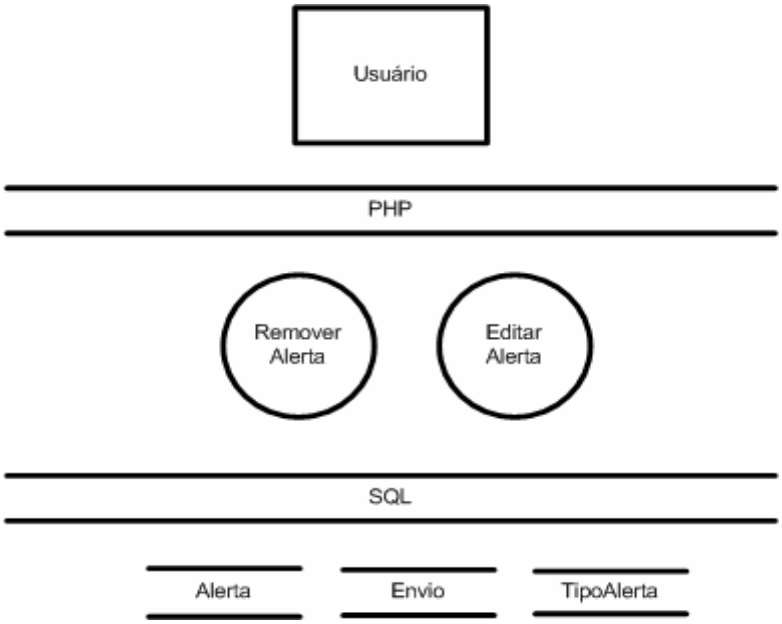

Figura 5.13 – Módulo relações de alerta

5.1.2. Dados

Abaixo listamos as entidades do nosso sistema. As chaves estrangeiras estão omitidas, mas poderão ser vistas no subcapítulo dependência entre os dados (5.2.2).

Usuario – Define os dados referentes aos usuários do sistema.

idUsuario – Código que identifica o usuário.

Nome – Nome completo do usuário.

CPF – CPF do usuário; será o login do mesmo no sistema.

Senha – Senha codificada do usuário.

TipoUsuario – Define os tipos de usuários existentes no sistema.

idTipoUsuario – Código que identifica o tipo de usuário.

Tipo – Tipo de usuário.

Telefone – Define os telefones dos usuários.

idTelefone – Código que identifica o telefone.

Telefone – Número do telefone do usuário.

TipoTelefone – Define os tipos de telefones possíveis.

idTpoTelefone – Código que identifica o tipo de telefone.

Tipo – Tipo de telefone.

Email – Define os emails dos usuários.

idEmail – Código que identifica o email.

Email – Email do usuário.

Estado – Define os Estados do Brasil.

idEstado – Código que identifica o Estado.

Estado – Sigla do Estado.

Cidade – Define as cidades do Brasil.

idCidade – Código que identifica a cidade.

Cidade – Nome da cidade.

Endereco – Define o endereço do usuário.

idEndereco – Código que identifica o endereço.

Numero – Número do endereço do usuário.

Complemento – Complemento do endereço do usuário.

Bairro – Bairro do usuário.

Logradouro – Define o logradouro do usuário

idLogradouro – Código que identifica o logradouro.

Rua – Nome da rua do usuário.

CEP – CEP correspondente ao logradouro.

Autor – Define os autores cadastrados no sistema.

idAutor – Código que identifica o autor.

Nome – Nome do autor.

Editora – Define as editoras cadastradas no sistema.

idEditora – Código que identifica a editora.

Editora – Nome da editora.

LivroAutor – Define as relações entre livros e autores do sistema.

idLivroAutor – Código que identifica a relação entre livro e autor.

Livro – Define os livros do sistema.

idLivro – Código que identifica o livro.

Titulo – Título do livro.

Ano – Ano do livro.

Edicao – Edição do livro.

Emprestimo – Define os empréstimos realizados.

idEmprestimo – Código que identifica o empréstimo.

Data_Emprestimo – Data de realização do empréstimo.

Data_Previsao - Data prevista para a devolução do livro.

Data_Devolucao – Data de realização da devolução.

Multa – Multa associada ao empréstimo.

Observacao – Alguma observação do empréstimo.

Reserva – Define as reservas realizadas.

idReserva – Código que identifica a reserva.

Data_Reserva – Data de realização da reserva.

Data_Limite - Data máxima para a realização do empréstimo associado a reserva.

Favorito – Define os favoritos cadastrados.

idFavorito – Código que identifica o favorito.

Alerta – Define os alertas agendados.

idAlerta – Código que identifica o alerta.

Titulo – Título do alerta.

Mensagem – Mensagem do alerta.

Data_Agendada – Data que o alerta deve ser enviado.

Horário_Agendado – Horário que o alerta deve ser enviado.

TipoAlerta – Define os possíveis tipo de alerta.

idTipoAlerta – Código que identifica o tipo de alerta.

Tipo – Tipo de alerta.

Envio – Define os envios dos alertas.

idEnvio – Código que identifica o envio.

Data_Envio – Data que o alerta foi enviado.

Horário_Envio – Horário que o alerta foi enviado.

Status – Resultado do envio.

5.2. Descrição das dependências

Nesse subcapítulo descrevemos, principalmente através de figuras, as dependências existentes entre os blocos do sistema, sejam eles os módulos funcionais como os dados.

5.2.1. Dependência entre módulos

A dependência entre os módulos do sistema pode ser melhor ilustrada na figura abaixo. Nela podemos observar as relações existentes entre os módulos definidos na decomposição feita no subcapítulo anterior (5.1.1).

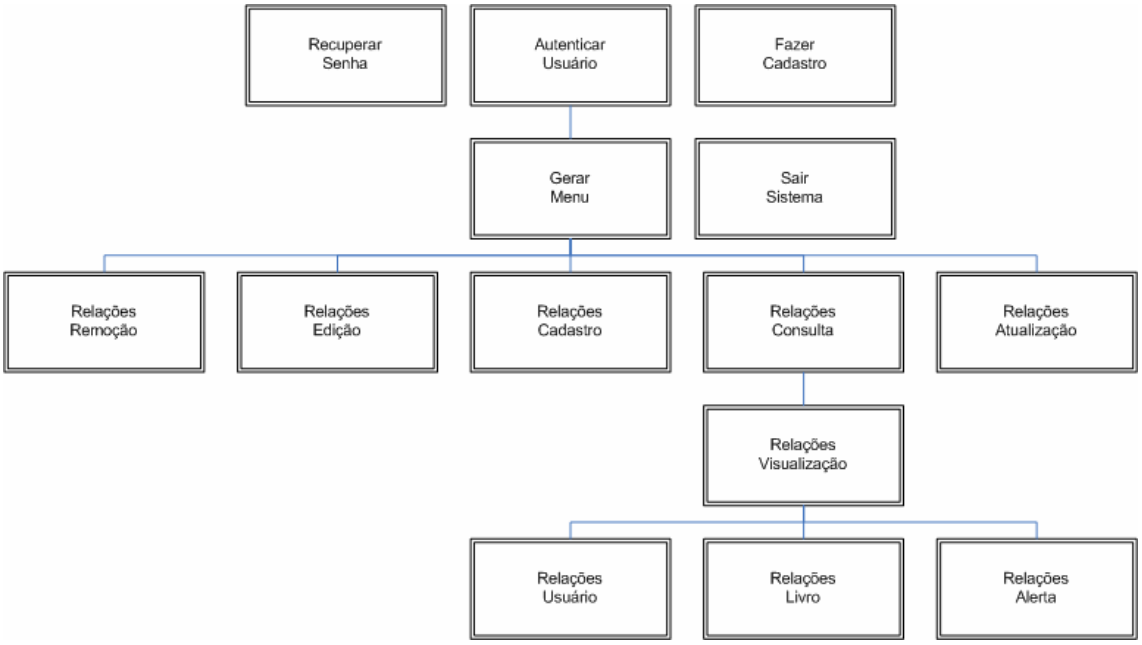

Figura 5.14 – Dependência entre módulos

Vale ressaltar que o módulo ‘Sair Sistema’ está diretamente relacionado com todos os outros módulos após ‘Autenticar Usuário’.

5.2.2. Dependência entre dados

A dependência entre os dados pode ser melhor vista através do diagrama entidade relacionamento abaixo. Nesse mesmo diagrama temos uma visão mais completa dos dados também, informando suas chaves e índices.

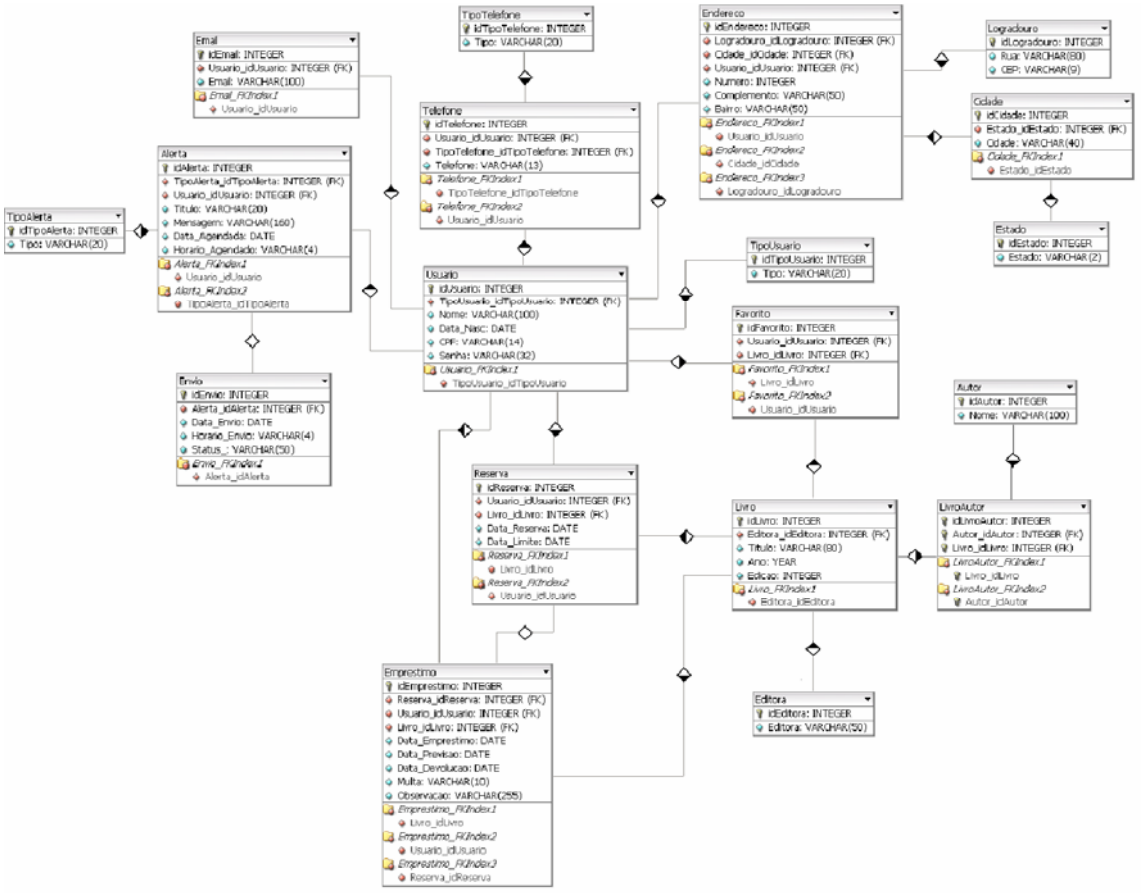

Figura 5.15 – Dependência entre dados

5.3. Arquitetura do sistema

A figura baixo ilustra a arquitetura do sistema.

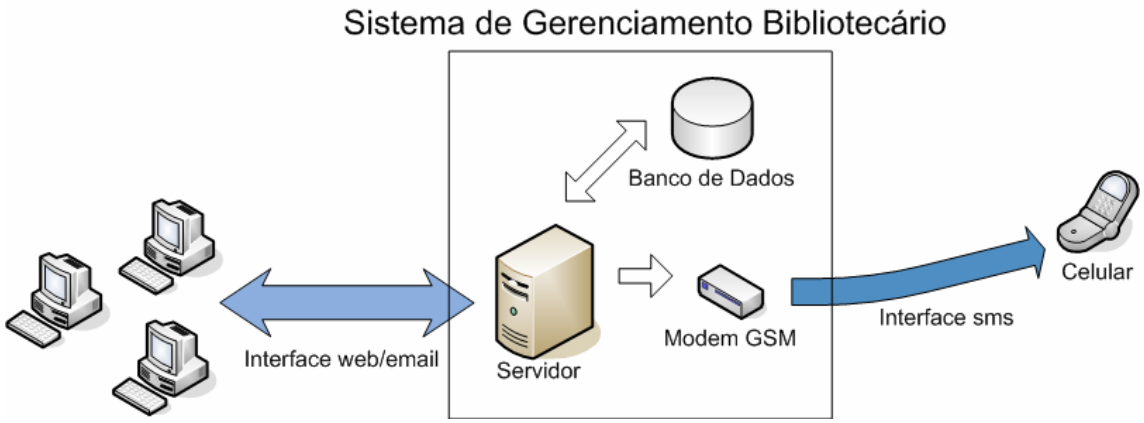

Figura 5.16 – Arquitetura

5.4. Projeto detalhado

Nesse capítulo mostraremos pseudocódigos dos módulos menores, assim como dos códigos de geração do banco de dados. No entanto, por ser muito extenso, apenas alguns deles serão abordados. Os demais podem ser encontrados no apêndice B. Além disso, o apêndice E pode ser consultado para localização dos códigos fontes originais na versão eletrônica desse documento.

5.4.1. Módulos

Abaixo temos os pseudocódigos mais importantes desse projeto.

Cadastrar alerta

```
### CÓDIGO ###

      recebe_dados();

      se ( verifica_dados() != OK )
      {
            mensagem_erro;
      }

      se ( cadastrar_alerta() != OK )
      {
            mensagem_erro();
      }

      se ( cadastra_arquivo_alerta() != OK )
      {
            mensagem_erro();
      }

      mensagem_ok();

### FIM DO CÓDIGO ###
```

Fazer autenticação

```
### CÓDIGO ###

      recebe_dados();

      se ( verifica_dados() != OK )
      {
            mensagem_erro;
      }

      se ( autenticar_usuario() != OK )
      {
            mensagem_erro();
      }

      se ( iniciar_sessao() != OK )
      {
```

```
            mensagem_erro();
      }

      redireciona_gerar_menu();

### FIM DO CÓDIGO ###
```

Reservar livro

```
### CÓDIGO ###

      recebe_dados();

      se ( verifica_dados() != OK )
      {
            mensagem_erro;
      }

      se ( livro_reservado()== NAO && livro_emprestado == NAO )
      {
            cadastra_reserva();
            cadastra_alerta();
      }
      senao
      {
            cadastra_reserva();
      }

      mensagem_ok();

### FIM DO CÓDIGO ###
```

Gerar menu

```
### CÓDIGO ###

      se ( verifica_usuario() != OK )
      {
            mensagem_erro;
      }

      case tipo_usuario:
      {
            cliente:
                  exibe_menu_cliente();
            administrador:
                  exibe_menu_administrador();
            default:
                  mensagem_erro();
      }

### FIM DO CÓDIGO ###
```

Emprestar livro

```
### CÓDIGO ###

      recebe_dados();

      se ( verifica_dados() != OK )
```

```
      {
            mensagem_erro;
      }

      se ( livro_reservado()== NAO && livro_emprestado == NAO )
      {
            cadastra_emprestimo();
            cadastra_alerta();
      }
      senao
      {
            mensagem_erro();
      }

      mensagem_ok();

### FIM DO CÓDIGO ###
```

Atualizar reservas

```
### CÓDIGO ###

      enquanto (reserva)
      {
            se ( reserva == TRAVADA )
            {
                  atualiza_reserva();
                  se ( envia_alerta() )
                  {
                        cadastra_alerta();
                  }
            }
      }

      mensagem_ok();

### FIM DO CÓDIGO ###
```

