UniCEUB - Centro Universitário de Brasília

FAET - Faculdade de Ciências Exatas e Tecnologia

Curso de Engenharia da Computação

Projeto Final

SISTEMA DE CONTROLE DE ACESSO INTEGRADO À WEB

Por

Arthur Araújo Oliveira

RA: 2006421/1

Professora Orientadora:

Prof.ª MC. Maria Marony Sousa Farias Nascimento

Brasília, DF - Junho de 2007.

UniCEUB - Centro Universitário de Brasília

FAET - Faculdade de Ciências Exatas e Tecnologia

Curso de Engenharia da Computação

Projeto Final

SISTEMA DE CONTROLE DE ACESSO INTEGRADO À WEB

Monografia apresentada à banca examinadora para conclusão do curso e obtenção do título de bacharel em Engenharia da Computação do Centro Universitário de Brasília - UniCEUB

Brasília, DF - Junho de 2007.

AGRADECIMENTOS

Em primeiro lugar, agradeço a Deus por me dar saúde e força para conquistar mais uma etapa da minha vida.

Ao meu pai Adailton Oliveira, minha mãe Maria de Fátima Araújo Oliveira, e meus irmãos, por me apoiarem em todos os momentos.

A todos os colegas de faculdade pelo apoio e incentivo. Em especial ao Júlio César, por passar seus conhecimentos e sabedoria durante essa caminhada.

A minha namorada Sabrina Aragão e amigos que sempre acreditaram em mim e entenderam todos os momentos em que estive ausente, vocês também fazem parte dessa conquista!

A minha professora orientadora MC. Maria Marony Sousa Farias Nascimento, pelo apoio.

E a todo corpo docente da instituição UniCeub, meu muito obrigado

“A mente que se abre a uma nova idéia jamais voltará ao seu tamanho original.”

(Albert Einstein)

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIAÇÕES E SIGLAS

API	Application Programming Interface
CW552	Kit de Desenvolvimento da ControlWare Automação
DAO	Data Access Object
DB-9	Conector Serial, padrão RS-232, com nove pinos
DB-25	Conector Serial, padrão RS-232, com vinte e cinco pinos
EPROM	Erasable Programmable Read-only Memory
HTML	Hyper Text Markup Language
HTTP	Hipertext Markup Language
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
JVM	Java Virtual Machine
JSP	Java Server Pages
MVC	Model View Controller
PDA	Personal Digital Assistant
PHP	Hypertext Preprocessor
RAM	Random Access Memory
RS-232	Padrão da Interface Serial
SCA	Sistema de Controle de Acesso
SDCC	Small Device C Compiler
SGBD	Sistema Gerenciador de Banco de Dados
SQL	Structured Query Language
TCP/IP	Transmission Control Protocol / Internet Protocol
UART	Universal Assynchronous Receiver and Transmiter
UML	Unified Modeling Language
URL	Universal Resource Locator

RESUMO

Neste trabalho foi desenvolvido um protótipo para o monitoramento contínuo de funcionários/usuários, através de um controle de acesso, utilizando um sistema web. No projeto é utilizada a base de dados já existente na empresa. No hardware desenvolvido são utilizados um leitor de código de barras, um teclado alfanumérico e o microcontrolador da família 8051. O software foi desenvolvido utilizando as linguagens de programação C e Java, um banco de dados MYSQL para validação do usuário e interface serial no padrão RS-232 para comunicação entre o protótipo e o computador. Para disponibilizar as informações obtidas pelo software é utilizado o servidor Tomcat, utilizando JSP para implementação das páginas com Java.

Palavras-chave: Controle de Acesso, Sistema WEB, Microcontrolador, Linguagens de programação Java e C.

ABSTRACT

In this work an archetype for the continuous supervision of officers/users was developed, through an access control, using a system web. In the project the existing database already in the company is used. In the developed hardware they are used the reader of bar code, an alphanumeric keyboard and the microcontroller of family 8051. Software was developed using the programming languages C and Java, a data base MYSQL for validation of the user and serial interface in standard RS-232 for communication between the archetype and the computer. To service the information gotten for software Tomcat, using JSP for implementation of the pages with Java is used as serving.

Word-key: Control of Access, System WEB, Microcontroller, Programming languages Java and C.

SUMÁRIO

CAPÍTULO 1 – INTRODUÇÃO

1.1 Motivação

Como o próprio nome sugere, sistema de controle de acesso são tecnologias desenvolvidas para controlar o acesso de pessoas em áreas restritas, aumentando significativamente a segurança do local. Existem várias maneiras de realizar este controle, como: smart card, biometria, cartão magnético, código de barras e teclado. Os dispositivos utilizados neste projeto para identificação dos funcionários/usuários foram o teclado e o código de barras, instrumento estes escolhidos por possuírem um baixo custo em relação aos demais métodos de controle.

Tradicionalmente os sistemas de controle de acesso são desenvolvidos em linguagens que necessitam ser configuradas na máquina do cliente, ou seja, caso o usuário queira acessar o sistema de qualquer computador de sua rede privada, será necessário fazer a instalação do mesmo em cada computador.

Visando um melhor monitoramento desses funcionários e utilizando a portabilidade de um sistema web, foi desenvolvido um protótipo de acesso que realiza o controle dos funcionários e um sistema web para monitorar o acesso dos mesmos. O sistema poderá ser acessado de qualquer computador da rede que possua um browser para sua visualização.

O protótipo de acesso e o sistema web utilizam uma base de dados já cadastrada e atualizada por um sistema existente na empresa, ou seja, esta base contém todas as informações necessárias para o monitoramento proposto.

Após uma revisão Bibliográfica, onde foram estudadas as diversas tecnologias e ferramentas existentes no mercado, optou-se então por utilizar:

- Banco de dados (MYSQL);
- Linguagem de Programação Java e C;
- UML;
- Microcontroladores;
- Eclipse;
- Servidor Web – Tomcat;
- Framework – Struts;

Foram utilizados os seguintes componentes:

- Teclado alfanumérico;
- Leitor de código de barras;
- Conversor USB/Serial;
- Kit CW552 composto por:
    1. Microcontrolador 80C552,
    2. “Display” Alfanumérico,
    3. Saída padrão RS232;

O projeto desenvolvido é a integração dos componentes citados anteriormente e tem como principal objetivo apresentar o protótipo de acesso e através de um sistema web interagir com as principais características de um controle de ponto: bloquear e liberar os cartões de identificação, alterar e visualizar os funcionários e gerar relatórios para uma melhor fiscalização e controle. Ressaltando que o sistema da empresa é o responsável por gerar o código de barras, logo será usado no projeto um código de barras já gerado pelo mesmo.

Na Figura 1.1, mostrada a seguir, é demonstrada uma visão geral do funcionamento do protótipo desenvolvido.