5.4.2. Dados

Nesse subcapítulo mostramos os scripts de criação do banco de dados. No apêndice E podem ser encontrados os arquivos dos scripts de inicialização de dados do sistema, que se encontram na versão eletrônica desse relatório.

```
### CÓDIGO ###

CREATE TABLE TipoUsuario (
  idTipoUsuario INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  Tipo VARCHAR(20) NOT NULL,
  PRIMARY KEY(idTipoUsuario)
);

CREATE TABLE Usuario (
  idUsuario INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  TipoUsuario_idTipoUsuario INTEGER UNSIGNED NOT NULL,
  Nome VARCHAR(100) NOT NULL,
  CPF VARCHAR(14) NOT NULL,
  Senha VARCHAR(32) NOT NULL,
```

```
  PRIMARY KEY(idUsuario),
  INDEX Usuario_FKIndex1(TipoUsuario_idTipoUsuario)
);

CREATE TABLE Email (
  idEmail INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  Usuario_idUsuario INTEGER UNSIGNED NOT NULL,
  Email VARCHAR(100) NOT NULL,
  PRIMARY KEY(idEmail),
  INDEX Email_FKIndex1(Usuario_idUsuario)
);

CREATE TABLE TipoTelefone (
  idTipoTelefone INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  Tipo VARCHAR(20) NOT NULL,
  PRIMARY KEY(idTipoTelefone)
);

CREATE TABLE Telefone (
  idTelefone INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  Usuario_idUsuario INTEGER UNSIGNED NOT NULL,
  TipoTelefone_idTipoTelefone INTEGER UNSIGNED NOT NULL,
  Telefone VARCHAR(13) NOT NULL,
  PRIMARY KEY(idTelefone),
  INDEX Telefone_FKIndex1(TipoTelefone_idTipoTelefone),
  INDEX Telefone_FKIndex2(Usuario_idUsuario)
);

CREATE TABLE Logradouro (
  idLogradouro INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  Rua VARCHAR(80) NOT NULL,
  CEP VARCHAR(9) NOT NULL,
  PRIMARY KEY(idLogradouro)
);

CREATE TABLE Estado (
  idEstado INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  Estado VARCHAR(2) NOT NULL,
  PRIMARY KEY(idEstado)
);

CREATE TABLE Cidade (
  idCidade INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  Estado_idEstado INTEGER UNSIGNED NOT NULL,
  Cidade VARCHAR(40) NOT NULL,
  PRIMARY KEY(idCidade),
  INDEX Cidade_FKIndex1(Estado_idEstado)
);

CREATE TABLE Endereco (
  idEndereco INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  Logradouro_idLogradouro INTEGER UNSIGNED NOT NULL,
  Cidade_idCidade INTEGER UNSIGNED NOT NULL,
  Usuario_idUsuario INTEGER UNSIGNED NOT NULL,
  Numero INTEGER UNSIGNED NOT NULL,
  Complemento VARCHAR(50) NULL,
  Bairro VARCHAR(50) NOT NULL,
  PRIMARY KEY(idEndereco),
  INDEX Endereco_FKIndex1(Usuario_idUsuario),
  INDEX Endereco_FKIndex2(Cidade_idCidade),
  INDEX Endereco_FKIndex3(Logradouro_idLogradouro)
```

```
);

CREATE TABLE Editora (
  idEditora INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  Editora VARCHAR(50) NOT NULL,
  PRIMARY KEY(idEditora)
);

CREATE TABLE Livro (
  idLivro INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  Editora_idEditora INTEGER UNSIGNED NOT NULL,
  Titulo VARCHAR(80) NOT NULL,
  Ano YEAR NOT NULL,
  Edicao INTEGER UNSIGNED NOT NULL,
  PRIMARY KEY(idLivro),
  INDEX Livro_FKIndex1(Editora_idEditora)
);

CREATE TABLE Autor (
  idAutor INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  Nome VARCHAR(100) NOT NULL,
  PRIMARY KEY(idAutor)
);

CREATE TABLE LivroAutor (
  idLivroAutor INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  Autor_idAutor INTEGER UNSIGNED NOT NULL,
  Livro_idLivro INTEGER UNSIGNED NOT NULL,
  PRIMARY KEY(idLivroAutor, Autor_idAutor, Livro_idLivro),
  INDEX LivroAutor_FKIndex1(Livro_idLivro),
  INDEX LivroAutor_FKIndex2(Autor_idAutor)
);

CREATE TABLE Favorito (
  idFavorito INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  Usuario_idUsuario INTEGER UNSIGNED NOT NULL,
  Livro_idLivro INTEGER UNSIGNED NOT NULL,
  PRIMARY KEY(idFavorito),
  INDEX Favorito_FKIndex1(Livro_idLivro),
  INDEX Favorito_FKIndex2(Usuario_idUsuario)
);

CREATE TABLE Reserva (
  idReserva INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  Usuario_idUsuario INTEGER UNSIGNED NOT NULL,
  Livro_idLivro INTEGER UNSIGNED NOT NULL,
  Emprestimo_idEmprestimo INTEGER UNSIGNED NULL,
  Data_Reserva DATE NOT NULL,
  Data_Limite DATE NULL,
  PRIMARY KEY(idReserva),
  INDEX Reserva_FKIndex1(Livro_idLivro),
  INDEX Reserva_FKIndex2(Usuario_idUsuario),
  INDEX Reserva_FKIndex3(Emprestimo_idEmprestimo)
);

CREATE TABLE Emprestimo (
  idEmprestimo INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  Usuario_idUsuario INTEGER UNSIGNED NOT NULL,
  Livro_idLivro INTEGER UNSIGNED NOT NULL,
  Reserva_idReserva INTEGER UNSIGNED NULL,
  Data_Emprestimo DATE NOT NULL,
```

```
  Data_Previsao DATE NOT NULL,
  Data_Devolucao DATE NULL,
  Multa VARCHAR(10) NULL,
  Observacao VARCHAR(255) NULL,
  PRIMARY KEY(idEmprestimo),
  INDEX Emprestimo_FKIndex1(Livro_idLivro),
  INDEX Emprestimo_FKIndex2(Usuario_idUsuario),
  INDEX Emprestimo_FKIndex3(Reserva_idReserva)
);

CREATE TABLE TipoAlerta (
  idTipoAlerta INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  Tipo VARCHAR(20) NOT NULL,
  PRIMARY KEY(idTipoAlerta)
);

CREATE TABLE Alerta (
  idAlerta INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  TipoAlerta_idTipoAlerta INTEGER UNSIGNED NOT NULL,
  Usuario_idUsuario INTEGER UNSIGNED NOT NULL,
  Titulo VARCHAR(20) NOT NULL,
  Mensagem VARCHAR(160) NOT NULL,
  Data_Agendada DATE NOT NULL,
  Horario_Agendado VARCHAR(4) NOT NULL,
  PRIMARY KEY(idAlerta),
  INDEX Alerta_FKIndex1(Usuario_idUsuario),
  INDEX Alerta_FKIndex2(TipoAlerta_idTipoAlerta)
);

CREATE TABLE Envio (
  idEnvio INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  Alerta_idAlerta INTEGER UNSIGNED NOT NULL,
  Data_Envio DATE NOT NULL,
  Horario_Envio VARCHAR(4) NOT NULL,
  Status VARCHAR(50) NOT NULL,
  PRIMARY KEY(idEnvio),
  INDEX Envio_FKIndex1(Alerta_idAlerta)
);

### FIM DO CÓDIGO ###
```

6. Desenvolvimento do projeto

Nesse capítulo, contamos a estória do projeto, desde o inicio até a sua conclusão, abordando os problemas encontrados e as soluções propostas.

6.1. História

O projeto iniciou-se a partir da idéia de implementar um sistema que fizesse o uso de envio de mensagens para celular. Como nos dias de hoje as aplicações para *web* estão cada vez mais na moda e como a biblioteca do departamento não possui um sistema computacional, surgiu a idéia de faz um sistema de gerenciamento bibliotecário via *web* que utilizasse o envio de mensagens de celular.

Essa aplicação começou a ser desenvolvida em uma das disciplinas do curso (Engenharia de Software) e, adicionando-se funções extras, tornou-se um projeto final, orientado pelo mesmo professor da cadeira.

A cadeira foi cursada no primeiro semestre do ano de 2007 e, nessa parte, grande parte do projeto foi analisada e planejada. No fim desse semestre começamos o desenvolvimento da aplicação, terminando a mesma em setembro do mesmo ano.

Desde então, o relatório final vem sendo confeccionado, com sua conclusão sendo feita em novembro de 2007.

Desde o inicio do projeto surgiram muitos problemas. Por isso, nos dois próximos capítulos iremos relacionar os mesmo com suas respectivas soluções.

6.2. Problemas

Nos próximos subcapítulos iremos mapear os problemas encontrados durante o desenvolvimento do projeto, para que, no próximo capítulo, as soluções dos mesmos sejam mostradas.

6.2.1. Envio dos alertas

O primeiro problema mapeado no projeto foi o envio dos alertas por *email* e *sms*. Esse era, inicialmente, o fator crítico do projeto. Por isso, foi decidido a criação de dois módulos. Uma para o envio de *emails* e outro para o envio de *sms's*.

Uma vez que esses módulos fossem criados e funcionassem corretamente, poderíamos dar seqüência no projeto.

6.2.2. Novas linguagens de programação

Depois de resolvido o problema dos alertas, novos desafios surgiram. Sendo estes o estudo de duas novas linguagens de programação (php e Shell Script).

Nenhuma dessas duas linguagens foi estudada durante o curso e teriam que ser aprendidas para o desenvolvimento do mesmo. Não existiam dúvidas de que seria possível o aprendizado. No entanto, foi uma das dificuldades, principalmente no início do desenvolvimento.

6.2.3. Porta 80 bloqueada

Outra questão relevante do projeto foi o servidor *web* que apresentava problemas no acesso de pessoas de fora da rede interna.

O servidor é instalado e pronto para funcionar na porta 80. No entanto, essa porta é bloqueada pelo provedor de internet do local onde estaria o servidor. Desse modo, fazia-se necessário uma solução. Caso contrário, o mesmo não estaria disponível para pessoal fora da rede interna.

6.2.4. Reserva de livros

Visto que, depois de solucionados os problemas expostos acima, poderíamos começar o desenvolvimento da aplicação, os próximos desafios foram todos associados às funcionalidades propriamente ditas do sistema.

Uma dessas funcionalidades foi a reserva de livros. Como várias pessoas poderiam reservar um livro e, não necessariamente, pegá-los emprestados, o sistema poderia ter casos de reservas infinitas não tratadas. Isso causaria um problema significativo. Se uma dessas reservas “travadas” não fosse “destravada”, ninguém mais conseguiria reservar o mesmo livro com sucesso.

6.2.5. Segurança do sistema

Outro ponto de interessante valor foram as questões de segurança do sistema, assim como a diferenciação entre os diversos tipos de usuário.

Como seria possível identificar qual o tipo de usuário que estaria logado e como garantir que ele não realizasse operações não permitidas? Esse foi um dos problemas onde a solução foi de significativo aprendizado, principalmente para aplicações *web*.

6.2.6. Caixa de seleção dependente

Tivemos problemas também nas questões do cadastro dos usuários. O problema estava nas caixas de opção que dependiam de outra caixa de opções. Esse foi o caso particular da seleção de Estado e Cidade pelo usuário.

O usuário selecionaria um Estado e, após isso, automaticamente seriam carregadas as opções de Cidades.

Além do problema da caixa, tínhamos que cadastrar todos os Estados e Cidades no sistema. Como, no caso das cidades, são muitas, isso também foi um dos problemas encontrados.

6.3. Soluções

Uma vez mapeados os problemas acima, iremos mostrar as soluções propostas para cada um deles nos subcapítulos subseqüentes.

6.3.1. Envio dos alertas

Para solucionar essa questão crucial do projeto, desenvolvemos dois módulos de envio de alertas. Um que enviasse *emails* e outro que enviasse *sms's*.

A questão do *email* foi resolvida através da utilização de um servidor smtp. Desenvolvemos um programa em C que se conecta no servido do *yahoo* e, desse modo, é capaz de enviar um *email*.

Esse programa usa os conceitos vistos na disciplina TCP-IP, e faz o uso da biblioteca padrão de *sockets* do Unix.

A questão das mensagens para celular foi um pouco mais complicada. Inicialmente, pensou-se na solução de integrar o sistema com uma operadora de telecomunicações. No entanto, outros problemas surgiriam. Tais como, seria necessário uma pessoa jurídica para adquirir um serviço coorporativo; a maioria das operadoras só fornece um serviço onde você consegue enviar mensagens apenas para os clientes dela, logo teriam que ser criadas conexões com mais de uma operadora; a questão de um possível custo elevado e um contrato associado; e, principalmente, a questão da VPN que teria que ser criada com os provedores, uma vez que não tínhamos disponível um IP fixo.

Desse modo, optamos pela utilização de um *modem* GSM. Essa utilização seria inviável se tivéssemos que programar a comunicação como mesmo. No entanto,

como existem bibliotecas que já cuidam dessa comunicação, o nosso problema estaria resolvido, bastando apenas estudar essa biblioteca e desenvolver em cima da mesma.

Ambas soluções podem ser vistas em maiores detalhes nos apêndices F e G, onde tratamos, respectivamente, dos módulos *sms* e *email*.

6.3.2. Novas linguagens de programação

Para melhor compreender as novas linguagens que seriam usadas, bastou a aquisição de um livro que abordava PHP com MySQL. No caso de Shell Script, foi de fundamental importância a ajuda de outras pessoas, com mais experiência no assunto.

No entanto, para o caso do Shell Script, a internet foi o meio onde encontramos as melhores soluções.

6.3.3. Porta 80 bloqueada

A solução para a questão da porta 80 bloqueada foi simples. Poderíamos optar entre configurar o servidor para escutar em outra porta, que não fosse bloqueada pelo provedor, ou então criar uma “ponte”.

Como o servidor estava em uma máquina interna da rede com IP inválido para a internet, optamos por escolher uma porta no roteador principal que não fosse bloqueada pelo provedor e fazer uma ligação com a porta 80 do servidor, dentro da rede interna.

6.3.4. Reserva de livros

Para solucionar o problema da reserva criamos uma função que atualiza as reservas cadastradas no banco de dados.

Desse modo, no momento em que a função for chamada, se houver alguma reserva “travada”, a mesma será “destravada”.

Como esse problema, normalmente está associado quando um usuário reserva um livro e não o pega emprestado, basta que a mesma função seja executada uma vez por dia. As reservas só perdem a validade na passagem de um dia para o outro. Então, idealmente, essa função deve ser chamada no início de cada dia.

6.3.5. Segurança do sistema

A questão da segurança no sistema foi resolvida utilizando-se sessões. Cada vez que um usuário se autentica no sistema é criada uma sessão, onde são armazenadas, entre outras informações, o tipo de usuário.

Dessa forma, basta que em cada página do sistema seja verificado se há uma sessão aberta e, para cada função que seja requisitada, seja verificado se o tipo de usuário daquela sessão tem autorização para realizar a mesma.

Algumas páginas do sistema só podem ser utilizadas pelo próprio sistema. Para isso, utiliza-se um bloqueio de IP. Apenas o IP do próprio servidor tem acesso a essas páginas.

6.3.6. Caixa de seleção dependente

Para solucionar o problema das caixas de seleção utilizamos um código em JavaScript. Esse código é executado sempre que a caixa de opção dos Estados é alterada e ele, automaticamente, dá um *refresh* na página passando as informações colocadas nos campos, de tal modo que na nova página apareça tudo que já estava preenchido e que possa ser criada a caixa de seleção das cidades baseado no que estava selecionado na caixa dos Estados.

Em relação à inserção de todas as cidades, criamos um programa que leu um arquivo texto contendo todas as cidades por estado e gerou a SQL para cada uma delas. Esse arquivo foi criado a partir de dados encontrados na internet.

7. Resultados

Nesse capítulo mostramos os principais resultados do sistema. Esses resultados são baseados em testes feitos durante e após a conclusão da aplicação.

Para a realização desses testes, foi criado um plano de testes que pode ser visto no apêndice C.

Nos próximos subcapítulos mostramos os testes das principais funcionalidades do sistema e o resultado obtido. Após mostrarmos esses testes, fazemos uma breve conclusão dos testes feitos na aplicação.

7.1. Autenticação

Tentou-se fazer alguns tipos de autenticação inválida antes de testar a autenticação normal.

Testamos com usuários inexistentes, senhas incorretas e, finalmente, usuário e senha correta. Em todos os casos o sistema respondeu bem, mostrando a mensagem adequada em caso de erro.

7.2. Geração dos menus

Após a autenticação deve ser mostrado um menu de acordo com o tipo de usuário. Como no sistema existem dois tipos, testamos para cada um deles e a aplicação mostrou os menus corretamente.

Além disso, forçamos um tipo usuário inexistente e a aplicação respondeu corretamente a esse erro.

7.3. Reserva de livros

Como uma das principais e mais importantes características do sistema é a possibilidade de reservar livros remotamente, testamos essa funcionalidade e aqui explicitamos.

As seguintes possibilidades para os livros no momento da reserva foram testadas: livros não emprestados e não reservados; livros emprestados e não reservados, livros não emprestados e reservados; livros emprestados e reservados.

Em todos os casos a aplicação mostrou a resposta desejada e originou os alertas devidos nas datas devidas.

7.4. Empréstimos de livros

Como associado aos empréstimos existem alertas que devem ser cadastrados, citamos o teste do mesmo.

As seguintes possibilidades para os livros no momento do empréstimo foram testadas: livros não reservados; livros reservados.

Em todos os casos a aplicação mostrou a resposta desejada e originou os alertas devidos nas datas devidas.

7.5. Atualização de reservas

Um outro ponto crítico do sistema é a atualização de reservas, uma vez que se não bem estruturadas, poderia gerar casos de reservas infinitas.

A possibilidade de ocorrer essa reserva infinita seria a seguinte: usuário reserva um livro e nunca o pega emprestado. Caso não existisse a atualização das reservas e existisse um outro usuário com reserva para o mesmo livro feita posteriormente, o mesmo nunca seria alertado de que o livro está disponível para ele.

A aplicação foi testada para esse caso e funcionou corretamente.

7.6. Agendamento de alertas

Como a grande inovação desse sistema e a maior motivação para o desenvolvimento do mesmo foram a possibilidade de envio de alertas para os usuários, citamos aqui os testes dessa parte.

Os alertas podem ser gerados automaticamente pelo sistema no caso de outras funções serem realizadas. Como exemplo: usuário pega livro emprestado (será gerado um alerta para a data de devolução prevista). No entanto, o módulo para esse agendamento pode ser usado isoladamente e foi isso que testamos para garantir que os agendamentos funcionam corretamente.

Geramos alertas para diversas datas e o sistema respondeu corretamente e os enviou do jeito e nas datas desejadas.

7.7. Resultado dos testes

Durante os testes foram encontrados muitos problemas e, por isso, essa fase foi fundamental para o correto desenvolvimento do sistema.

Além disso, muitos casos não mapeados na análise do projeto foram descobertos durante a fase de testes e, dessa forma, possibilitou uma re-análise e um

aprimoramento da aplicação. Do mesmo modo, foram mapeados pontos de melhoria e aspectos que devem ser tratados em uma nova versão. Isso será tratado no próximo capítulo.

8. Conclusão

Nesse capítulo concluímos o relatório do projeto final. Nele iremos apresentar uma avaliação crítica da tecnologia empregada, da metodologia utilizada, assim como das ferramentas.

Depois dessa avaliação, iremos discutir o que foi aprendido e fazer propostas futuras em relação ao sistema desenvolvido nesse projeto.

8.1. Avaliação da tecnologia

As duas principais tecnologias usadas no projeto foram o *sms* e o *email*. Para que pudéssemos utilizá-las foi necessário um estudo a parte e o desenvolvimento de dois módulos. Desse modo, tivemos a oportunidade de realizar dois pequenos projetos em separado.

Em relação à parte do *sms*, o sistema o utilizou da forma mais adequada aos nossos recursos. A solução usando um *modem* foi interessante, diferente da idéia original e proporcionou um ganho muito grande de conhecimento na área.

Além disso, podemos dizer que o seu papel foi plenamente cumprido no que diz respeito ao meio de comunicação com o usuário.

Em relação à parte do *email*, o sistema o utilizou de uma forma alternativa ao padrão de envio de *emails*. Normalmente usa-se um servidor de *email* no próprio sistema. Em nosso caso, fizemos proveito de um servidor já existente (*yahoo*), que nos proporcionou ganho de conhecimento na interação com outros servidores, principalmente no que diz respeito ao protocolo de comunicação smtp.

Com essa solução conseguimos usar um remetente “mais confiável”, visto que o *yahoo* é mundialmente reconhecido. Nos testes preliminares usando nosso próprio servidor para o envio das mensagens notamos que elas sempre caiam nas caixas de *spam* dos destinatários.

Mesmo assim, quando enviamos para alguns provedores (*hotmail*, por exemplo) as mensagens continuavam sendo tratadas como *spam*. O ideal seria que os *emails* sempre chegassem aos destinos sem nenhum tipo de bloqueio. Talvez isso seja inviável nos dias de hoje, com tantos *anti-spams*.

8.2. Avaliação da metodologia

Como o sistema trata-se de um software, iremos avaliar a metodologia de projeto usada para o desenvolvimento do mesmo, que foi a estruturada.

Por se tratar de uma metodologia mais facilmente entendida e capaz de atender os objetivos desse sistema, ela foi adequada e proporcionou que desenvolvêssemos o projeto de forma mais objetiva e organizada.

Através da mesma conseguimos prever os pontos mais críticos do projeto, assim como planejar as melhores soluções. Conseguimos também identificar alguns problemas que puderam ser novamente analisados e corrigidos, como já tratados no capítulo anterior.

Dentre as fases incluídas nessa metodologia, citamos a modelagem relacional e a listagem de eventos. Essas duas nos possibilitaram visualizar melhor o sistema e desenvolvê-lo da maneira mais adequada.

8.3. Avaliação das ferramentas

Nesse projeto, utilizamos diversas ferramentas. Muitas delas, sem nenhuma experiência de utilização anterior. Por esse motivo, algumas não foram, inicialmente, utilizadas da melhor maneira. Com a prática, durante o próprio desenvolvimento, foram sendo melhor usadas.

Todas essas ferramentas atenderam plenamente aos objetivos do sistema, destacando-se o banco de dados MySQL, a linguagem de programação PHP e o servidor Apache, que são os pilares do sistema.

8.4. Aprendizado

Como muitas das ferramentas e dos conceitos necessários ao desenvolvimento do projeto nunca tinham sido estudados antes, podemos dizer que o projeto proporcionou o aprendizado de novas tecnologias e ferramentas. Citamos como as principais, o envio de *sms*, a utilização de um servidor *web* e a programação em PHP.

Além dessas, alguns métodos, conceitos e ferramentas já haviam sido usadas em outras oportunidades e, com o projeto, tivemos a oportunidade de fortalecer os conhecimentos sobre os mesmos. Dentre os quais destacamos a metodologia estruturada de engenharia de software; a programação, propriamente dita, assim como a mesma voltada para redes de comunicações; e, por fim, a modelagem de banco de dados.

8.5. Propostas futuras

Após a conclusão desse projeto, ficam algumas sugestões de melhoria para o sistema em versões futuras.