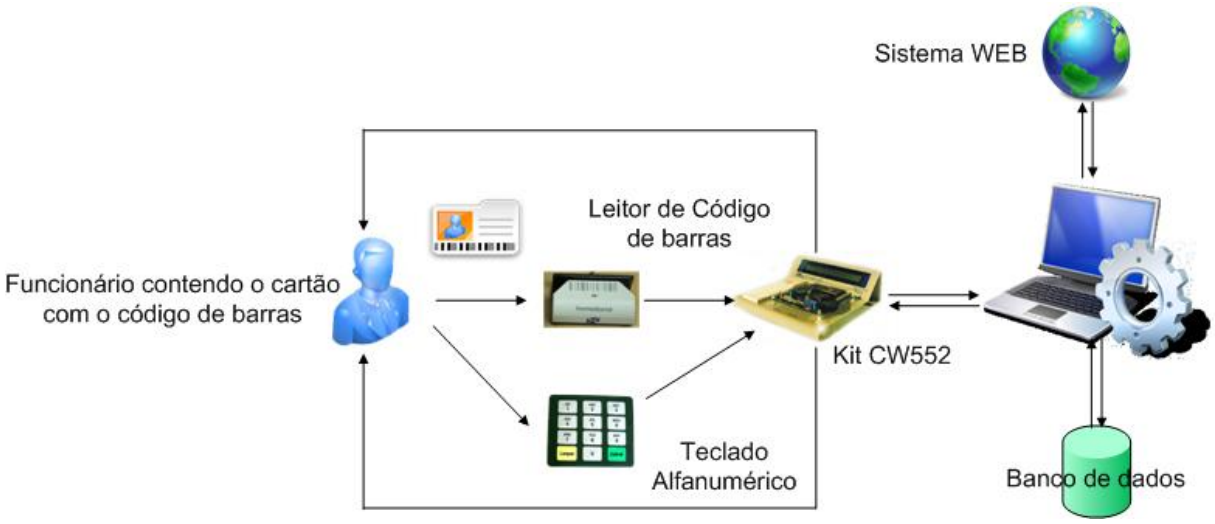

Figura 1.1 – Fluxo do Projeto Desenvolvido

O funcionário possuirá um cartão contendo um código de barras, que será sua chave de identificação no sistema. Após passar o seu cartão e o mesmo for validado, o funcionário terá que digitar sua senha para que o acesso seja liberado. Sendo ambos validados, o funcionário estará autorizado a entrar no ambiente monitorado. No sistema será registrada a data, a hora e o local acessado.

1.2 Estrutura do Trabalho

O trabalho está organizado em cinco capítulos. Os primeiros três capítulos fazem a apresentação do tema do projeto, fornecem o embasamento teórico do trabalho e as tecnologias utilizadas. No capítulo seguinte é apresentado o software e protótipo de acesso desenvolvido e o último capítulo traz as considerações finais e propostas futuras. A organização detalhada é descrita a seguir:

Capítulo 2: são apresentados os aspectos básicos do sistema, tais como: banco de dados, Framework Struts, linguagem de programação Java e outros.

Capítulo 3: é descrito o hardware utilizado, o Kit CW552, a comunicação serial e interrupções utilizadas.

Capítulo 4: são apresentados o sistema web e protótipo de acesso desenvolvido, modelagem de dados, arquitetura, telas e fluxogramas do sistema.

Capítulo 5: são apresentadas as conclusões, resultados, dificuldades encontradas no projeto e as sugestões de trabalhos futuros.

CAPÍTULO 2 – FUNDAMENTAÇÕES TEÓRICAS

2.1 Sistemas WEB

Sistemas web são aplicações desenvolvidas para Internet ou rede privada (intranet) que utilizam o *browser* para sua visualização. A aplicação é executada em um ambiente distribuído, no qual cada parte que compõe o programa pode estar localizada em uma máquina diferente.

Tradicionalmente, essas aplicações são um conjunto de páginas HTML com códigos em outras linguagens como PHP ou Java, permitindo criar aplicações muito dinâmicas. Entretanto, é um engano pensar que essas aplicações não trazem consigo certa complexidade ao se definir seu escopo (Mundo Java, 2007). Deve ser levado em conta um conjunto de atributos:

- **Usabilidade:** Deve permitir a acessibilidade do sistema. O sistema deve obter uma resposta rápida e informação de qualidade, o usuário deve obter com facilidade o conteúdo de seu interesse, caso contrário irá à procura de um outro sistema.
- **Funcionalidade:** O sistema WEB deve ter capacidade de busca e recuperação de informações/links/funções, aspectos navegacionais e outros relacionados ao domínio da aplicação.
- **Confiabilidade:** garantir o funcionamento do sistema, o processamento correto de links, a validação de usuários, considerando a presença de falhas.
- **Manutenibilidade:** O sistema WEB deve ser fácil de manutenção.

2.2 Arquitetura Cliente / Servidor

Em uma aplicação cliente-servidor, o cliente se conecta a um servidor de aplicação ou sistema de base de dados, conforme ilustrado no Figura 2.1. Neste tipo de arquitetura os aplicativos ficam separados fisicamente do SGBD, isso ocorre porque este tipo de projeto parte do seguinte pressuposto: computadores diferentes executam tarefas diferentes. Dessa forma, nós temos vários aplicativos que fazem

vários acessos e solicitações ao banco de dados e o SGBD é o responsável pelo recebimento das requisições e devolução dos resultados, que são os servidores de banco de dados. No entanto, para aplicações complexas que envolvem uma grande interação com banco de dados, o modelo pode ocasionar sobrecarga na estação cliente e na rede. Outro problema é que a máquina SERVIDOR deve ser robusta, além do sistema ter dependência total do servidor (Linha de Código, 2007).