A primeira delas é em relação ao envio dos *emails*. Fica proposto um estudo sobre como evitar que algumas mensagens sejam consideradas *spam* por determinados provedores. No próprio *email*, fica proposto o uso de anexos.

Um outro ponto de melhoria seria a automatização das atualizações de reserva, visto sua importância fundamental para a consistência do sistema. Uma outra sugestão seria limitar o número de reservas e empréstimos possíveis a um usuário no mesmo período, visto que a maioria das bibliotecas tem essa política.

Uma outra sugestão seria a utilização de figuras na descrição dos livros. A maioria da livrarias virtuais utiliza imagens e isso facilita muito a identificação de livros.

Por fim, a última sugestão, seria no campo da segurança. A idéia seria a implementação de um mecanismo capaz de identificar múltiplas tentativas de autenticação em um curto intervalo de tempo. Se isso fosse identificado, a autenticação faria o uso de imagens aleatórias para garantir que não está sendo feito nenhum tipo de invasão ou tentativa de descobrir senhas de usuários.

Bibliografia

- Muto, C.A., “PHP & MySQL – Guia Introdutório”, Brasport, 2006.
- Pressman, R.S., “Engenharia de Software”, McGraw-Hill, 2006.
- Schildt, H., “C - Completo e Total”, Pearson Education do Brasil, 1996.
- http://br.php.net/, acessado pela última vez em 19/10/2007.
- http://www.devin.com.br/eitch/shell_script/, acessado pela última vez em 19/10/2007.
- http://www.pxh.de/fs/gsmlib/, acessado pela última vez em 19/10/2007.

Apêndice A – Especificação de requisitos de software

Fazer cadastro no sistema

O processo fazer cadastro recebe os campos necessários preenchidos pelo usuário e verifica se estão todos corretos. Caso esteja, realiza o cadastro no sistema.

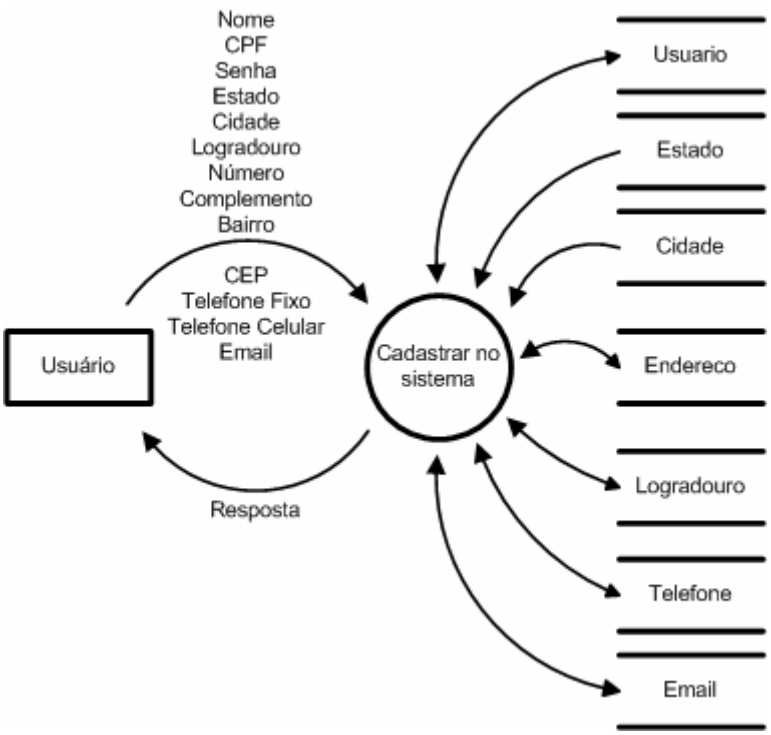

Figura A.1 – DFD fazer cadastro no sistema

Recuperar senha

O processo recuperar senha recebe os campos necessários preenchidos pelo usuário e verifica se estão todos corretos. Além disso, verifica se o usuário está cadastrado no sistema. Caso esteja, altera a senha do mesmo e a envia.

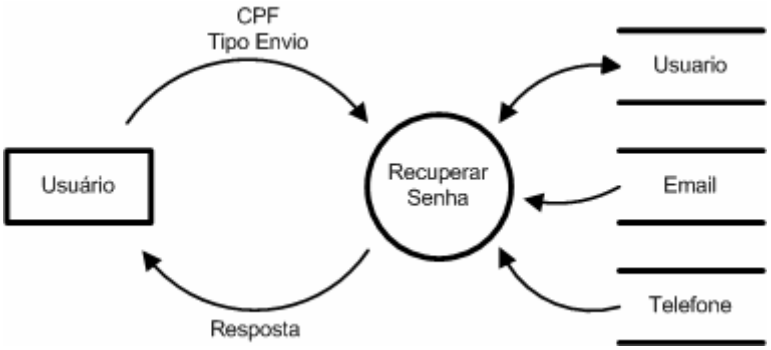

Figura A.2 – DFD recuperar senha

Sair do sistema

O processo sair do sistema termina a sessão do usuário.

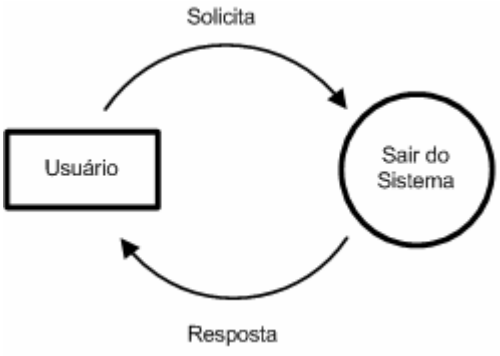

Figura A.3 – DFD sair do sistema

Alterar senha

O processo alterar senha recebe os campos necessários preenchidos pelo usuário e verifica se estão todos corretos. Caso esteja, realiza a alteração.

Figura A.4 – DFD alterar senha

Cadastrar autor

O processo cadastrar autor recebe os campos necessários preenchidos pelo usuário e verifica se estão todos corretos. Caso esteja, realiza o cadastramento no sistema.

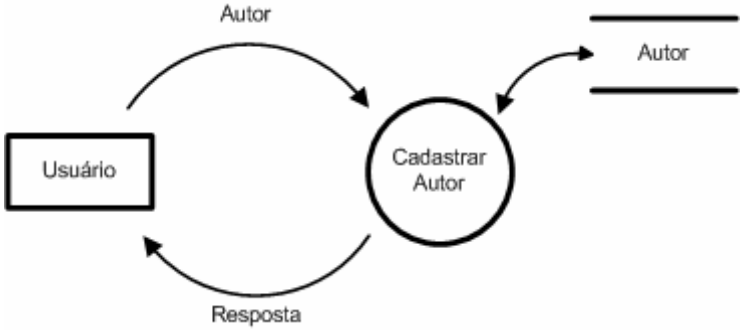

Figura A.5 – DFD cadastrar autor

Cadastrar editora

O processo cadastrar editora recebe os campos necessários preenchidos pelo usuário, verifica se estão todos corretos. Caso esteja, realiza o cadastramento.

Figura A.6 – DFD cadastrar editora

Alterar dados pessoais

O processo alterar dados recebe os campos necessários preenchidos pelo usuário e verifica se estão todos corretos. Caso esteja, realiza a alteração.

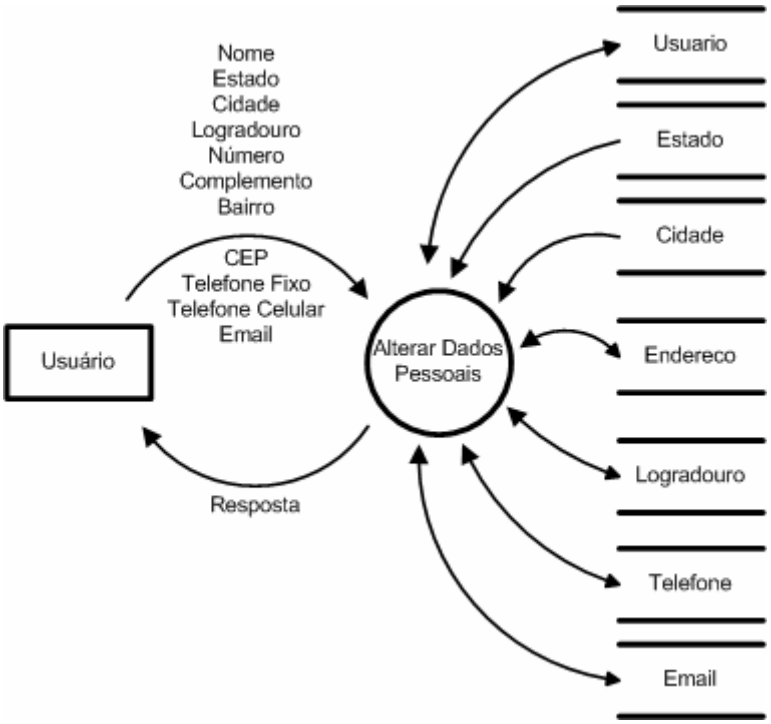

Figura A.7 – DFD alterar dados

Editar autor

O processo editar livro recebe os campos necessários preenchidos pelo usuário, verifica se estão todos corretos. Caso esteja, edita-o no sistema.

Figura A.8 – DFD editar autor

Remover autor

O processo remover autor recebe os campos necessários preenchidos pelo usuário e verifica se estão todos corretos. Além disso, verifica se o autor possui algum livro cadastrado. Caso não possua, realiza a remoção.

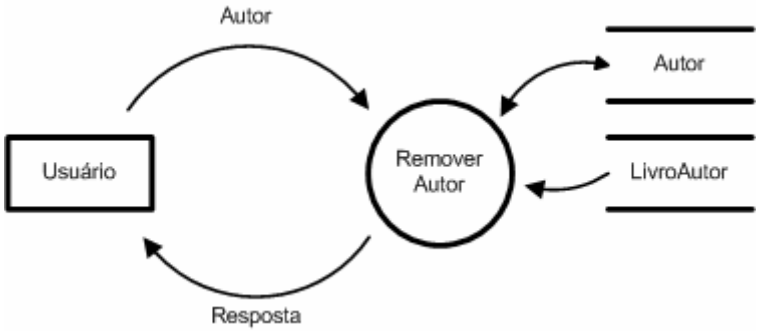

Figura A.9 – DFD remover autor

Editar editora

O processo editar editora recebe os campos necessários preenchidos pelo usuário e verifica se estão todos corretos. Caso esteja, realiza a edição.

Figura A.10 – DFD editar editora

Remover editora

O processo remover editora recebe os campos necessários preenchidos pelo usuário, verifica se estão todos corretos. Além disso, verifica se a editora possui algum livro cadastrado. Caso não possua, remove-a do sistema.

Figura A.11 – DFD remover editora

Cadastrar livro

O processo cadastrar livro recebe os campos necessários preenchidos pelo usuário, verifica se estão todos corretos. Caso esteja, cadastra-o no sistema.

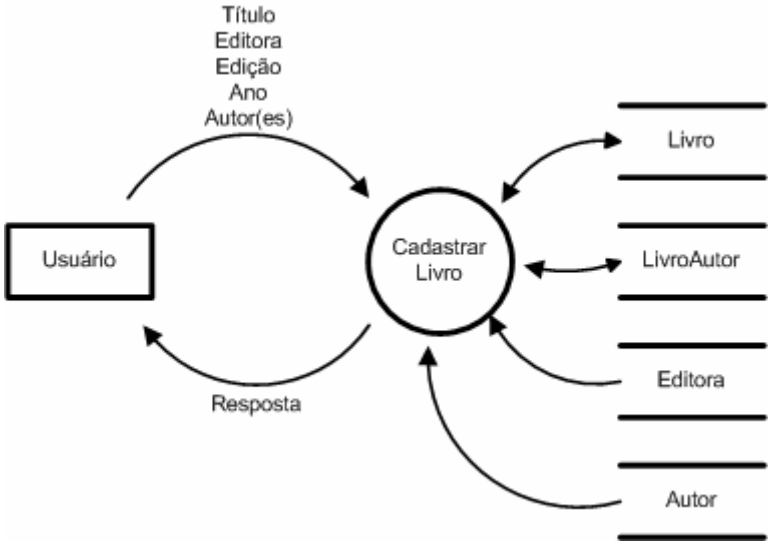

Figura A.12 – DFD cadastrar livro

Editar livro

O processo editar livro recebe os campos necessários preenchidos pelo usuário, verifica se estão todos corretos. Caso esteja, edita-o no sistema.

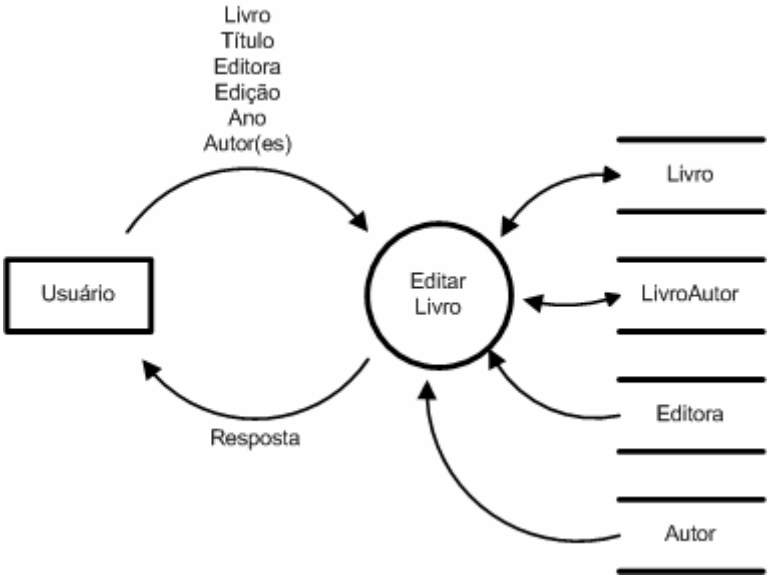

Figura A.13 – DFD editar livro

Remover livro

O processo remover livro recebe os campos necessários preenchidos pelo usuário, verifica se estão todos corretos. Além disso, verifica se o livro possui algum empréstimo ou reserva ou se é favorito de algum usuário. Caso sim, além de removê-lo do sistema, remove os demais registros associados também.

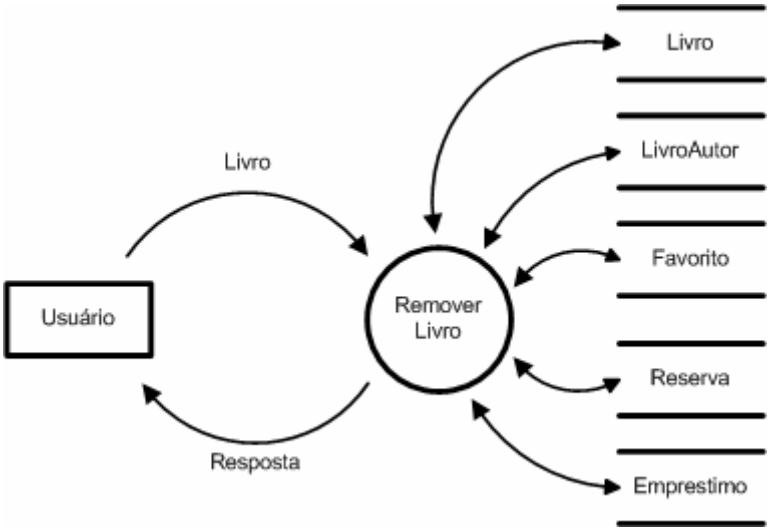

Figura A.14 – DFD remover livro

Editar alerta

O processo editar alerta recebe os campos necessários preenchidos pelo usuário, verifica se estão todos corretos. Caso esteja, edita-o no sistema.

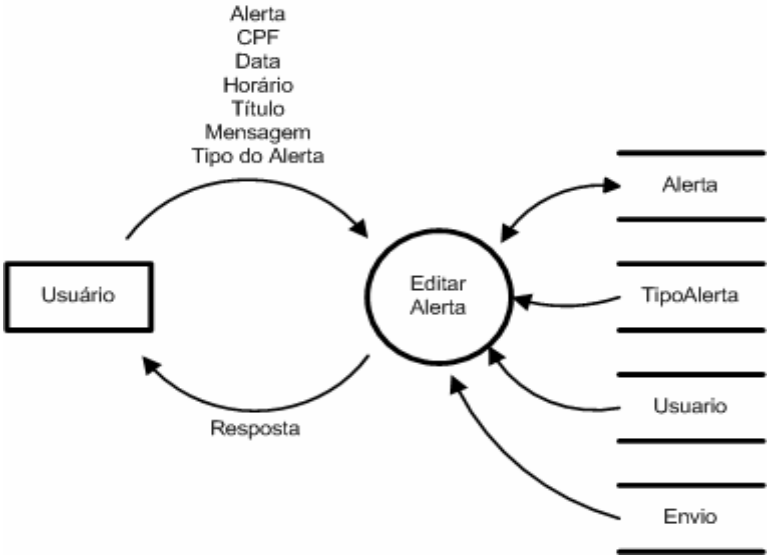

Figura A.15 – DFD editar alerta

Remover alerta

O processo remover alerta recebe os campos necessários preenchidos pelo usuário, verifica se estão todos corretos. Além disso, verifica se o alerta já foi enviado. Caso não, remove-o do sistema.

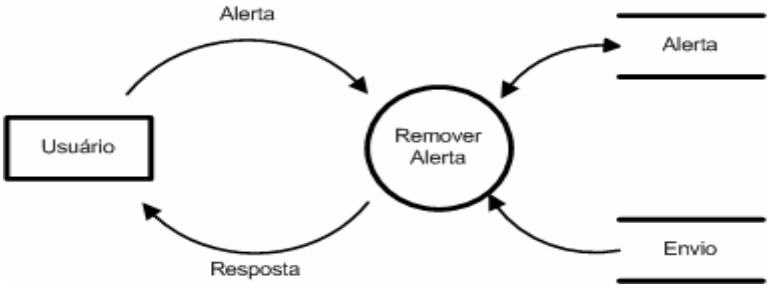

Figura A.16 – DFD remover alerta

Enviar alerta

O processo enviar alerta monitora os alertas agendados. Quando o horário programado para o alerta chegar, o mesmo é colocado na fila de envios a serem feitos e será enviado pelo sistema. Após o seu envio, o status do mesmo é cadastrado no sistema.

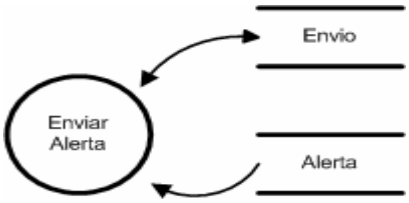

Figura A.17 – DFD enviar alerta

Procurar livro

O processo procurar livro recebe os campos necessários preenchidos pelo usuário, verifica se estão todos corretos. Caso sim, faz um consulta com os parâmetros informados e exibe o resultado.

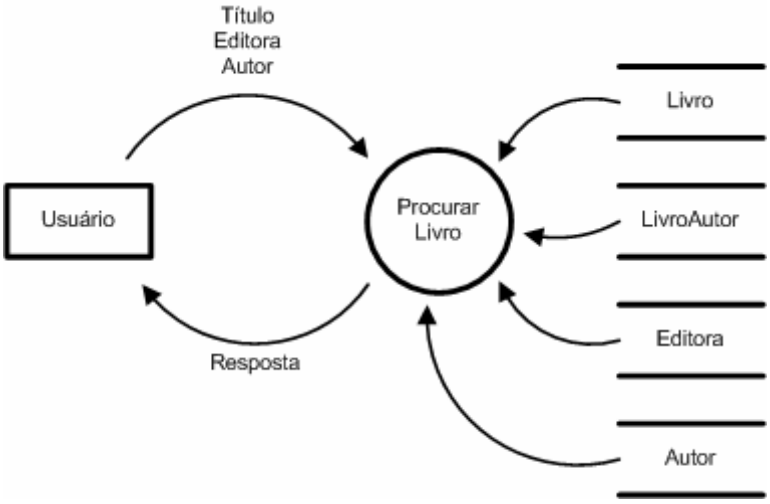

Figura A.18 – DFD procurar livro

Procurar usuário

O processo procurar usuário recebe os campos necessários preenchidos pelo usuário, verifica se estão todos corretos. Caso sim, faz um consulta com os parâmetros informados e exibe o resultado.

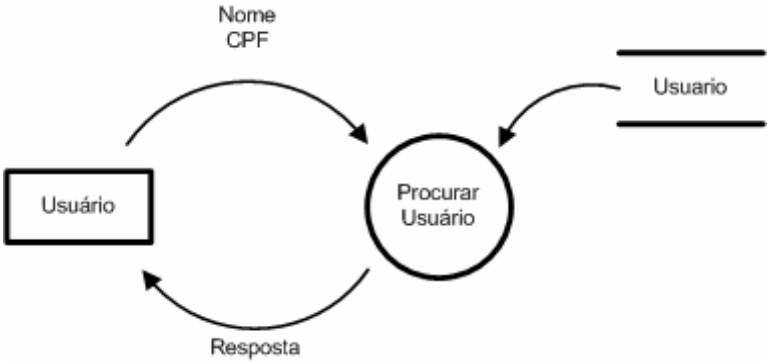

Figura A.19 – DFD procurar usuário

Procurar alerta

O processo procurar alerta recebe os campos necessários preenchidos pelo usuário, verifica se estão todos corretos. Caso sim, faz um consulta com os parâmetros informados e exibe o resultado.

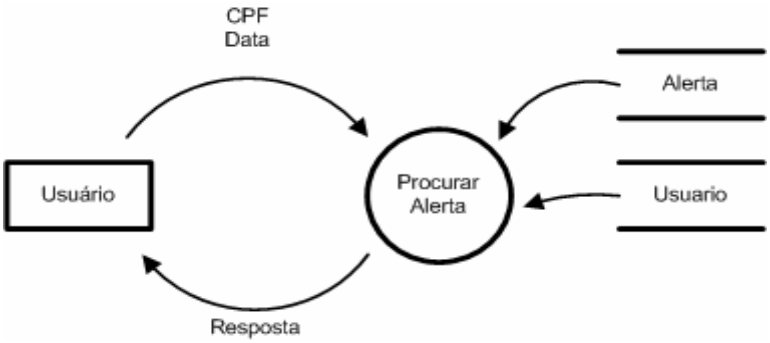

Figura A.20 – DFD procurar alerta

Visualizar usuário

O processo visualizar usuário recebe o usuário e faz um consulta com para pegar as informações do mesmo. Além disso, exibe as opções possíveis a serem realizadas com o mesmo.

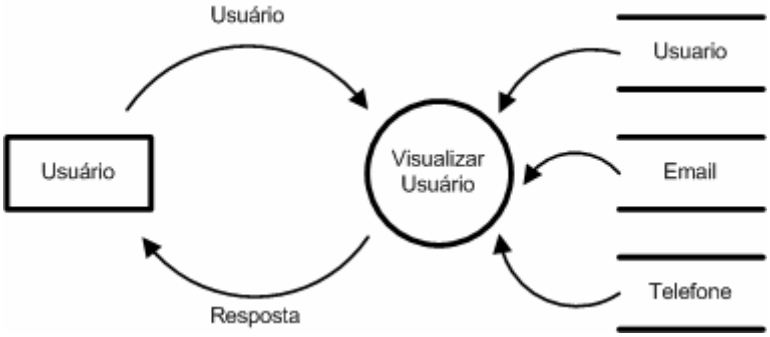

Figura A.21 – DFD visualizar usuário

Visualizar alerta

O processo visualizar alerta recebe o alerta e faz um consulta com para pegar as informações do mesmo. Além disso, baseado nas informações do alerta, exibe as opções possíveis a serem realizadas com o mesmo.

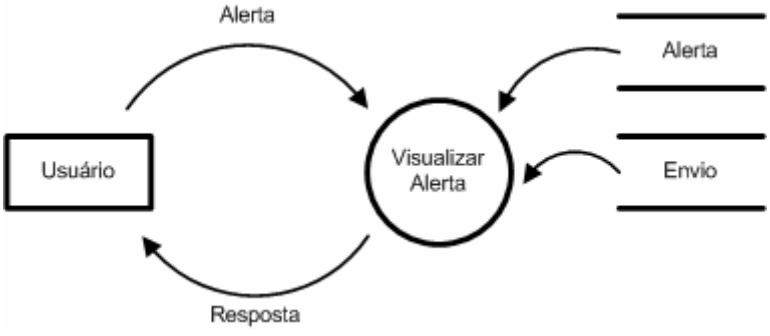

Figura A.22 – DFD visualizar alerta

Remover usuário

O processo remover usuário recebe os campos necessários preenchidos pelo usuário, verifica se estão todos corretos. Além disso, verifica se existem alertas para o mesmo. Caso sim, remove o próprio usuário e os alertas associados.

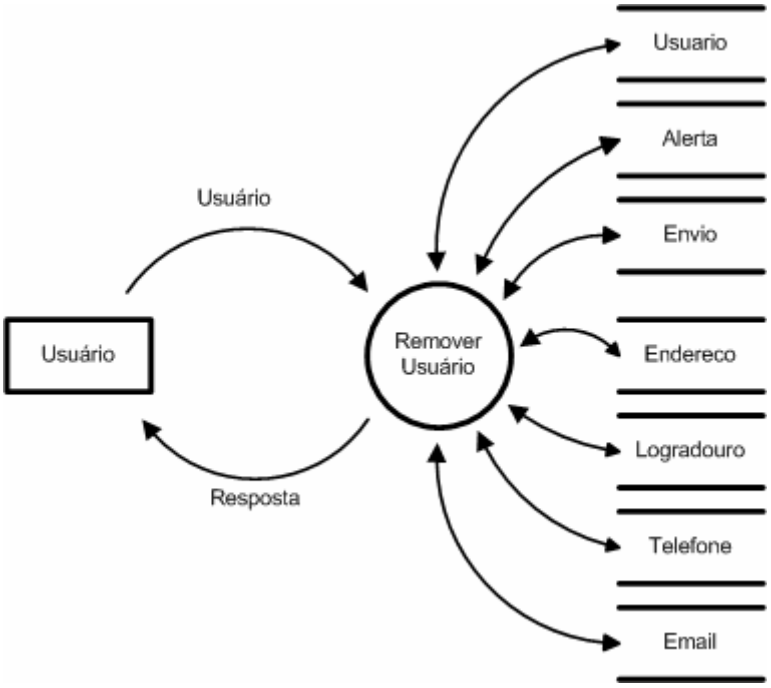

Figura A.23 – DFD remover usuário

Colocar livro nos favoritos

O processo colocar livro nos favoritos recebe o livro, verifica sua existência e, caso já não exista esse favorito para o usuário, cria um registro no banco.

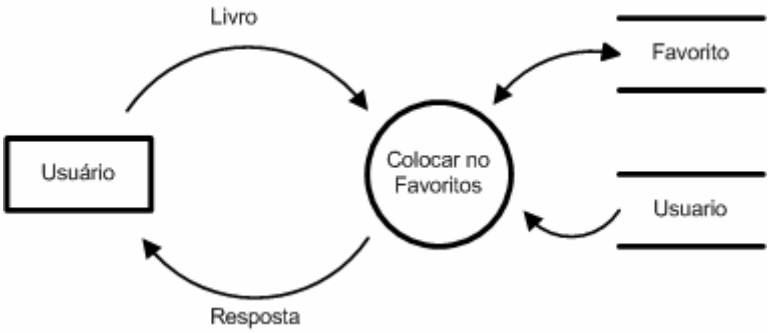

Figura A.24 – DFD colocar nos favoritos

Devolver livro

O processo devolver livro recebe o livro e faz um consulta para descobrir se deverá existir uma multa associada. Coleta as informações necessárias e cadastra a devolução no banco.

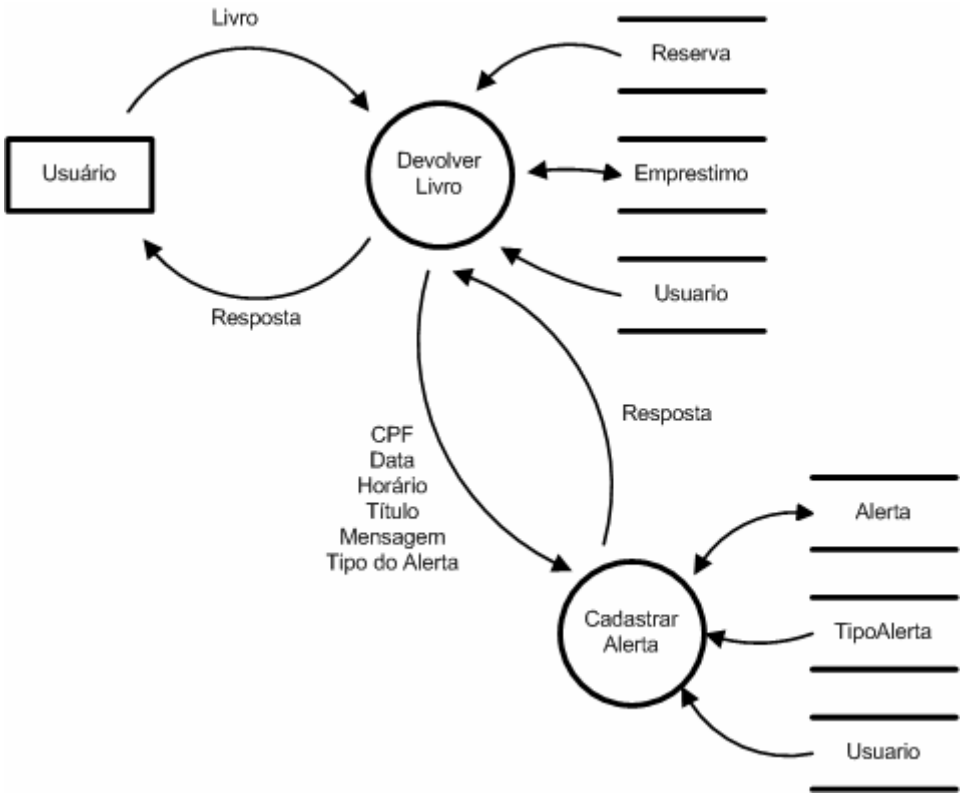

Figura A.25 – DFD devolver livro

Visualizar livro

O processo visualizar livro recebe o livro e faz um consulta para pegar as informações do mesmo. Além disso, baseado no usuário que solicitou e nas informações do livro, exibe as opções possíveis a serem realizadas.

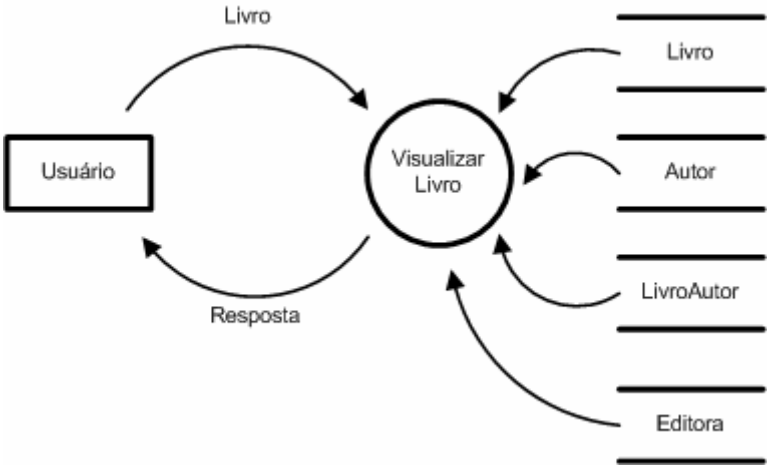

Figura A.26 – DFD visualizar alerta

Visualizar reserva, empréstimo ou favorito

O processo visualizar reserva, empréstimo ou favorito recebe o usuário e faz um consulta baseada no usuário em questão e mostra as informações dos livros que estão reservados, emprestados ou nos favoritos.

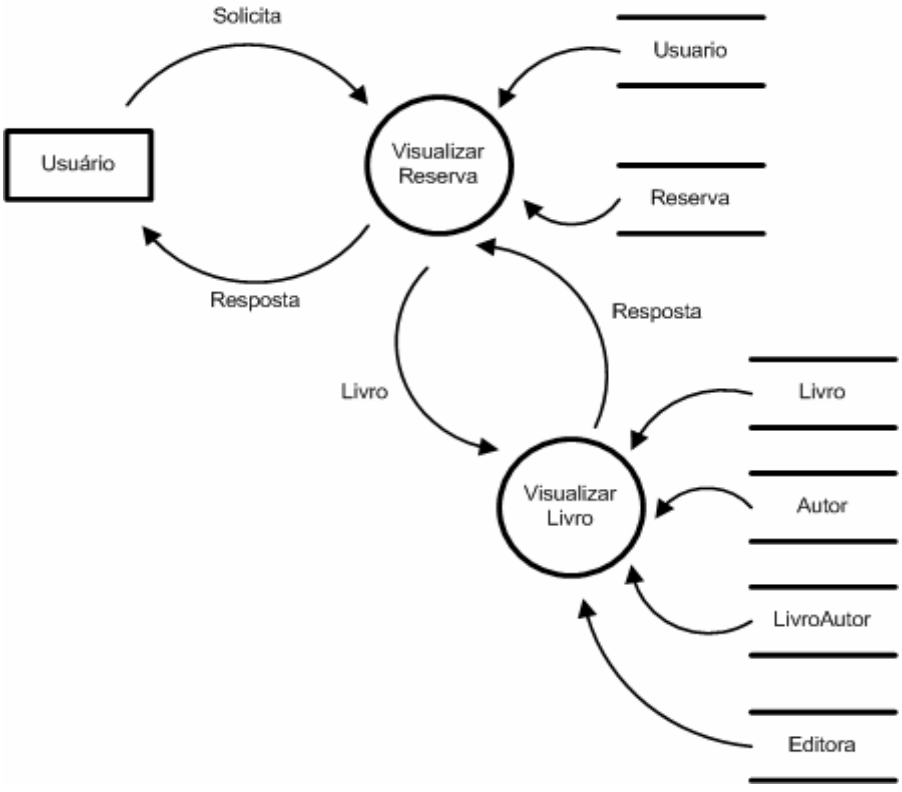

Figura A.27 – DFD visualizar reserva

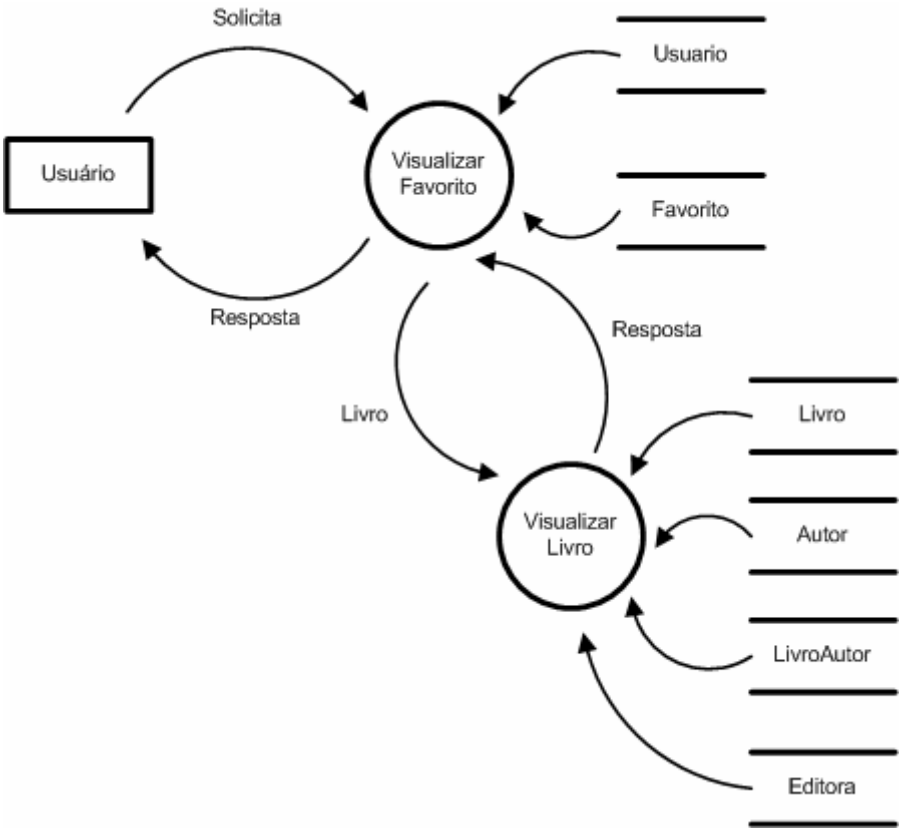

Figura A.28 – DFD visualizar favorito

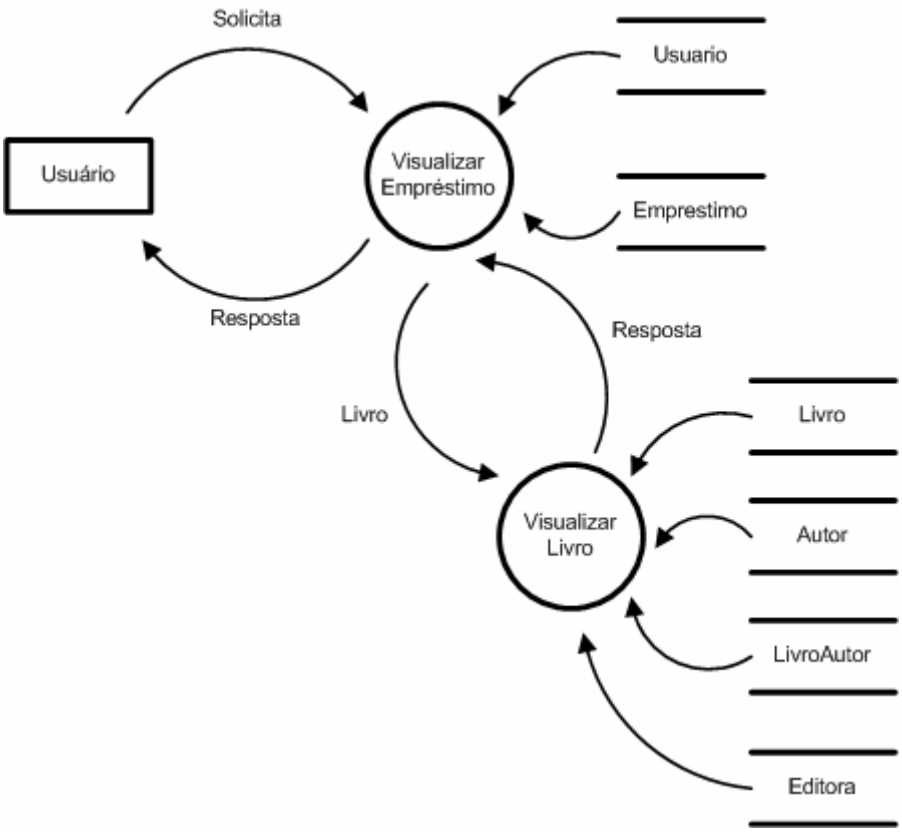

Figura A.29 – DFD visualizar empréstimo

Apêndice B – Projeto detalhado

Fazer cadastro no sistema

```
### CÓDIGO ###

      recebe_dados();

      se ( verifica_dados() != OK )
      {
            mensagem_erro;
      }

      se ( existe_usuario() != NAO_EXISTE )
      {
            mensagem_erro();
      }

      se ( cadastra_dados() != OK )
      {
            mensagem_erro();
      }

      mensagem_ok();

### FIM DO CÓDIGO ###
```