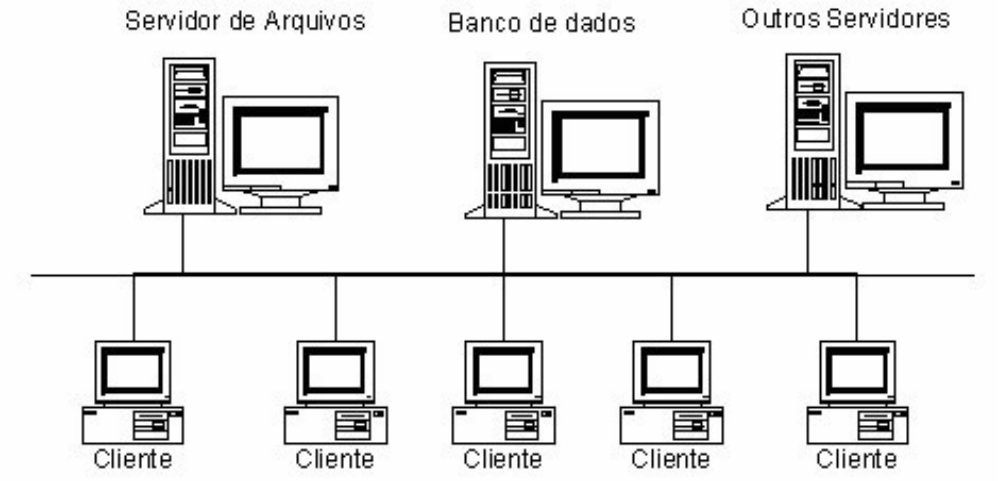

Figura 2.1 – Modelo Cliente/Servidor tradicional

2.3 A Linguagem Java

Anunciada a partir de 1995 pela Sun Microsystens, o Java surgiu não apenas como uma linguagem de programação, mas como uma plataforma de desenvolvimento capaz de proporcionar grandes aplicações. A linguagem Java não pára de crescer, atualmente ele já se encontra presente em celulares, *pagers*, PDAs e, se encontra em constante evolução, pois toda semana surge algo novo sobre o Java na página da Sun (Furgeri, 2002).

A principal vantagem do Java é ser uma linguagem multiplataforma, um fato importantíssimo, pois os programadores não necessitam ficar preocupados em saber em qual máquina o programa será executado, ou seja, um mesmo programa será executado em diversas plataformas, isso é possível, porque cada plataforma possui sua máquina virtual (JVM) e ela é responsável pelo processo de compilação. Após ser compilados as classes são gerados bytecodes, que são específicos a qualquer máquina física. Outros aspectos que caracterizam a linguagem são:

orientação a objetos, que é hoje universalmente adotada como padrão de mercado, portabilidade, suporte a comunicação em rede e acesso remoto a banco de dados. Uma desvantagem é que, por ser uma linguagem interpretada, necessita de mais processamento e torna-se mais lenta. Na Figura 2.2 é ilustrada a seqüência de um programa em Java (Furgeri, 2002).

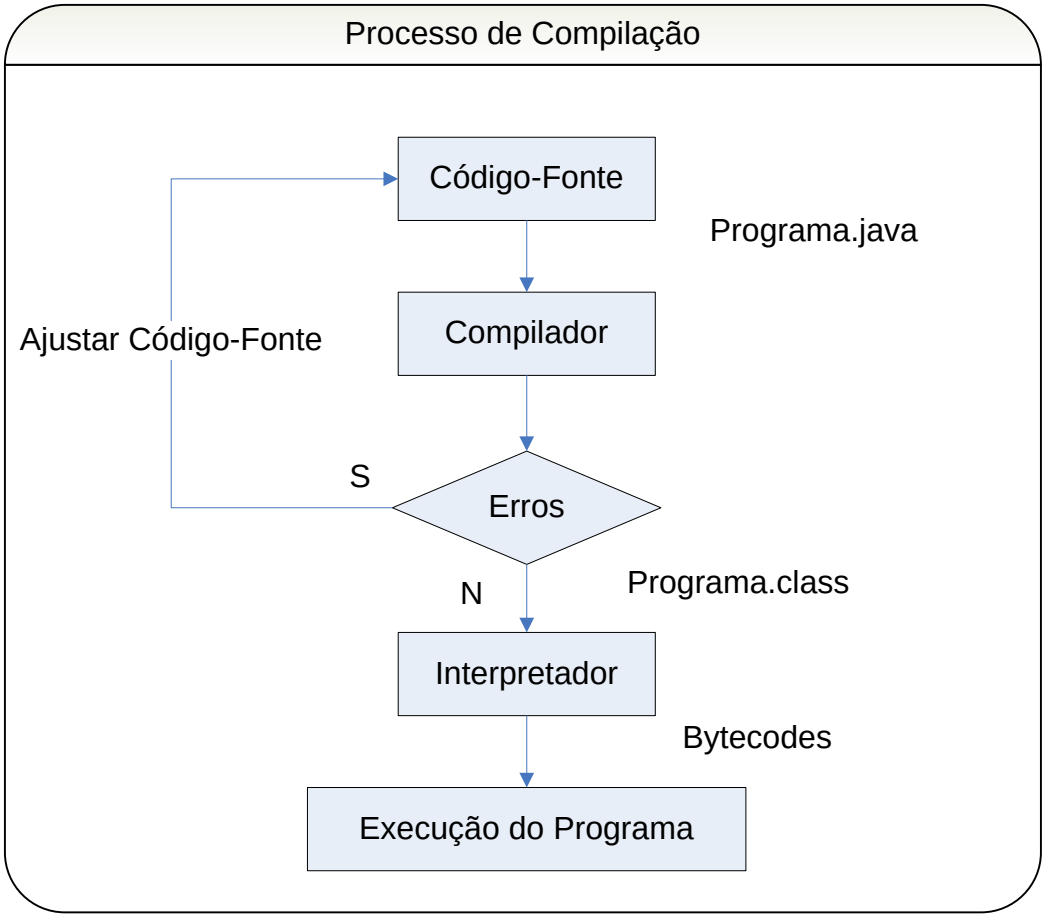

Figura 2.2 – Seqüência de desenvolvimento de um programa Java

2.3.1 Ambiente de Desenvolvimento

Como quaisquer outras linguagens, existem muitas opções e diversas maneiras de desenvolver aplicações em Java. A IDE escolhida para o desenvolvimento da aplicação web foi o Eclipse. Esta é uma ferramenta gratuita desenvolvida pela IBM, que integra a tecnologia de plugins, onde se permite personalizar o ambiente de desenvolvimento, seja ele, um simples projeto, até

aplicações robustas. Plugins são programas integrados aos browsers e ferramentas que desempenham uma tarefa específica (som, vídeo, etc.) que podem ser acoplados de acordo com a necessidade do projeto (Eclipse, 2006). Além do Java, é possível configurar o Eclipse em outras linguagens de programação como C++ , C# ou PHP, basta verificar os plugins existentes para a linguagem desejada.

2.3.2 Java Communications API

Para facilitar o desenvolvimento de softwares são criados conjuntos de rotinas e padrões estabelecidos de forma a facilitar a utilização de suas funcionalidades por outros programas e aplicativos sem que aja um conhecimento aprofundado sobre a implementação do software. Somente usam seus serviços. Esses padrões são chamados de API. O protótipo de acesso foi desenvolvido utilizando um desses serviços, especificamente a API de comunicação com as portas paralela e serial.