Recuperar senha

```
### CÓDIGO ###

      recebe_dados();

      se ( verifica_dados() != OK )
      {
            mensagem_erro;
      }

      se ( existe_usuario() != NAO_EXISTE )
      {
            mensagem_erro();
      }

      se ( cadastra_novasenha() != OK )
      {
            mensagem_erro();
      }

      se ( envia_novasenha() != OK )
      {
            mensagem_erro();
      }

      mensagem_ok();

### FIM DO CÓDIGO ###
```

Sair do sistema

```
### CÓDIGO ###

      termina_sessao();

### FIM DO CÓDIGO ###
```

Alterar senha

```
### CÓDIGO ###

      recebe_dados();

      se ( verifica_dados() != OK )
      {
            mensagem_erro;
      }

      se ( cadastra_novasenha() != OK )
      {
            mensagem_erro();
      }

      mensagem_ok();

### FIM DO CÓDIGO ###
```

Cadastrar autor

```
### CÓDIGO ###

      recebe_dados();

      se ( verifica_dados() != OK )
      {
            mensagem_erro;
      }

      se ( cadastra_autor() != OK )
      {
            mensagem_erro();
      }

      mensagem_ok();

### FIM DO CÓDIGO ###
```

Cadastrar editora

```
### CÓDIGO ###

      recebe_dados();

      se ( verifica_dados() != OK )
      {
            mensagem_erro;
      }

      se ( cadastra_editora() != OK )
```

```
      {
            mensagem_erro();
      }

      mensagem_ok();

### FIM DO CÓDIGO ###
```

Alterar dados pessoais

```
### CÓDIGO ###

      recebe_dados();

      se ( verifica_dados() != OK )
      {
            mensagem_erro;
      }

      se ( edita_usuario() != OK )
      {
            mensagem_erro();
      }

      mensagem_ok();

### FIM DO CÓDIGO ###
```

Editar autor

```
### CÓDIGO ###

      recebe_dados();

      se ( verifica_dados() != OK )
      {
            mensagem_erro;
      }

      se ( edita_autor() != OK )
      {
            mensagem_erro();
      }

      mensagem_ok();

### FIM DO CÓDIGO ###
```

Remover autor

```
### CÓDIGO ###

      recebe_dados();

      se ( verifica_dados() != OK )
      {
            mensagem_erro;
      }
```

```
      se ( remove_autor() != OK )
      {
            mensagem_erro();
      }

      mensagem_ok();

### FIM DO CÓDIGO ###
```

Editar editora

```
### CÓDIGO ###

      recebe_dados();

      se ( verifica_dados() != OK )
      {
            mensagem_erro;
      }

      se ( edita_editora() != OK )
      {
            mensagem_erro();
      }

      mensagem_ok();

### FIM DO CÓDIGO ###
```

Remover editora

```
### CÓDIGO ###

      recebe_dados();

      se ( verifica_dados() != OK )
      {
            mensagem_erro;
      }

      se ( remove_editora() != OK )
      {
            mensagem_erro();
      }

      mensagem_ok();

### FIM DO CÓDIGO ###
```

Cadastrar livro

```
### CÓDIGO ###

      recebe_dados();

      se ( verifica_dados() != OK )
      {
            mensagem_erro;
      }
```

```
      se ( cadastra_livro() != OK )
      {
            mensagem_erro();
      }

      enquanto ()
      {
            se ( cadastra_livroautor() != OK )
            {
                  mensagem_erro();
            }
      }

      mensagem_ok();

### FIM DO CÓDIGO ###
```

Editar livro

```
### CÓDIGO ###

      recebe_dados();

      se ( verifica_dados() != OK )
      {
            mensagem_erro;
      }

      se ( edita_livro() != OK )
      {
            mensagem_erro();
      }

      enquanto ()
      {
            se ( cadastra_livroautor() != OK )
            {
                  mensagem_erro();
            }
            se ( remove_livroautor() != OK )
            {
                  mensagem_erro();
            }
      }

      mensagem_ok();

### FIM DO CÓDIGO ###
```

Remover livro

```
### CÓDIGO ###

      recebe_dados();

      se ( verifica_dados() != OK )
      {
            mensagem_erro;
      }
```

```
      se ( remove_livro() != OK )
      {
            mensagem_erro();
      }

      enquanto ()
      {
            se ( remove_livroautor() != OK )
            {
                  mensagem_erro();
            }
      }

      mensagem_ok();

### FIM DO CÓDIGO ###
```

Editar alerta

```
### CÓDIGO ###

      recebe_dados();

      se ( verifica_dados() != OK )
      {
            mensagem_erro;
      }

      se ( edita_alerta() != OK )
      {
            mensagem_erro();
      }

      se ( edita_arquivo_alerta() != OK )
      {
            mensagem_erro();
      }

      mensagem_ok();

### FIM DO CÓDIGO ###
```

Remover alerta

```
### CÓDIGO ###

      recebe_dados();

      se ( verifica_dados() != OK )
      {
            mensagem_erro;
      }

      se ( remove_alerta() != OK )
      {
            mensagem_erro();
      }

      se ( remove_arquivo_alerta() != OK )
      {
```

```
            mensagem_erro();
      }

      mensagem_ok();

### FIM DO CÓDIGO ###
```

Enviar alerta

```
### CÓDIGO ###

      enquanto () {
            se ( verifica_arquivo_alerta() == DATA_HORARIO_ATUAL )
            {
                  envia_alerta;
            }
      }

### FIM DO CÓDIGO ###
```

Procurar livro

```
### CÓDIGO ###

      recebe_dados();

      se ( verifica_dados() != OK )
      {
            mensagem_erro;
      }

      se ( procurar_livro() != OK )
      {
            mensagem_erro();
      }

      exibe_resultados();

### FIM DO CÓDIGO ###
```

Procurar usuário

```
### CÓDIGO ###

      recebe_dados();

      se ( verifica_dados() != OK )
      {
            mensagem_erro;
      }

      se ( procurar_usuario() != OK )
      {
            mensagem_erro();
      }

      exibe_resultados();

### FIM DO CÓDIGO ###
```

Procurar alerta

```
### CÓDIGO ###

      recebe_dados();

      se ( verifica_dados() != OK )
      {
            mensagem_erro;
      }

      se ( procurar_alerta() != OK )
      {
            mensagem_erro();
      }

      exibe_resultados();

### FIM DO CÓDIGO ###
```

Visualizar usuário

```
### CÓDIGO ###

      recebe_dados();

      se ( verifica_dados() != OK )
      {
            mensagem_erro;
      }

      se ( pega_dados_usuario() != OK )
      {
            mensagem_erro();
      }

      exibe_usuario();

### FIM DO CÓDIGO ###
```

Visualizar alerta

```
### CÓDIGO ###

      recebe_dados();

      se ( verifica_dados() != OK )
      {
            mensagem_erro;
      }

      se ( pega_dados_alerta() != OK )
      {
            mensagem_erro();
      }

      exibe_alerta();

### FIM DO CÓDIGO ###
```

Remover usuário

```
### CÓDIGO ###

      recebe_dados();

      se ( verifica_dados() != OK )
      {
            mensagem_erro;
      }

      se ( remove_usuario() != OK )
      {
            mensagem_erro();
      }

      mensagem_ok();

### FIM DO CÓDIGO ###
```

Colocar livro nos favoritos

```
### CÓDIGO ###

      recebe_dados();

      se ( verifica_dados() != OK )
      {
            mensagem_erro;
      }

      se ( coloca_favoritos() != OK )
      {
            mensagem_erro();
      }

      mensagem_ok();

### FIM DO CÓDIGO ###
```

Devolver livro

```
### CÓDIGO ###

      recebe_dados();

      se ( verifica_dados() != OK )
      {
            mensagem_erro;
      }

      se ( devolve_livro() != OK )
      {
            mensagem_erro();
      }

      se ( verifica_livro_reservado == OK )
      {
            atualiza_reserva();
            cadastra_alerta();
```

```
      }

      mensagem_ok();

### FIM DO CÓDIGO ###
```

Visualizar livro

```
### CÓDIGO ###

      recebe_dados();

      se ( verifica_dados() != OK )
      {
            mensagem_erro;
      }

      se ( pega_dados_livro() != OK)
      {
            mensagem_erro();
      }

      exibe_livro();

### FIM DO CÓDIGO ###
```

Visualizar reserva, empréstimo e favorito

```
### CÓDIGO ###

      recebe_dados();

      se ( verifica_dados() != OK )
      {
            mensagem_erro;
      }

      se ( pega_dados_reserva/emprestimo/favorito() != OK)
      {
            mensagem_erro();
      }

      exibe_resultados();

### FIM DO CÓDIGO ###
```

Apêndice C – Plano de Testes

C.1. Introdução

Esse apêndice irá mostrar a documentação do plano de testes para o sistema desenvolvido nesse projeto final.

Como introdução, apresentaremos a finalidade, o escopo e o resumo do documento. Após, iremos falar dos testes propriamente ditos.

C.1.1. Finalidade

O objetivo deste documento é descrever o plano de testes do sistema, de forma que todos os pontos deste sejam percorridos e conseqüentemente testados para verificar o seu bom funcionamento.

C.1.2. Escopo

O sistema que está sendo desenvolvido chama-se Sistema de Gerenciamento Bibliotecário Utilizando Interface WEB/SMS e tem como objetivo desenvolver um software de gerência de livros de uma biblioteca.

Seus benefícios são o controle integrado do acervo da biblioteca, a redução do trabalho manual com papéis e a facilidade para o cliente realizar busca e reservas de livros.

C.1.3. Resumo

Serão discutidas formas de se testar o sistema, através de sua divisão em módulos e, posteriormente, em *clusters* e seus agrupamentos. Em cada um desses pontos, especificaremos qual tipo de testes serão realizados, inclusive, os explicando conceitualmente.

C.2. Descrição Geral

Nesse capítulo fazemos uma breve descrição dos testes, incluindo as estratégias e os ambientes envolvidos nos mesmos.

C.2.1. Estratégias

Neste documento serão planejados testes modulares que se dividirão em caminhos e laços. A integração entre os módulos será feita no sentido descendente. Teremos ainda, um plano de testes para requisitos não funcionais.

C.2.2. Ambiente de testes

Os ambientes de testes que mais utilizaremos será a interface WEB e o servidor onde se encontrará a base de dados MySQL e os Scripts de agendamento de alertas.

C.3. Testes

Nesse capítulo abordamos os testes propriamente ditos. Para tanto, dividimos o mesmo em dois subcapítulos.

O primeiro trata dos testes modulares e o segundo dos testes integradores. Vale ressaltar que utilizaremos o conceito de caixa branca e caixa preta.

Entenda-se por caixa branca, técnica de teste que avalia o comportamento interno do componente de software. Essa técnica trabalha diretamente sobre o código-fonte do componente de software para avaliar aspectos tais como: teste de condição, teste de fluxo de dados, teste de ciclos e teste de caminhos lógicos.

Entenda-se por caixa preta, técnica de teste em que o componente de software a ser testado é abordado como se fosse uma caixa-preta, ou seja, não se considera o comportamento interno do mesmo. Dados de entrada são fornecidos, o teste é executado e o resultado obtido é comparado a um resultado esperado previamente conhecido.

C.3.1. Testes modulares

Nesse subcapítulo mostramos os módulos do sistema. Para fazer isso, ilustraremos os mesmos por tipo de usuário. No sistema temos dois: Administrador e cliente.

A primeira figura mostra os módulos que incluem as ações do cliente e as demais figuras incluem as ações do administrador. Vale ressaltar que ambas as figuras ilustram o processo desde o inicio, autenticação no sistema, até a última função possível vista de cima para baixo.

Nas figuras não está representado o módulo de sair do sistema (fazer logoff), no entanto, em qualquer momento após ser feita a autenticação, o usuário poderá efetuar sua saída do sistema.

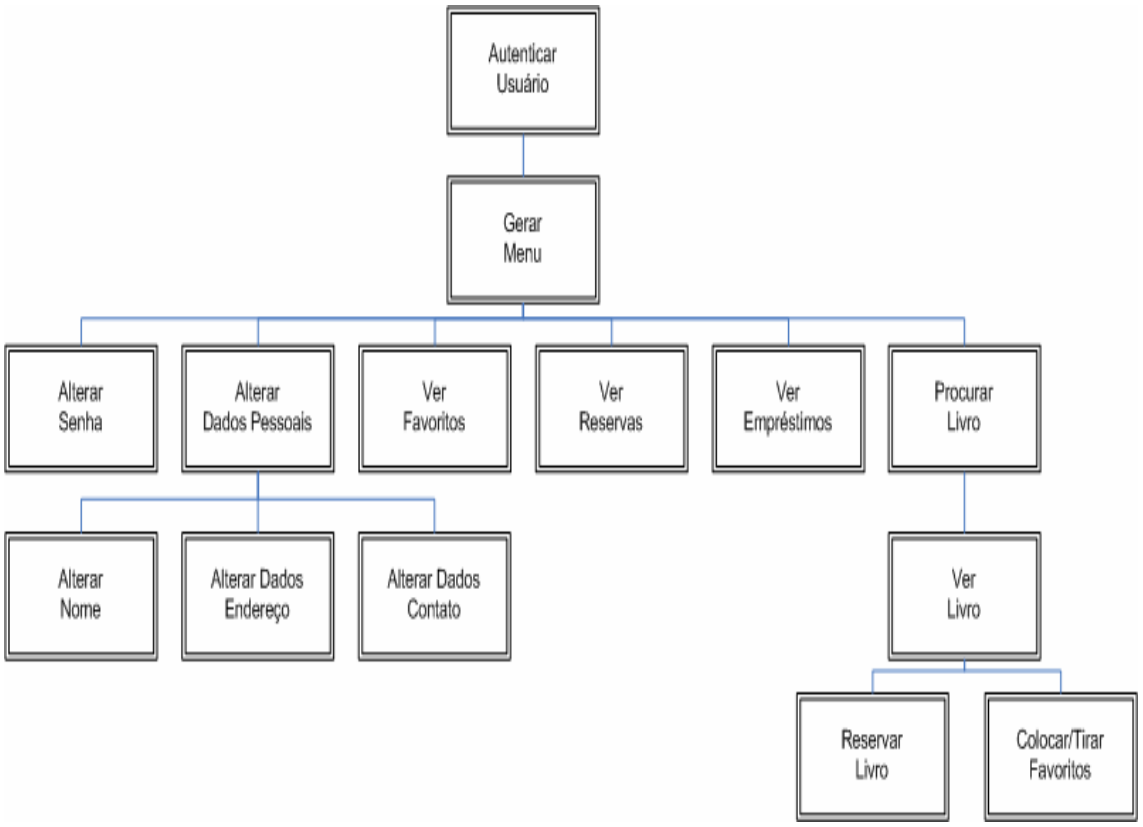

Figura C.1 – Testes Modulares para Usuário Cliente

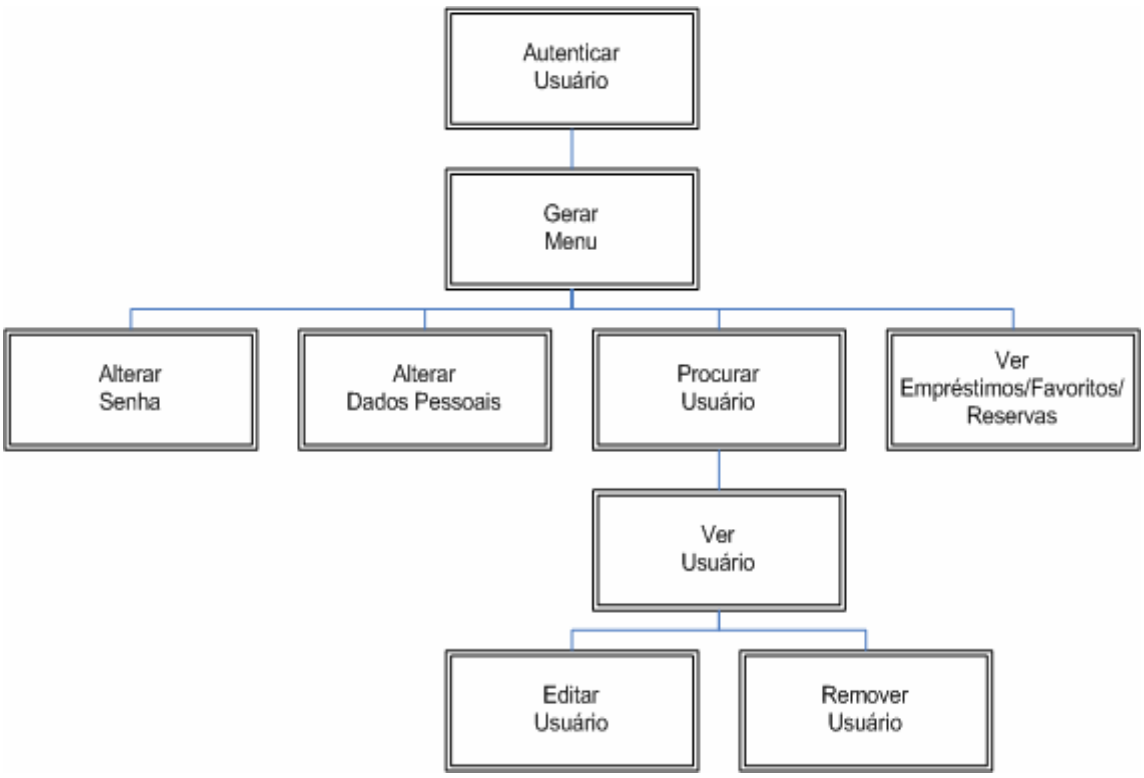

Figura C.2 – Testes Modulares para Usuário Administrador (informações de usuários)

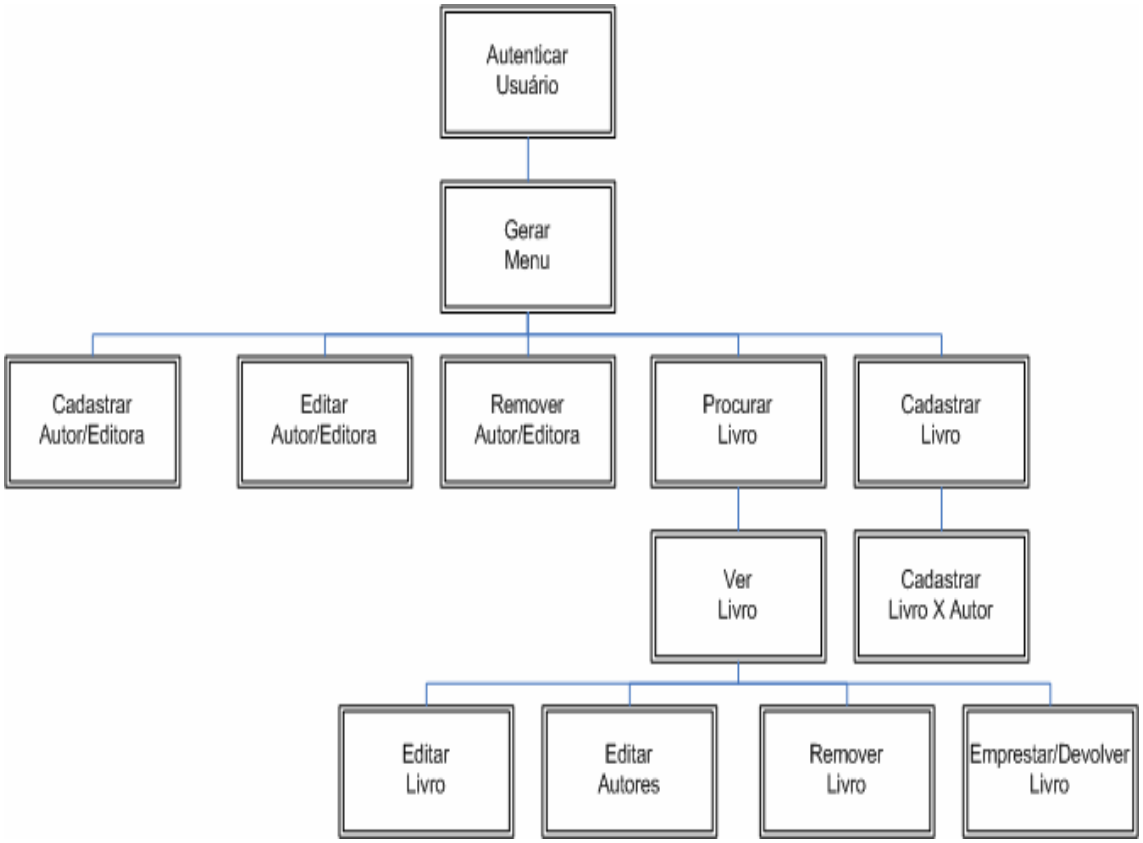

Figura C.3 – Testes Modulares para Usuário Administrador (informações de livros)