Neste projeto foi utilizada a API de comunicação do Java (Java Communications), fornecida gratuitamente pela Sun Microsystens no endereço *http://java.sun.com/products/javacomm/index.jsp*, que permite a comunicação com a interface serial e paralela nos padrões RS-232 e IEEE-1284. Após baixar a API, descompactá-la, é necessário configurar o ambiente de desenvolvimento. São especificamente três arquivos necessários: win32com.dll, javax.comm.properties e comm.jar. O arquivo ”*win32com.dll*” permite acesso direto às portas do PC. E esta DLL emula uma interface para acessar diretamente os recursos das portas, sem ter que preocupar com as permissões do sistema operacional. O arquivo “*javax.comm.properties*” especifica o tipo de driver que será usado para se acessar os recursos das portas do computador. No caso do projeto, o driver utilizado foi *com.sun.comm.Win32Driver,* específico do Sistema Operacional Windows. Por último, o arquivo “*comm.jar* ” contém as classes Java para o acesso às portas de comunicação (Sun Microsystens, 2007). Todo o processo de instalação do Java Communications API encontra-se no Apêndice A.

2.4 UML

A UML é uma linguagem padronizada para modelagem de sistemas de software orientados a objetos, sendo adotada pela indústria de software e também

por fornecedores de ferramentas CASE (Computer-Aided Software Engineering) como linguagem padrão.

A UML é utilizada para documentar, visualizar, especificar e construir sistemas de software orientados a objetos. Pode ser aplicada a qualquer tipo de sistemas de software, podendo também ser aplicada para modelar processos de trabalho, projetos de hardware, dentre outros (Guedes, 2005).

Um diagrama UML é uma representação gráfica de um conjunto de elementos do sistema. Esta linguagem disponibiliza diagramas que permitem representar diferentes partes do modelo de um sistema (Guedes, 2005). A seguir tem-se a descrição de cada um destes diagramas:

- **Diagrama de caso de uso:** representam um conjunto de atores, casos de uso e os relacionamentos entre eles;
- **Diagrama de interação:** representam colaborações entre objetos, para realizarem algum tipo de comportamento para um sistema;
- **Diagrama de seqüência:** dá ênfase à ordenação temporal em que as mensagens são trocadas entre os objetos;
- **Diagrama de colaboração:** dá ênfase à ordenação estrutural em que as mensagens são trocadas entre os objetos de um sistema;
- **Diagrama de gráficos e estados:** representa os estados possíveis de um objeto em particular;
- **Diagrama de atividades:** representa a modelagem do fluxo de controle de uma atividade para outra no sistema;
- **Diagrama de classe:** representa a estrutura de um sistema (esqueleto);
- **Diagrama de componentes:** representa um conjunto de componentes e suas respectivas dependências;
- **Diagrama de objetos:** representa as instâncias das classes de um sistema.
- **Diagrama de implantação:** representa a configuração e a arquitetura de um sistema, ligando seus respectivos componentes.

No projeto, foram utilizados os diagramas de classes, componentes e implantação, estes usados para demonstrar a estrutura do projeto, todas as classes criadas no sistema web e explicar a função de cada componente dentro da arquitetura utilizada.

2.5 Servidor Apache Tomcat

O Tomcat é um software livre e de código aberto surgido dentro do conceituado projeto Apache Jakarta e permite a execução de aplicações para web. A principal característica técnica é a implementação das tecnologias baseadas na linguagem de programação Java, mais especificamente nas tecnologias de Servlets e JSP (Portal Java, 2006).

Tecnicamente, o Tomcat é um Container Web, que tem a capacidade de atuar também como servidor web/HTTP, ou seja, robusto e eficiente o suficiente para ser utilizado em um ambiente de produção (Portal Java, 2006).

Do ponto de vista operacional, a principal finalidade das tecnologias de Servlets e JSP é permitir a criação dinâmica de conteúdos conforme a seqüência de passos descritos a seguir:

- Um usuário, no seu browser, solicita algum documento (indicado por um URL) a um servidor Tomcat;
- O servidor, ao receber uma solicitação (URL) do usuário, executa o Servlet ou JSP correspondente àquela URL (a associação entre URL e Servlet ou JSP é especificada no arquivo WEB.xml). O conteúdo gerado pelo Servlet ou JSP, normalmente um documento no formato HTML, é uma combinação de tags HTML (incluídos explicitamente) e o resultado de algum processamento (por exemplo, algoritmo Java e/ou acesso a um banco de dados).
- O usuário recebe o conteúdo gerado pelo servidor Tomcat e o exibe através do seu browser.

2.6 Framework Struts

O Framework Struts foi criado por Craig R. McClanahan e doado para a ASF (*Apache Software Fundation*) em 2000. O nome Struts deve-se ao papel por ele desenvolvido nas aplicações WEB. Da mesma forma que a construção de uma casa, ponte, prédio necessitam de uma base mantenedora, os engenheiros de software

usam o Struts para suportar cada camada de uma aplicação comercial (Java Free, 2006).

Os problemas com modelos de aplicações não são novidades. É possível se deparar com alguns cenários onde o problema acaba sendo o modelo Cliente/Servidor. Os acessos aos dados diretamente nas páginas, à subutilização de componente, sobrecarga de processamento e constantes "travamentos" da aplicação são alguns deles.

Uma aplicação que utiliza o Struts é composta por várias tecnologias e *design-patterns*, ele não é uma tecnologia específica, mas sim um conjunto destas que possibilita o desenvolvimento de aplicações WEB utilizando-se o modelo MVC (Model-View-Controller). Este modelo é um padrão que separa de maneira independente a aplicação, representa os objetos de negócio (Model) da camada de apresentação, representa a interface com o usuário ou outro sistema (View) e o Controle de fluxo da aplicação (Controller) (Java Free, 2006).

2.6.1 Detalhes do Funcionamento

Na Figura 2.3 é representado o fluxo normal de uma aplicação Struts.