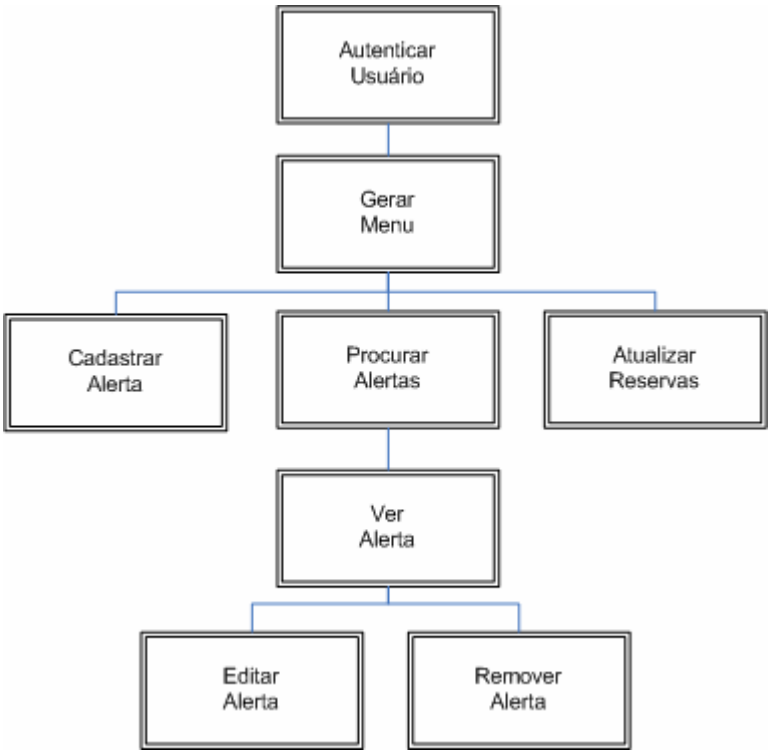

Figura C.4 – Testes Modulares para Usuário Administrador (outras informações)

C.3.1.1. Testes a realizar

Serão realizados, portanto, testes de caixa branca em cada um desses módulos.

C.3.1.2. Agrupamento dos módulos em *clusters*

Faremos, então, o seguinte agrupamento dos módulos:

1. Autenticar Usuário e Sair Sistema.
2. Gerar Menu.
3. Cadastrar Autor, Editar Autor e Remover Autor.
4. Cadastrar Editora, Editar Editora e Remover Editora.
5. Cadastrar Livro, Cadastrar Livro X Autor, Editar Livro e Remover Livro.
6. Procurar Livro, Ver Livro e Reservar Livro.
7. Procurar Livro, Ver Livro e Colocar Livro nos Favoritos.
8. Procurar Livro, Ver Livro e Emprestar Livro.
9. Procurar Livro, Ver Livro e Devolver Livro.
10. Cadastrar Alerta, Editar Alerta e Remover Alerta.
11. Atualizar Reservas.
12. Alterar Senha e Alterar Dados Pessoais.
13. Ver Empréstimos, Ver Reservas e Ver Favoritos.

C.3.2. Testes de integração

Nesse subcapítulo abordamos os testes integrados da aplicação. Ou seja, realizamos os testes de união dos módulos como um todo.

Isso será feito através de testes seqüenciais dos clusters, onde daremos algumas entradas e observaremos as saídas.

C.3.2.1. Seqüência de integração dos clusters

Testaremos as seguintes seqüências:

- 1
- 1, 2
- 1, 2, 3
- 1, 2, 4
- 1, 2, 5
- 1, 2, 3, 4, 5
- 1, 2, 6
- 1, 2, 7
- 1, 2, 8
- 1, 2, 9
- 1, 2, 10
- 1, 2, 11
- 1, 2, 12
- 1, 2, 13

C.3.2.2. Testes a realizar

Serão realizados, portanto, testes de caixa preta.

Acrescentamos ainda nesses testes, os que fazem parte do cadastro no sistema e perda de senha. Não citamos nenhum módulo, pois os mesmos serão testados já integrados no sistema. Serão feitos testes de caixa preta das seguintes ações: Cadastro no sistema e esqueceu a senha.

C.3.3. Testes para requisitos não funcionais

Serão testados a amigabilidade do sistema para as funções onde haja interação com o cliente da biblioteca e a segurança do sistema, tratando dos casos onde alguém tente efetuar alguma operação sem estar efetivamente logado ou que não tenha permissão.

C.4. Validação

Será criada a base de dados e ela será preparada para o funcionamento normal para assim apresentar o sistema ao cliente.

Apêndice D – Manual do usuário

D.1. Introdução

Esse manual visa atender as dúvidas dos usuários do sistema de gerenciamento bibliotecário. Os três primeiros capítulos desse apêndice abordam qualquer tipo de usuário.

Como existem dois tipos de usuário, cliente e administradores, e, para cada um deles, existem muitas funções, iremos apenas citar as mesmas e fazer breves comentários de como elas devem ser executadas. Muitas páginas do sistema não serão ilustradas aqui; no entanto, a maioria delas é bastante intuitiva e amigável para o usuário.

Vale ressaltar que, sempre ao fim de uma operação, é exibido no topo da página o resultado da mesma. Em caso de sucesso a mensagem será exibida na cor verde. Caso ocorra algum erro, a mensagem será exibida em vermelho.

D.2. Tela inicial

A tela principal do sistema é esta ilustrada na figura abaixo. Podemos observar pela mesma, que existem três opções: Fazer o cadastro, recuperar senha esquecida e autenticar-se no sistema. Nos próximos capítulos iremos abordar cada uma delas.

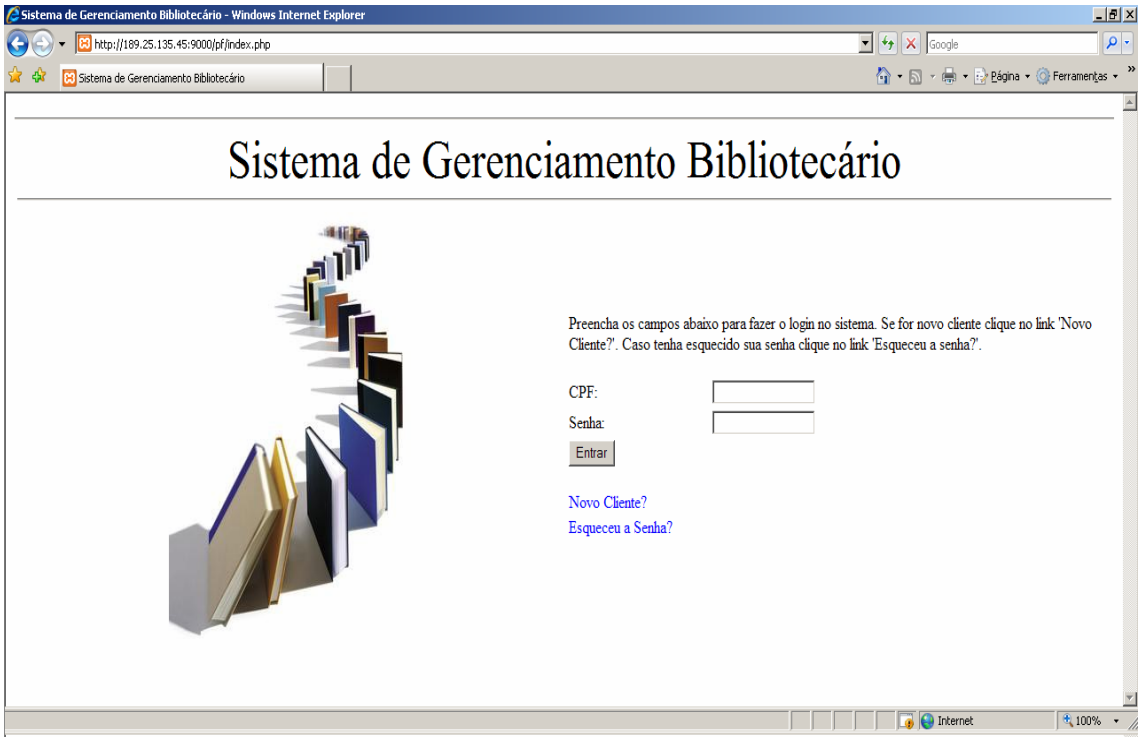

Figura D.1 – Página inicial

D.3. Cadastro no sistema

Após clicar no link ‘Novo Cliente?’, teremos a seguinte tela para preenchimento de dados:

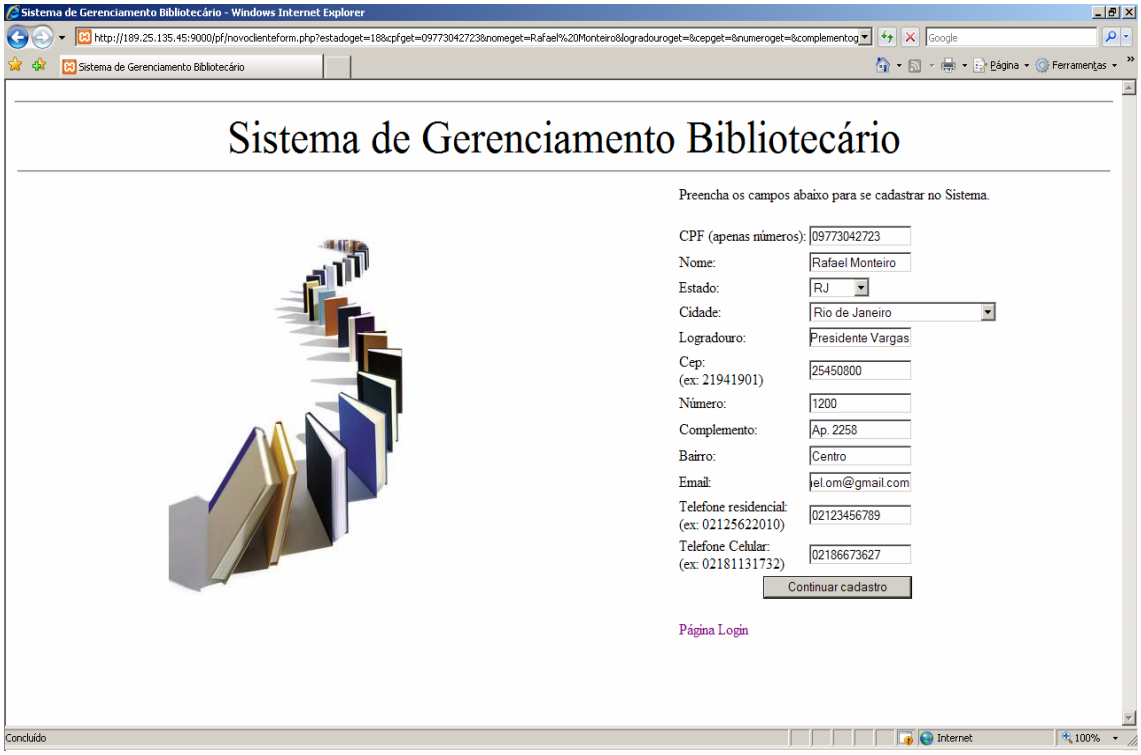

Figura D.2 – Cadastro parte 1

Os campos devem ser preenchidos conforme exemplificado na imagem acima. Após seu correto preenchimento, teremos a tela de preenchimento de senha no sistema:

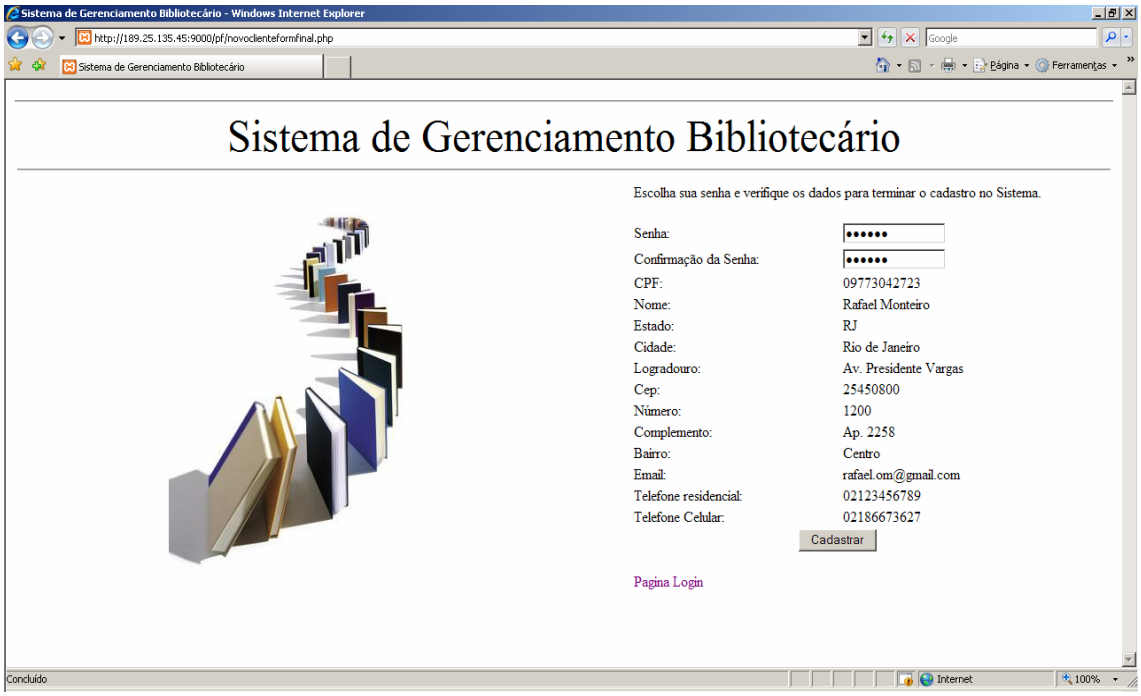

Figura D.3 – Cadastro parte 2

Se todos os dados estiverem corretamente preenchidos e o usuário for novo no sistema, irá aparecer a tela de sucesso no cadastro, conforme a figura abaixo:

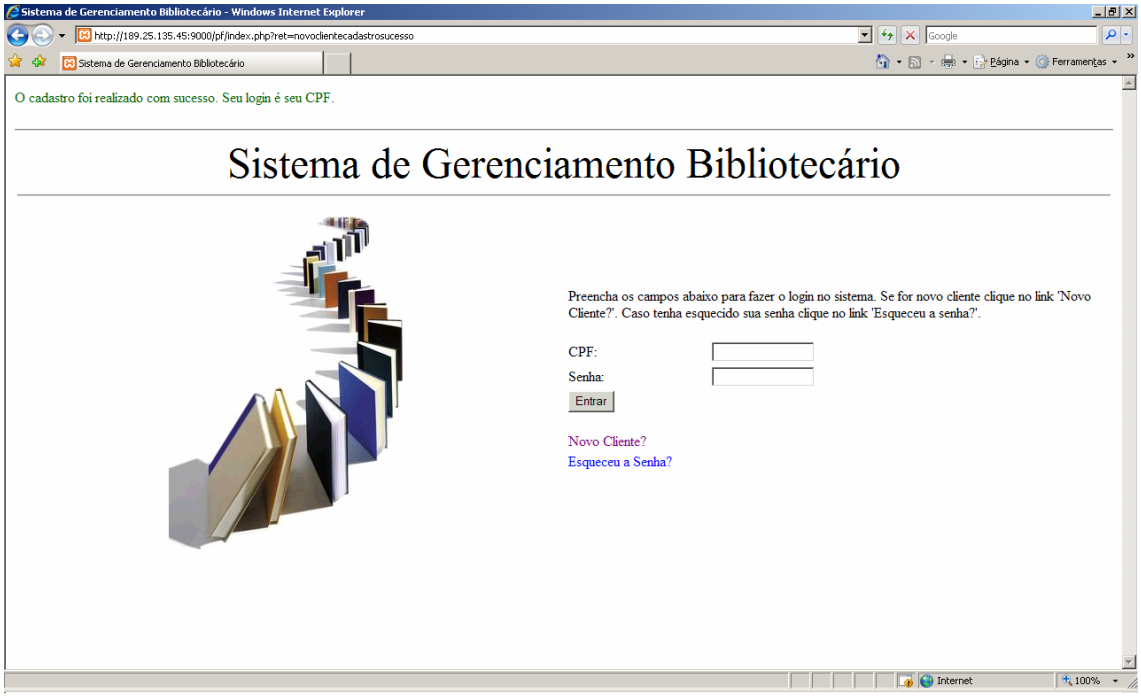

Figura D.4 – Cadastro concluído

D.4. Recuperar senha esquecida

Caso o usuário tenha esquecido sua senha, será possível receber uma nova senha, pelo meio de envio desejado. Para isso, o mesmo deverá selecionar a opção ‘Esqueceu a Senha?’ no menu principal e preencher a seguinte tela:

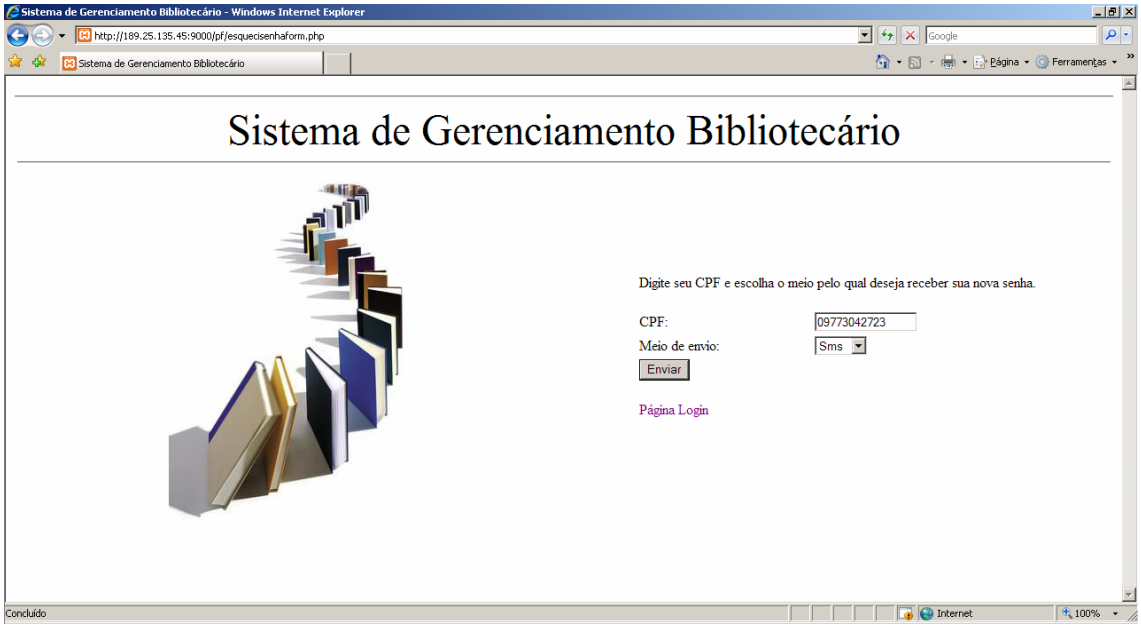

Figura D.5 – Esqueci a senha

Se os dados fornecidos estiverem corretos, será exibida a mensagem de sucesso no envio da nova senha:

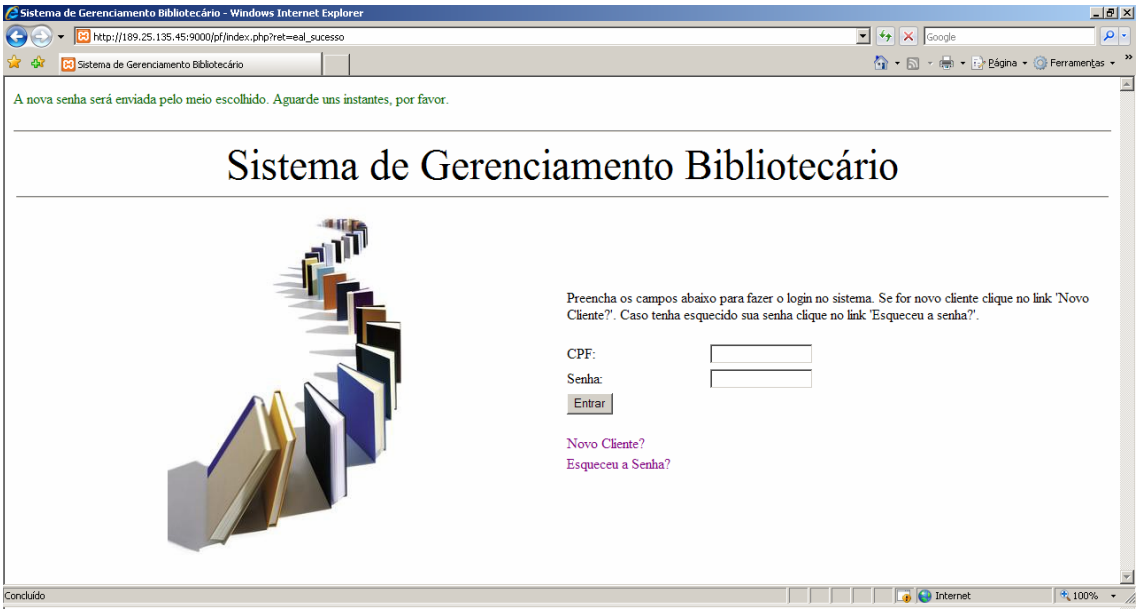

Figura D.6 – Esqueci a senha concluído

D.5. Autenticar-se no sistema

Ao entrar com seu cpf e senha na tela principal do sistema, o usuário irá realizar a autenticação. Caso o cpf e a senha estejam corretos ele receberá a tela de menu com as opções possíveis a se realizar, conformes as próximas figuras ilustram, sendo uma para clientes e a outra para administradores:

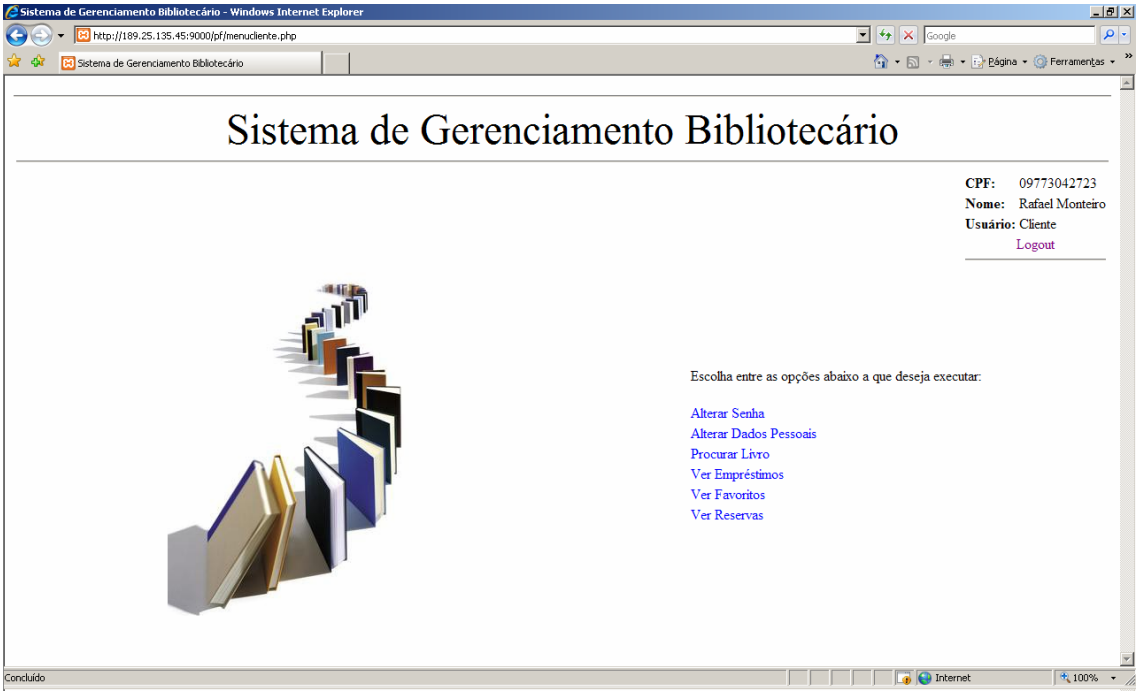

Figura D.7 – Menu do cliente

Figura D.8 – Menu do administrador

Para cada uma das opções disponibilizadas no menus acima, mostraremos sua operação nos próximos subcapítulos. No entanto, devido sua fácil operabilidade, iremos omitir diversas figuras. Voltamos a salientar que no fim de cada execução é mostrada uma mensagem no topo da tela informando o sucesso ou insucesso da mesma.

D.6. Funções de clientes

D.6.1. Alterar senha

Ao selecionar a opção alterar senha no meu, o cliente receberá uma página pedindo a senha atual, a nova senha e a confirmação da nova senha. Preenchendo esses campos corretamente, a troca da senha será executada e o menu será novamente exibido com a mensagem de sucesso no topo.

Caso ocorra algum problema, a mesma página de alteração de senha pode ser exibida ao cliente, mostrando no seu topo o erro ocorrido. Há a possibilidade de o sistema exibir o menu e reportar o erro no topo também. Isso varia de acordo com o erro.

D.6.2. Alterar dados pessoais

Ao selecionar a opção de alterar dados pessoais, o cliente receberá uma página com todos os seus dados. Nessa mesma página ele poderá optar por qual tipo de dados quer alterar: Nome, contato ou endereço.

Uma vez escolhido o tipo de dados que deseje alterar, será exibida uma tela com os dados atuais preenchidos, onde o usuário pode alterá-los e selecionar o botão de confirmação para registrar no sistema os novos valores.

Caso tudo seja realizado sem nenhum problema, o menu será exibido, informando o sucesso da operação. Caso ocorra algum erro, a tela de alteração será exibida novamente, informando o erro ocorrido.

D.6.3. Procurar livro

Caso a opção de busca de livros seja selecionada, uma página contendo os campos de busca será exibida. O usuário deve preencher com os filtros desejados e confirmar a solicitação.

Na próxima tela, serão exibidos os resultados para aqueles filtros. Caso ocorra algum erro, o mesmo será exibido no topo da tela.

Se a busca retornar algum livro, será possível visualizá-lo, como será mais explicado no próximo tópico.

D.6.4. Ver livro

Se o usuário selecionar um livro, o mesmo será exibido em uma nova tela, com todas as suas informações e com as opções disponíveis. No caso dos clientes, essas opções são a de reservar e colocar nos favoritos.

Para ambas as opções, a próxima tela a ser exibida é o menu, informando-se no topo o resultado da operação (reservar ou colocar nos favoritos).

D.6.5. Ver empréstimos, Ver favoritos e Ver reservas

Essas funções não exigem entrada do usuário e exibirão uma tela com os livros pedidos, sejam eles os emprestados, favoritos ou reservados. Nessa tela será possível selecionar um livro e visualizá-lo conforme explica no tópico anterior.

D.7. Funções de administradores

D.7.1. Alterar senha

O procedimento é bastante semelhante ao do cliente. No entanto, o administrador tem o campo usuário, que deve ser preenchido informando quem terá sua senha alterada.

Ao confirmar a alteração de senha, o menu será exibido com a mensagem resultante da operação, seja sucesso ou erro. Dependendo do erro, a tela de alteração será exibida novamente.

D.7.2. Alterar dados pessoais

É exatamente igual ao procedimento que o cliente deve fazer, diferenciando-se apenas pela tela abaixo onde o administrador deve informar o cpf do usuário que deseja alterar os dados pessoais.

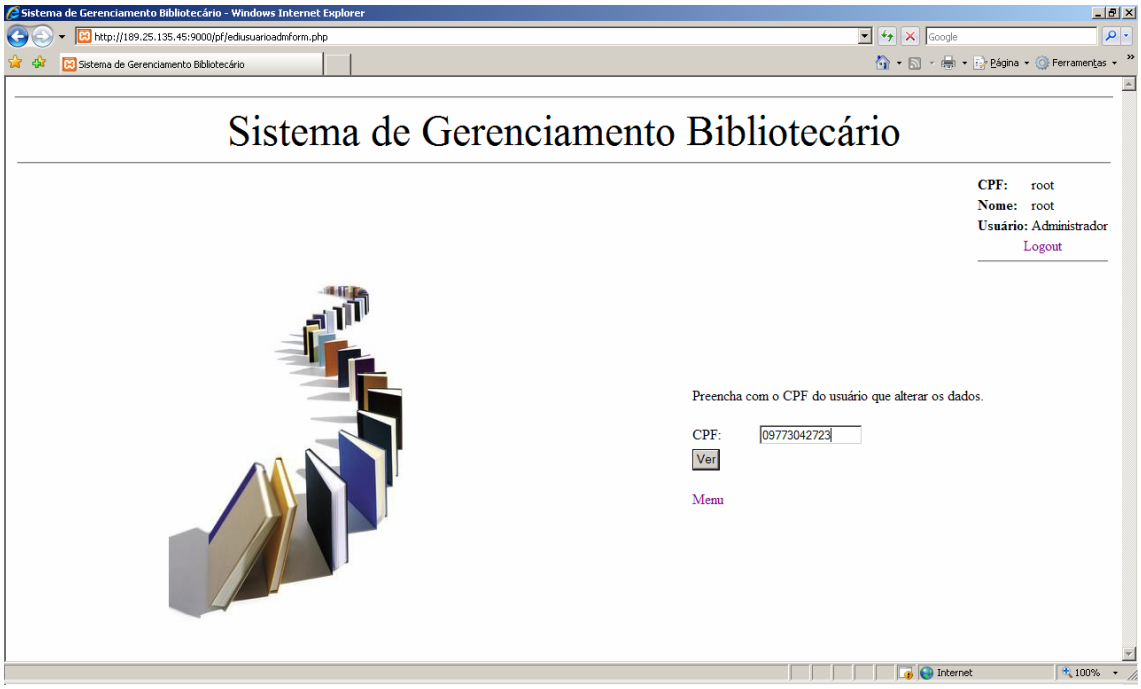

Figura D.9 – Seleção do usuário a ser alterado

Conforme já explicado na alteração de dados do cliente, será exibida uma tela onde os dados poderão ser vizualizados e onde poderá ser selecionado qual deles serão alterados.

No próxima figura ilustramos essa tela, com as opções disponíveis para alteração. No caso do administrador, será possível ainda alterar o tipo do usuário.

Figura D.10 – Visualização dos dados a serem alterados

D.7.3. Cadastro, edição e remoção de autores e editoras

No caso do cadastro, será exibida uma tela para preenchimento com o nome do auto ou editora. Ao confirmar, será exibido menu com o resultado da operação no topo do mesmo. Poderá, ainda, ser exibida novamente a tela de cadastro caso alguém erro ocorra. Nesse caso, o mesmo também será exibido no topo da tela.

Para a edição, será exibida uma caixa de seleção onde o autor ou editora a ser editado seja escolhido. Logo abaixo terá um campo para o preenchimento do novo nome. Se a operação for executada com sucesso, o menu será exibido com a mensagem correspondente. Em caso de erro, o mesmo será exibido, seja no menu ou na própria tela de edição, dependendo do erro.

Já para a remoção, será exibida apenas a caixa de seleção do autor ou editora a ser removida. O processo de resposta à operação é igual ao da edição.

D.7.4. Cadastro de alertas

O procedimento é o semelhante ao cadastro de autores ou editoras.

D.7.5. Cadastro de livros

O procedimento é o semelhante ao cadastro de autores ou editoras. No entanto, após a conclusão do cadastro das informações básicas do livro, será necessário cadastrar as relações livro autor do mesmo.

Para isso, será exibida uma tela onde o autor pode ser selecionado. Além disso, será oferecida uma caixa de seleção onde o administrador pode optar por encerrar o cadastro do livro não adicionando aquele autor, adicionando apenas aquele autor ou adicionando aquele autor e continuar o processo de seleção de autores par ao livro. Apenas para o último caso que essa tela é exibida novamente (já que outro autor deve ser cadastrado para o livro). Nos dois primeiros casos, o menu é exibido.

Para cada operação, será sempre exibido no topo da tela seguinte o resultado da mesma.

D.7.6. Procurar livro

Caso a opção de busca de livros seja selecionada, uma página contendo os campos de busca será exibida. O usuário deve preencher com os filtros desejados e confirmar a solicitação.

Na próxima tela, serão exibidos os resultados para aqueles filtros. Caso ocorra algum erro, o mesmo será exibido no topo da tela.

Se a busca retornar algum livro, será possível visualizá-lo, como será mais explicado no próximo tópico.

D.7.7. Ver livro

Se o usuário selecionar um livro, o mesmo será exibido em uma nova tela, com todas as suas informações e com as opções disponíveis. No caso dos administradores, essas opções são a de editar, remover e emprestar ou devolver.

Para cada uma dessas as opções teremos um tópico na seqüência, as explicando melhor.

D.7.8. Editar livro

Existem duas opções de edição de livros. Na primeira editamos os dados básicos. Esse procedimento é semelhante à edição de autores ou editoras. Na segunda, editamos as relações livro autor para o livro em questão.

Nessa última, podemos remover autores cadastrados para o livro, ou adicionar outros autores. Ambos os processos são semelhantes, respectivamente, a remoção de autores ou editoras e ao próprio cadastro de relações livro autor no cadastro de um novo livro.

Abaixo podemos ver essa tela de edição.

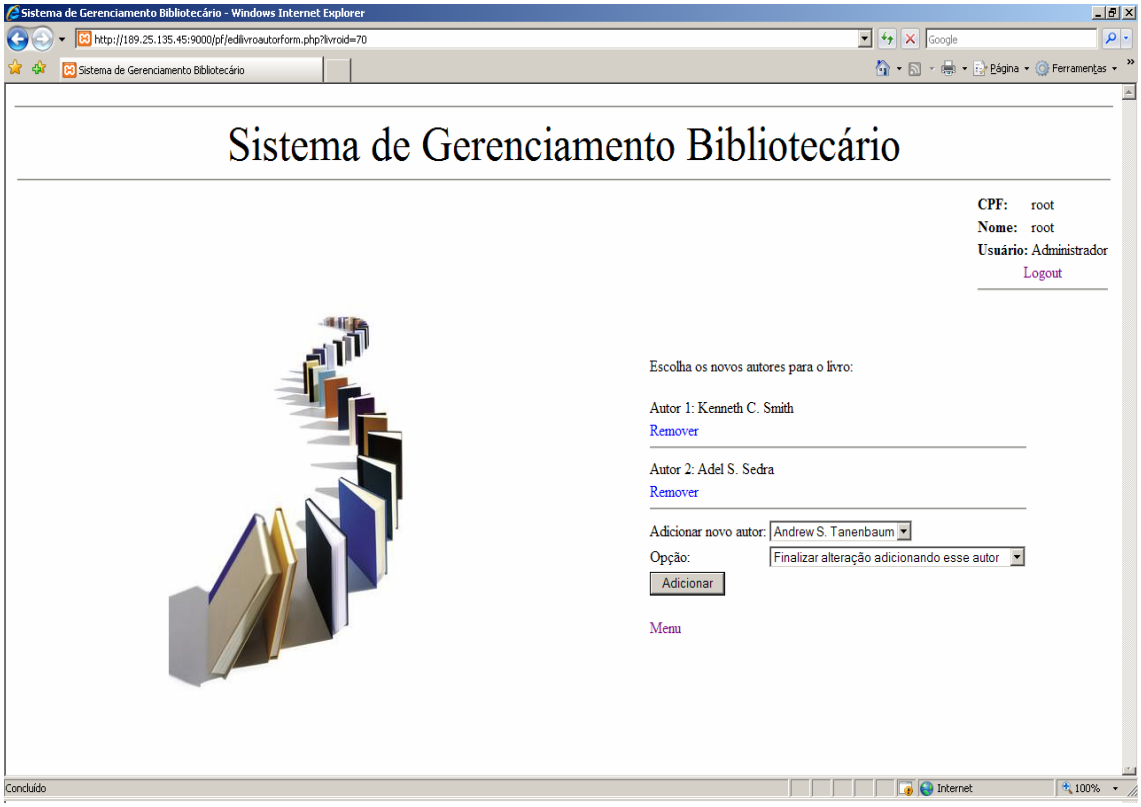

Figura D.11 – Alteração de livros

Note que os links ‘Remover’ servem para remover autores já cadastrados e as caixas de opção servem para cadastrar novos autores para o livro em questão.

Para as operações, o menu ou a própria tela de edição podem ser exibida, sempre contendo no topo o resultado da última operação realizada.

D.7.9. Remover livro

Se o usuário decidir remover um livro, será pedido a sua senha antes de ocorrer essa remoção. Após isso, o menu é exibido com o resultado da mesma.

D.7.10. Emprestar ou devolver livro

Se o livro estiver emprestado, será exibida opção de devolver, onde deverão ser preenchidos alguns dados da devolução, como multa associada e observações. O resultado da operação será exibido na próxima tela, que ode ser o menu ou a própria tela de devolução caso ocorra algum erro.

D.7.11. Procurar alerta

Caso a opção de busca de alertas seja selecionada, uma página contendo os campos de busca será exibida. O usuário deve preencher com os filtros desejados e confirmar a solicitação.

Na próxima tela, serão exibidos os resultados para aqueles filtros. Caso ocorra algum erro, o mesmo será exibido no topo da tela.

Se a busca retornar algum alerta, será possível visualizá-lo. Essa visualização funciona da mesma forma que para os livros. Só que nesse caso, apenas será possível editar ou remover. Para ambos os casos o procedimento é semelhante ao dos livros, apenas alterando os dados a serem informados.

D.7.12. Procurar usuário

Caso a opção de busca de usuários seja selecionada, uma página contendo os campos de busca será exibida. O usuário deve preencher com os filtros desejados e confirmar a solicitação.

Na próxima tela, serão exibidos os resultados para aqueles filtros. Caso ocorra algum erro, o mesmo será exibido no topo da tela.