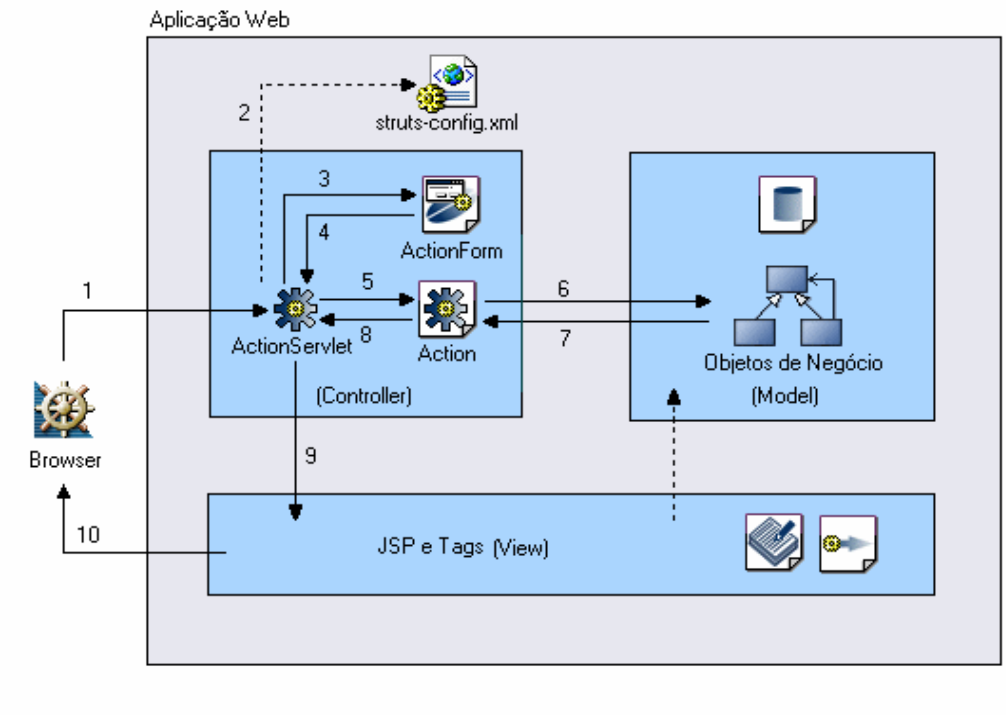

Figura 2.3 – Fluxo de uma aplicação Struts.

1. O usuário faz uma solicitação através de uma URL no browser. Ex: *http:///localhost:8080/sca/jsp/login/exibirInsercao.do*. Note que, no final da URL, ha um “.do“ que será usado para mapear o servlet controller da struts.
2. Se for a primeira solicitação que o container recebeu para esta aplicação, ele irá mapear o método init da ActionServlet (controller da Struts) e irá carregar as configurações do arquivo struts-config.xml em estruturas de dados na memória. Vale lembrar que esta passagem só será executada uma única vez, pois, nas solicitações subseqüentes, a servlet consulta estas estruturas na memória para decidir o fluxo a ser seguido.
3. Baseado no fluxo definido no arquivo struts-config.xml, que neste momento já se encontra carregado em estruturas na memória, o ActionSerlet identificará qual o ActionForm (classe para a validação dos dados) irá mapear. A classe ActionForm por meio do método validate irá verificar a integridade dos dados que foram recebidos na solicitação que vem do browser.
4. O controle da aplicação é retomado pelo ActionServlet, que analisa o resultado da verificação do ActionForm.
5. O ActionServlet, baseado no fluxo da aplicação (estruturas já carregadas em memória), mapeia uma classe Action. A classe Action passará pelo método execute que irá delegar a requisição para a camada de negócio.
6. A camada de negócio irá executar algum processo (geralmente um model). O resultado da execução deste processo (objetos já populados) será usado na camada de apresentação para exibir os dados.
7. Quando o controle do fluxo da aplicação voltar ao Action que invocou o processo da camada de negócio, será analisado o resultado e definido qual o mapa adotado para o fluxo da aplicação. Neste ponto, os objetos que foram populados na camada de negócio serão "atachados" como atributos na seção do usuário.
8. Baseado no mapeamento feito pelo Action, o Controller faz um forward para o JSP para apresentar os dados.
9. Na camada de apresentação (View), os objetos que foram setados como atributos da sessão do usuário serão consultados para montar o html para o browser.
10. Chega o html da resposta requisitada pelo usuário.

2.7 Banco de Dados - MYSQL

O MYSQL é um sistema de gerenciamento de banco de dados com o objetivo de se criar uma base de dados relacional capaz de equilibrar estabilidade, segurança, rapidez e baixo custo. Ele utiliza a linguagem SQL (Linguagem de Consulta Estruturada) como interface, esta linguagem permite a definição da estrutura de dados, consulta e modificação de registros, bem como, especificação de restrições de segurança (Manzano, 2007). As principais características do MYSQL são:

- Portabilidade (suporta praticamente qualquer plataforma atual);
- Compatibilidade (existem drivers ODBC,JDBC e .NET e módulos de interface para diversas linguagens de programação, como Delphi, Java, C/C++ e outras);
- Pouco exigente quanto a recursos de hardware;
- Software Livre.

CAPÍTULO 3 – HARDWARE PROPOSTO

3.1 Kit de Desenvolvimento CW552

No desenvolvimento do protótipo de acesso foi utilizado o Kit de desenvolvimento CW552, uma ferramenta de desenvolvimento simplificada e completa para sistemas baseados em microcontroladores (ELS, 2001). O Kit CW552 é representado pela Figura 3.1.

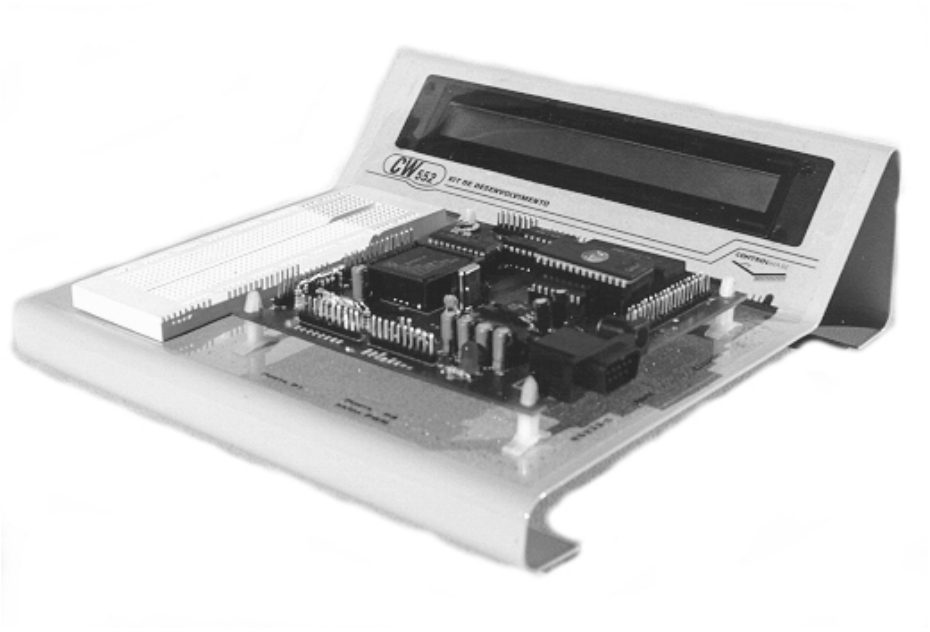

Figura 3.1 – Kit de Desenvolvimento CW552

3.1.1 Apresentação

O Kit CW552 possui um microcontrolador 80c552, no qual fornece um ambiente de desenvolvimento integrado e compatível com a família do microcontrolador 8051. Ele é composto por uma base de suporte, uma fonte de alimentação, um display de cristal líquido, uma interface serial RS-232 e uma placa de circuito impresso com o microcontrolador. Sua comunicação com o computador é realizada através de um cabo serial no padrão RS-232 com a ligação cruzada (também conhecido como cross-over) (ELS, 2001). Todos os pinos de entrada e saída são facilmente acessíveis como mostrado no layout da placa na Figura 3.2.