Se a busca retornar algum usuário, será possível visualizá-lo. Essa visualização funciona da mesma forma que para os livros. Só que nesse caso, apenas será possível editar ou remover. Para ambos os casos o procedimento é semelhante ao dos livros, apenas alterando os dados a serem informados.

D.7.13. Ver empréstimos, Ver favoritos e Ver reservas

É exatamente igual ao procedimento que o cliente deve fazer, diferenciando-se apenas pela tela onde o administrador deve informar o cpf do usuário que ver os livros emprestados, favoritos ou reservados. Após isso o procedimento é idêntico.

D.7.14. Atualizar reservas

Essa função é executada pelo menu e seu resultado é mostrado no mesmo. Não existem entradas de dados.

D.8. Sair do sistema

Depois de autenticado no sistema, tanto o cliente como o administrador poderão efetuar sua saída a qualquer momento apenas clicando em ‘Logout’ no menu superior direito.

Apêndice E – Código fonte

Como o código fonte do sistema é muito extenso, nesse apêndice iremos apenas informar quais os arquivos que o compõe e como eles estão estruturados. Todos eles podem ser encontrados na versão eletrônica desse relatório.

Primeiramente, dividimos os códigos nas seguintes categorias:

- Os códigos *web*, compostos por todos os arquivos com extensão .php, que são os responsáveis pela interface exibida ao usuário final e pelas funções de acesso ao banco de dados. Alguns desses códigos também geram arquivos no servidor, assim como chamam comandos no mesmo.

- Os códigos de agendamento dos alertas, compostos todos por Shell Scripts no servidor.

- Os códigos dos programas de envio das mensagens propriamente ditas. O código do envio de *emails* está disponível e foi feito em C. O código do envio de *sms* não será mostrado, porém pode ser conseguido no site da biblioteca utilizada para esse fim.

- Os códigos de criação do banco de dados.

Nos próximos subcapítulos iremos mostrar como esses códigos estão estruturados.

E.1. Código *web*

Esse código é o que contém o maior número de arquivos. Para o mesmo, temos dois arquivos auxiliares, o *funcoes.php* e o *constantes.php*. O primeiro contém as funções auxiliares e o segundo as constantes do sistema.

Os demais códigos são os formulários para entradas de dados e os códigos que os executam. Os formulários terminam sempre com a palavra *form* e os que o executam tem o mesmo nome, exceto dessa palavra. São eles:

- alterardadoscontatoform.php e alterardadoscontato.php; para alterar os dados de contato de um usuário.

- alterardadosenderecoform.php e alterardadosendereco.php; para alterar os dados de endereço de um usuário.

- alterardadosnomeform.php e alterardadosnome.php; para alterar o nome de um usuário.

- alterarsenhaform.php e alterarsenha.php; para alterar a senha de um usuário.

- alterarsenhaadmform.php e alterarsenhaadm.php; para o administrador alterar a senha de qualquer usuário.

- alterardadostipoform.php e alterardadostipo.php; para alterar o tipo de usuário de um usuário.

- atualizarreservas.php; para atualizar o status das reservas.

- cadalertaform.php e cadalerta.php; para cadastrar um alerta no sistema.

- cadautorform.php e cadautor.php; para cadastrar um autor no sistema.

- cadeditoraform.php e cadeditora.php; para cadastrar uma editora no sistema.

- cadlivroform.php e cadlivro.php; para cadastrar um livro no sistema.

- cadlivroautorform.php e cadlivroautor.php; para cadastrar uma relação livro autor no sistema.

- devlivroform.php e devlivro.php; para devolver um livro.

- edialertaform.php e edialerta.php; para editar um alerta no sistema.

- ediautorform.php e ediautor.php; para editar um autor no sistema.

- edieditoraform.php e edieditora.php; para editar uma editora no sistema.

- edilivroform.php e edilivro.php; para editar um livro no sistema.

- edilivroautorform.php e edilivroautor.php; para editar uma relação livro autor no sistema.

- ediusuario.php; para exibir um página de edição de um usuário.

- ediusuarioadmform.php; para exibir a página de ecolha do usuário a ser editado.

- emplivroform.php e emplivro.php; para emprestar um livro.

- emplivroresform.php e emplivrores.php; para emprestar um livro reservado.

- esquecisenhaform.php e esquecisenha.php; para gerar uma nova senha para um usuário.

- favlivro.php; para colocar um livro nos favoritos de um usuário.

- index.php; página inicial do sistema.

- login.php e logout.php; para autenticar e sair do sistema.

- menu.php, menuadm.php e menucliente.php; para exibir o menu do usuário.

- novoclienteform.php, novoclienteformfinal.php e novocliente.php; para cadastrar um novo usuário no sistema.

- procuraalertaform.php e procuraalerta.php; para buscar um alerta no sistema.

- procuralivroform.php e procuralivro.php; para buscar um livro no sistema.

- procurausuarioform.php e procurausuario.php; para buscar um usuário no sistema.

- remalertaform.php e remalerta.php; para remover um alerta do sistema.

- remautorform.php e remautor.php; para remover um autor do sistema.

- remeditoraform.php e remeditora.php; para remover uma editora do sistema.

- remfavlivro.php; para remover um livro do favoritos de um usuário.

- remlivroform.php e remlivro.php; para remover um livro do sistema.

- remlivroautorform.php e remlivroautor.php; para remover uma relação livro autor do sistema.

- remusuarioform.php e remusuario.php; para remover um usuário do sistema.

- reslivro.php; para reservar um livro.

- veralerta.php; para ver um alerta.

- veremprestimoadmform.php, veremprestimoadm.php e veremprestimos.php; para ver os empréstimos de um usuário do sistema.

- verfavoritosadmform.php, verfavoritosadm.php e verfavoriotos.php; para ver os favoritos de um usuário do sistema.

- verreservasadmform.php, verreservasadm.php e verreservas.php; para ver as reservas de um usuário do sistema.

- verlivro.php, verlivroadm.php e verlivrocliente.php; para ver um livro.

- verusuario.php; para ver um usuário.

- enviaremail.php e enviarsms.php; para enviar os alertas do sistema.

E.2. Código de agendamento

Para realizar o agendamento dos alertas, foram criados diversos Shell Scripts. Esses scripts podem ser divididos em três grupos. Um que monitora o horário dos alertas agendados, para que no horário especificado o mesmo possa ser enviado e dois que são responsáveis pelo envio dos mesmo, sendo um para o *sms* e o outro pra o *email.*

Para a monitoração do horário dos alertas temos o *alert_sch,* o *alert_script*, o *alert_sch_start* e o *alert_sch_stop*, sendo cada um desses, respectivamente, responsáveis por criar o serviço em uma máquina linux, executar a monitoração propriamente dita, inicializar o processo e finalizar o processo.

Para o envio dos *sms's* temos o *alert_sms*, o *sms_script*, o *alert_sms_start* e o *alert_sms_stop*, sendo cada um desses, respectivamente, responsáveis por criar o

serviço em uma máquina linux, executar o envio propriamente dito, inicializar o processo e finalizar o processo.

Para o envio dos *emails* temos o *alert_email*, o *email_script*, o *alert_email_start* e o *alert_email_stop*, sendo cada um desses, respectivamente, responsáveis por criar o serviço em uma máquina linux, executar o envio propriamente dito, inicializar o processo e finalizar o processo.

E.3. Código dos envios dos alertas

Para o envio dos alertas temos apenas o código fonte do programa que envia o *email*. Esse é o arquivo *enviaemail.c*.

E.4. Código de criação do banco de dados

Cada uma das tabelas que podem ser vistas no modelo relacional tiveram seu código de criação feito pelo programa DBDesigner e foram unidas em único arquivo a ser processado via linha de comando para a criação do banco. Esse arquivo é o *SQL_BD_PF_Criar.sql*.

Uma vez criadas as tabelas, criamos os scripts de inicialização de dados para o sistema. São esses o *Script_Inicializar_Estado.sql*, *Script_Inicializar_Cidade.sql*, *Script_Inicializar_TipoAlerta.sql*, *Script_Inicializar_TipoTelefone.sql* e *Script_Inicializar_TipoUsuario*, responsáveis, respectivamente, por inicializar os estados, as cidades, os tipos de alertas, os tipos de telefone e, finalmente, os tipos de usuários.

Apêndice F - Módulo *SMS*

O módulo *sms* foi criado para garantir a viabilidade do projeto, visto que esse tipo de comunicação era uma das idéias centrais do mesmo.

Para conseguir enviar essas mensagens de celular, usamos um modem *gsm* e uma biblioteca especifica para essa finalidade. O *modem* usado é o G24 da *Motorola* e a biblioteca é a *gsmlib*.

Para garantir o funcionamento do modem foi apenas necessário plugar o mesmo na porta serial do computador e executar o comando *"ln –sf /dev/ttyS0 /dev/mobilephone"*. Esse comando cria um link entre o modem e o arquivo do linux que será usado pela biblioteca para enviar as mensagens.

Uma vez que a biblioteca esteja instalada corretamente na máquina, usamos o comando *gsmsendsms* para enviar as mensagens.

Desse modo, criamos uma página onde seriam colocadas as informações necessárias ao envio da mensagem, e codificamos para que o comando de envio de torpedo da biblioteca funcionasse conforme o desejado.

A interface do módulo pode ser observada na figura abaixo.

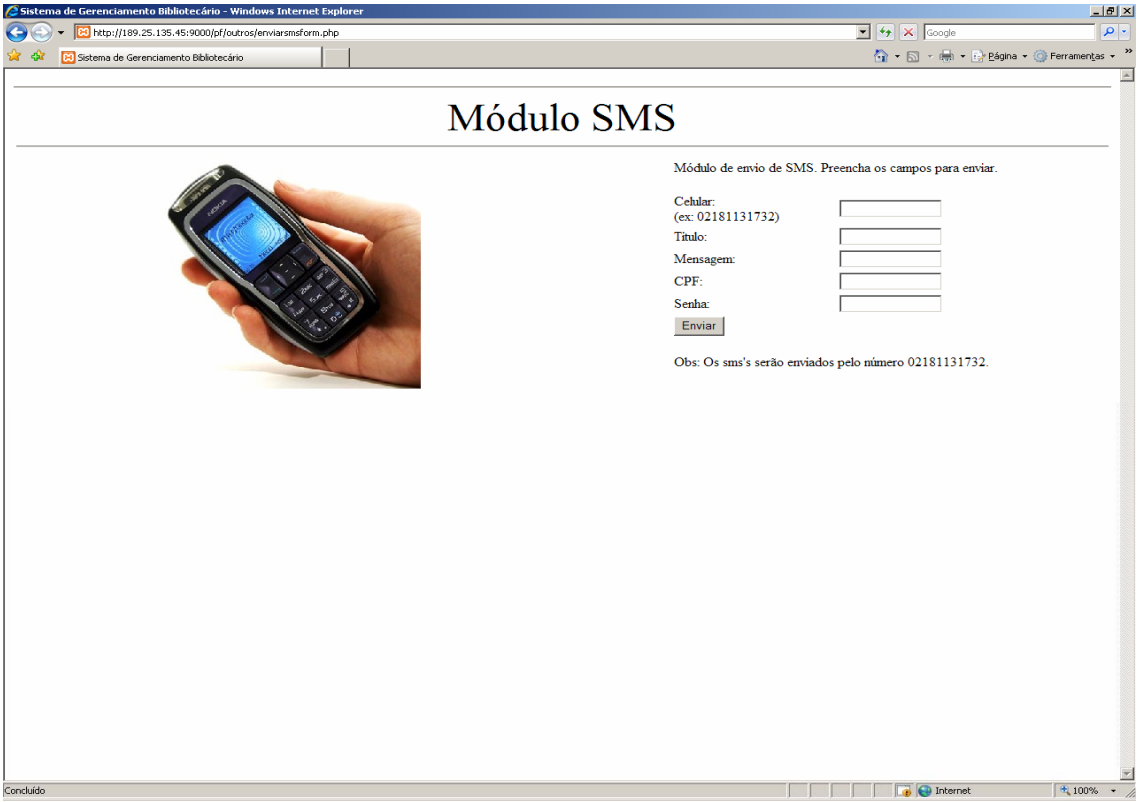

Figura F.1 – Módulo *SMS*

O usuário entra com os dados e é montado o comando para ser executado pelo sistema. Abaixo exibimos o código fonte de envio da mensagem.

```
### CÓDIGO ###
<?php
      include "/opt/lampp/htdocs/pf/include/funcoes.php";
      include "/opt/lampp/htdocs/pf/include/constantes.php";

      $cmd = "gsmsendsms";
      $tel = $_POST['cel'];
      $retaux = verificar_telefone($tel);
      if ($retaux != OK)
      {
            header("Location:enviarsmsform.php?ret=$retaux");
            exit;
      }
      $tit = $_POST['tit'];
      $msg = $_POST['msg'];
      $cpf = $_POST['cpf'];
      $senha = $_POST['senha'];
      $senhacript = md5($senha);
      $arg1 = escapeshellarg($tel);
      $arg2 = escapeshellarg($msg);
      $data = date("Y-m-d");
      $horario = date("Hi");
      $aux = autentica_usuario($cpf, $senhacript);
      if ($aux != OK)
      {
            header("Location:enviarsmsform.php?ret=$aux");
            exit;
      }
      else
      {
            $id = inserir_alerta_sms(ID_TIPOALERTA_SMS,
id_pelo_cpf($cpf), $tit, $msg, $data, $horario);
            system ($cmd." ".$arg1." ".$arg2, $ret);
            if ($ret == 0)
            {
                  inserir_envio($id, $data, $horario, "Modulo SMS: OK -
".$tel);
                  header("Location:enviarsmsform.php?ret=".OK);
                  exit;
            }
            else
            {
                  inserir_envio($id, $data, $horario, "Modulo SMS:
NOK");
                  header("Location:enviarsmsform.php?ret="
.MODULOSMS_ERRO);
                  exit;
            }
      }
?>
### FIM DO CÓDIGO ###
```

A parte chave do código é o comando *system* que chama o programa responsável pelo envio do *sms*.

Apêndice G - Módulo *EMAIL*

O módulo *email* foi criado para garantir uma das funcionalidades desejadas do projeto.

Para conseguir enviar essas mensagens eletrônicas, usamos um programa em C, que através da biblioteca *sockets*, se conectava ao provedor de *email yahoo* e, seguindo o protocolo smtp, enviava as mensagens.

Desse modo, criamos uma página onde seriam colocadas as informações necessárias ao envio da mensagem e codificamos para que o programa criado fosse chamado, passando essas informações.

A interface do módulo pode ser observada na figura abaixo.

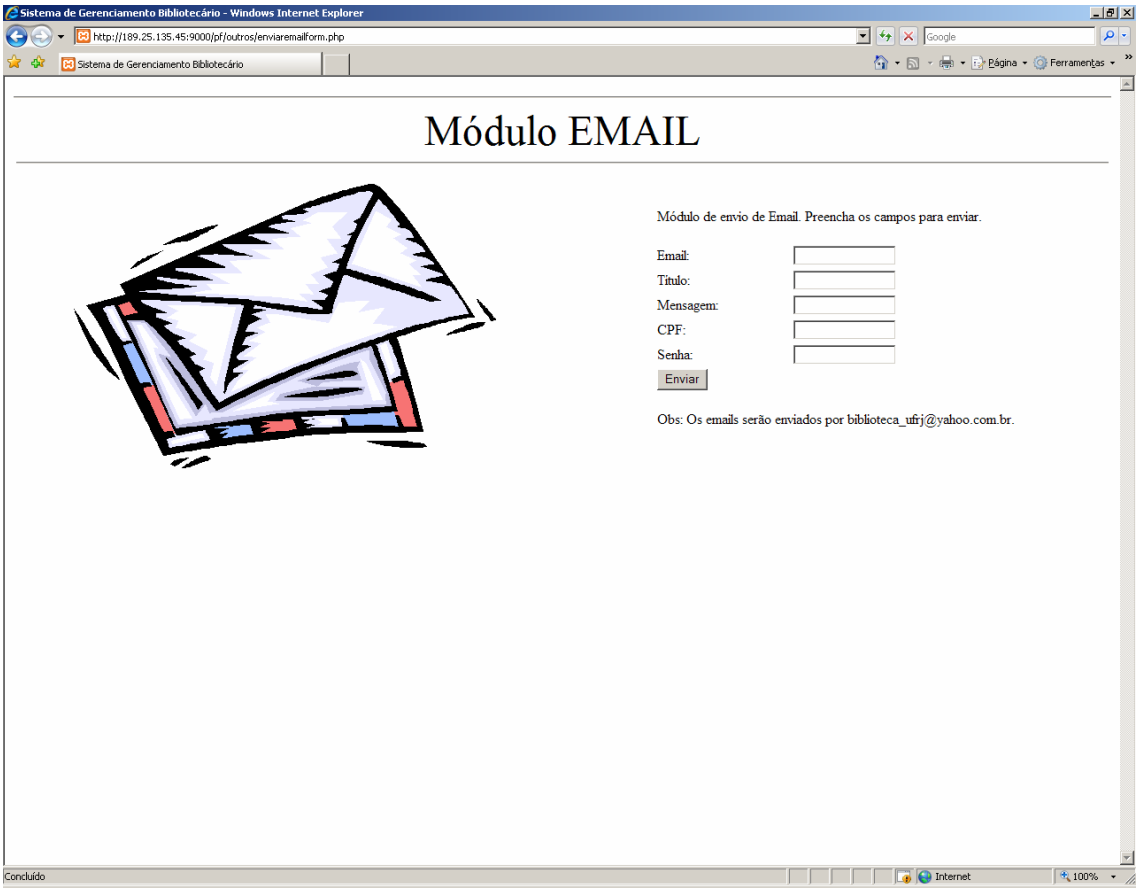

Figura G.1 – Módulo *EMAIL*

O usuário entra com os dados e é montado o comando para ser executado pelo sistema.

Na próxima página exibimos o código fonte de envio da mensagem.

```
### CÓDIGO ###
<?php
      include "/opt/lampp/htdocs/pf/include/funcoes.php";
      include "/opt/lampp/htdocs/pf/include/constantes.php";

      $cmd = "enviaemail";
      $des = $_POST['email'];
      $retaux = verificar_email($des);
      if ($retaux != OK)
      {
            header("Location:enviaremailform.php?ret=$retaux");
            exit;
      }
      $tit = $_POST['tit'];
      $msg = $_POST['msg'];
      $cpf = $_POST['cpf'];
      $senha = $_POST['senha'];
      $senhacript = md5($senha);
      $arg1 = escapeshellarg($des);
      $arg2 = escapeshellarg($tit);
      $arg3 = escapeshellarg($msg);
      $data = date("Y-m-d");
      $horario = date("Hi");
      $aux = autentica_usuario($cpf, $senhacript);
      if ($aux != OK)
      {
            header("Location:enviaremailform.php?ret=$aux");
            exit;
      }
      else
      {
            $id = inserir_alerta_email(ID_TIPOALERTA_EMAIL,
id_pelo_cpf($cpf), $tit, $msg, $data, $horario);
            system ($cmd." ".$arg1." ".$arg2." ".$arg3, $ret);
            if ($ret == 0)
            {
                  inserir_envio($id, $data, $horario, "Modulo EMAIL: OK
- ".$des);
                  header("Location:enviaremailform.php?ret=".OK);
                  exit;
            }
            else
            {
                  inserir_envio($id, $data, $horario, "Modulo EMAIL:
NOK");
                  header("Location:enviaremailform.php?ret="
.MODULOEMAIL_ERRO);
                  exit;
            }
      }
?>
### FIM DO CÓDIGO ###
```

A parte chave do código é o comando *system* que chama o programa responsável pelo envio do *email.*