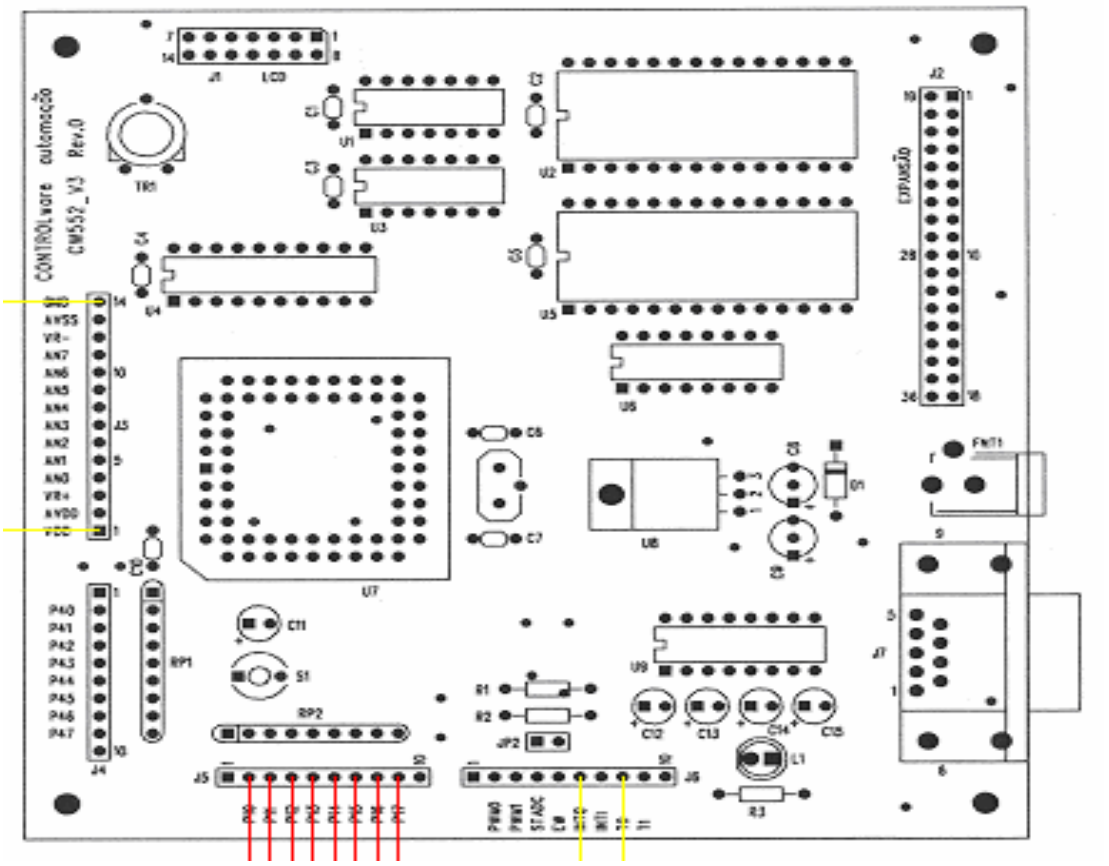

Figura 3.2 – Layout do Kit CW552

Pinos utilizados pelo teclado alfa numérico.

Pinos utilizados pelo leitor de código de barras.

3.1.2 Compilador SDCC

O Compilador SDCC (Small Device C Compiler) – é um compilador de domínio público voltado a programação de microcontroladores que trabalham com palavras de 8 bits. Sua função é traduzir o código criado em C para uma linguagem de montagem suportada pelo processador e fornecida pelo fabricante. Dentre as famílias suportadas por ele temos o 8051, utilizado no projeto (Nicolosi, 2005).

O SDCC tem um papel importantíssimo dentro do projeto, ele é responsável por compilar o código em C e gerar um arquivo chamado de Intel-Hex, cuja finalidade será descrita no item 3.1.3.

3.1.3 Execução do Programa

A função do microcomputador neste projeto é tratar os dados enviados e recebidos do sistema. A execução de um programa no CW552 é realizada da seguinte forma: em primeiro lugar, é desenvolvido um programa em C, após a escrita do programa é feita à compilação e geração de um arquivo Intel-Hex, utilizando o compilador SDCC e gravada na memória EPROM, o arquivo compilado é carregado na memória RAM do CW552, para finalmente ser executado. Na Figura 3.3 mostra o fluxo da execução do programa.

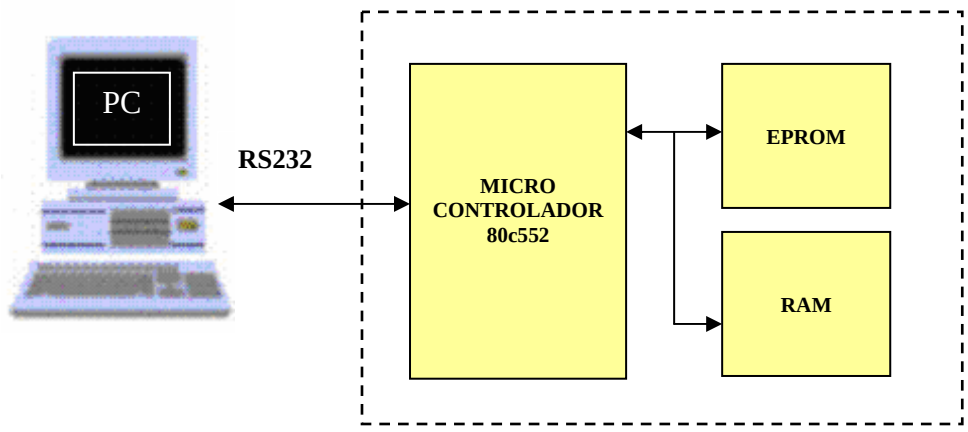

Figura 3.3 – Fluxo da execução do programa

3.2 Interface Serial

A comunicação serial ou interface serial é uma porta de comunicação utilizada para se conectar a outros equipamentos de hardware. Ela se encontra presente nos computadores desde o inicio da década de 80 e se caracteriza por transmitir e receber um bit de cada vez em seqüência preestabelecida e pré-prolongada (Vasconcelos, 2002). No 80C552, utilizado no projeto, temos uma interface serial do tipo *full duplex*, desta forma, podemos enviar e receber dados de forma simultânea com o microcontrolador.

O padrão utilizado no projeto é o RS-232 uma interface do tipo UART (Universal Assynchronous Receiver and Transmiter) que opera no modo assíncrono, o que indica ao receptor que será enviado um dado, ou seja, a transição de 1 para 0 indica o inicio da transmissão. Este padrão ainda especifica dois conectores um com 9 pinos (DB-9) e outro com 25 pinos (DB-25) conforme mostrado na Figura 3.4 (Vasconcelos, 2002).

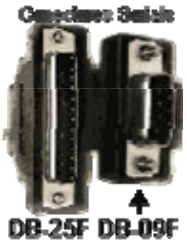

Figura 3.4 – Conectores RS-232

3.2.1 Formato dos Dados transmitidos

Os dados transmitidos pela porta serial funcionam de acordo com o diagrama mostrado na Figura 3.5. Quando está em repouso ele fornece uma tensão correspondente ao bit 1, ao iniciar a transmissão dos dados é acionado o primeiro bit 0, chamado de start bit, os dados então começam a ser transmitidos, um bit de cada vez, podem ser transmitidos 5 ,6, 7 ou 8 bits dependendo de como a UART esteja programada. Terminados os bits de dados, é enviado um bit opcional de paridade, e por ultimo, um bit finalizador chamado de stop bit, que tem sempre o valor 1. Imediatamente após o stop bit, pode ser enviado o start bit do dado seguinte (Vasconcelos, 2002).

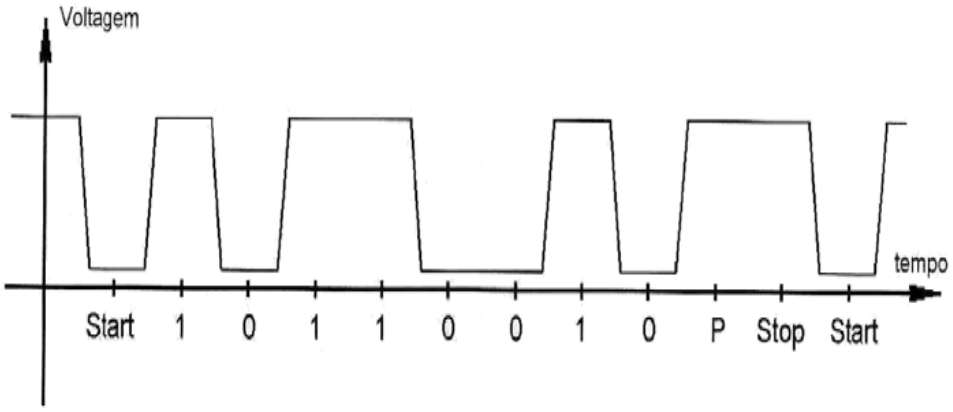

Figura 3.5 – Transmissão de uma porta serial

3.2.2 Taxa de Transmissão – Baud Rate

O *baud rate* é uma medida com dimensão de bits/segundo, mas não representa exatamente o número de bits de dados transmitidos a cada segundo pela interface. Sua unidade é o *baud*, e seu valor é o inverso do período de transmissão de 1 bit. Por exemplo, se tomarmos um *baud rate* de 4.800 *bauds*, significa que cada bit é transmitido em um tempo de:

1s / 4800 = 0,00020833 *s*

Ou seja, 208, 33 μs. O cálculo da taxa de transmissão do projeto foi feito utilizando a Fórmula (1), disponibilizada no *datasheet* do CW552 (ELS, 2001).

$$Baud\ Rate = \frac{2}{64} \times \frac{11.059.000}{12 \times (256 - TH1)} \qquad (1)$$

Esta fórmula foi desenvolvida para cálculos de *baud rate* de microcontroladores com cristal de 11,059 MHz. A taxa de transmissão utilizada para fazer os testes foi a de 9.600 bps, e para isso o valor de TH1 ficou sendo FD (hexadecimal) (ELS, 2001).

3.2.3 Comunicação Serial do Projeto

A transmissão serial do projeto utilizou os pinos de transmissão de dados (TX), pino 3, e recepção de dados (RX) pino 2. O aterramento, caso ocorra nível de tensão acima de 5 volts é feito pelo pino 5 (GND). Na Figura 3.6 é mostrada a configuração do conector utilizado para efetivar a comunicação dos dados.

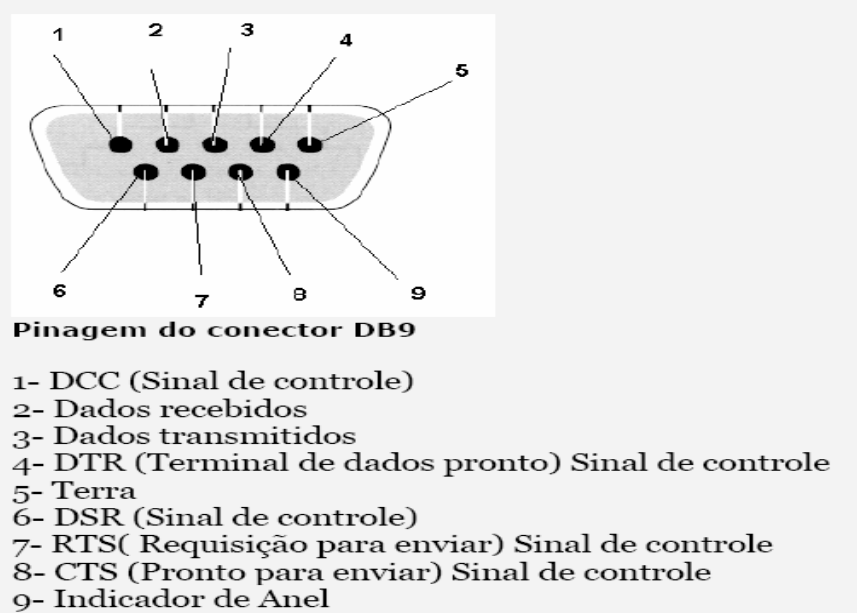

Figura 3.6 – Configuração do conector serial DB-9

Para realizar a comunicação serial com o Java, foi preciso configurará-lo conforme item 2.4.2. Além desta configuração, para reconhecer a porta serial, durante o desenvolvimento é necessário especificar todos os parâmetros de comunicação com a porta, como: qual porta ele fará a comunicação (COM1, COM 2), o *baud raute* (taxa de transmissão), o *Data bits*, o *Stop bits* e por último o *bit de paridade.* A Java Communications API está composta por uma classe principal *CommPort* e duas subclasses, *SerialPort* e *ParallelPort*, que permitem fazer os dois tipos de comunicação, como demonstrado na Figura 3.7. O método criado para configurar a porta de comunicação é chamado de ***AbrirPorta()***, e nele é implementado todas as configurações necessárias para iniciar a comunicação. Os códigos referentes à comunicação serial serão apresentados e explicados no Capítulo 4.

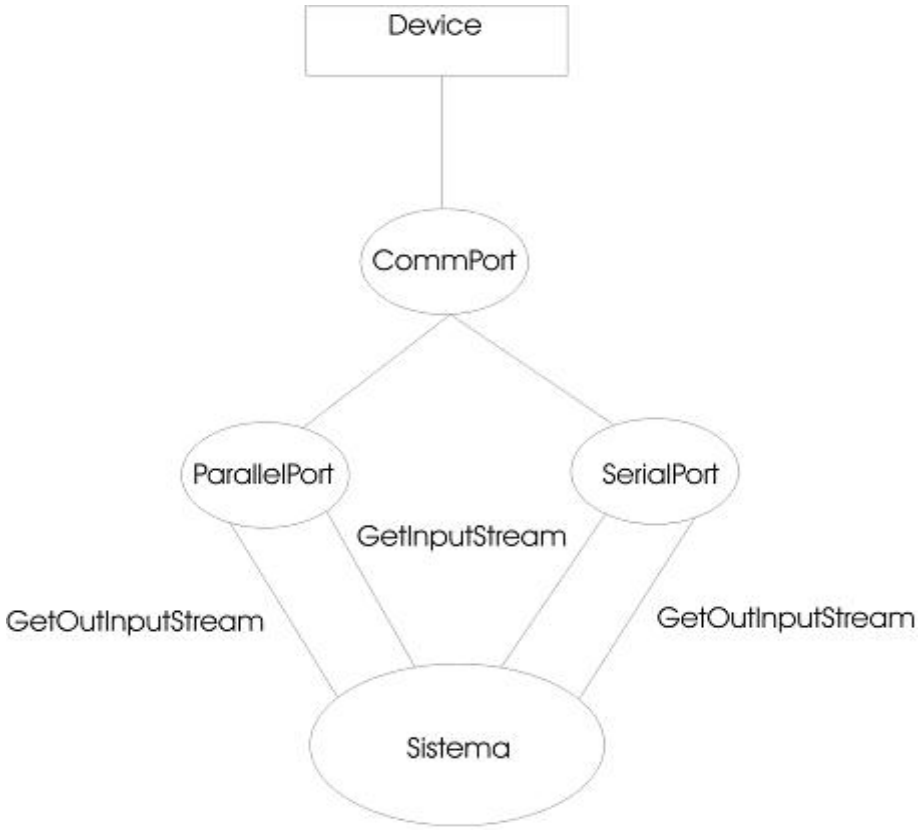

Figura 3.7 – Estrutura da classe serial do Java

3.3 Interrupções

Interrupção é um evento externo ou interno que obriga o microprocessador a suspender suas atividades temporariamente, para atender a este evento que o interrompeu. Em resumo, é uma ocorrência que faz o microprocessador parar a sua rotina e se desviar para outro ponto do software, em que se localiza o serviço de interrupção que foi gerado pela ocorrência (Nicolosi, 2004).

O microcontrolador 8051 possui três fontes de interrupção: a interrupção por software (instrução), a interrupção pedida por periférico externo e a interrupção pedida por periférico interno (timer/counter, porta serial, etc.). O 80C552 pode ser interrompido de 15 maneiras. As primeiras cinco, mostradas na Tabela 1, são compatíveis com o microcontrolador 8051 (ELS, 2001).

Tabela 1 – Interrupções 8051 / 80c552		
Tipo	**Descrição**	**Endereço**
Externo	Interrupção externa *INTO;*	0x0003
Externo	Interrupção externa *INT1*;	0x0012
Interno/Externo	Temporizador / contador interno *TIMER0*;	0x000B
Interno/Externo	Temporizador / contador interno *TIMER1*;	0x001B
Interno	Canal de comunicação serial.	0x0023

As interrupções do 80c552 são realizadas por meio de vetores, quer dizer, o vetor de interrupção (endereço de início da interrupção) é fixo, e não pode ser mudado pelo usuário. O desvio das interrupções CW552 é realizado pelo programa monitor gravado no EPROM do 80C552. Na verdade, o que está escrita no EPROM é um desvio incondicional para um endereço na memória RAM. Por exemplo, a interrupção externa INT0 tem como endereço físico 0x0003. No endereço 0x0003 gravado na EPROM está um comando de desvio incondicional para o endereço 0x8003 que está na memória de programa do 80C552 (ELS, 2001).

3.3.1 Como Programar as Interrupções

Temos três palavras chaves para a programação das interrupções: a primeira é chamada de INTERRUPT ENABLE – IE, endereço A8h, para liberar cada interrupção desejada a outra INTERRUPT PRIORITY – IP, endereço B8h, para programar a prioridade das interrupções liberadas no IE e por último TCON, endereço 88h, que se programa o tipo de disparo e se tem os flags IE0 e IE1 (Nicolosi, 2004).

No do bloco IE, temos as chaves específicas de ligar/liberar as interrupções específicas, são elas: EA, EX0, ET0, EX1, ET1 e ES, juntas elas formam a palavra IE, conforme mostrado na Tabela 2 (Nicolosi, 2004).

Tabela 2 – Bits que compõem o IE									
Nomes	EA	X	X	ES	ET1	EX1	ET0	EX0	**IE**
End. Bit	AF	AE	AD	AC	AB	AA	A9	A8	A8H

- EA (Enable All) - em "0" desabilita todas as interrupções, em "1" permite que cada interupção seja habilitada individualmente;
- ES (Enable Serial) - Habilita ou desabilita a interrupção pedida pelo canal de serial, "0" desabilita e "1" habilita se EA = 1;
- ET1 (Enable Timer 1) - Habilita ou desabilita a interrupção pedida pelo temporizador 1, "0" desabilita e "1" habilita se EA = 1;
- EX1 (Enable External 1) - Habilita ou desabilita a interrupção externa 1, "0" desabilita e "1" habilita se EA = 1;
- ET0 (Enable Timer 0) - Habilita ou desabilita a interrupção pedida pelo temporizador 0, "0" desabilita e "1" habilita se EA = 1;
- EX0 (Enable External 0) - Habilita ou desabilita a interrupção externa 0, "0" desabilita e "1" habilita se EA = 1.

Cada interrupção no 8051 poderá ser individualmente programada para um dos dois níveis de prioridade (0 ou 1), o que permite jogar uma interrupção em dois grupos: grupo de alta prioridade, que são atendidas em primeiro lugar e grupo de baixa prioridade, que são atendidas em segundo lugar. Isto é feito através dos bits do registro IP, a palavra IP - INTERRUPT PRIORITY é formada pelos bits PX0, PT0, PX1, PT1 e PS, vide ilustração da Tabela 3 (Nicolosi, 2004). Uma interrupção de nível baixo poderá ser interrompida por outra de nível alto, mas não por outra de nível baixo. Uma interrupção de nível alto não poderá ser interrompida por qualquer outra fonte de interrupção.

Tabela 3 – Bits que compõem o IP									
Nomes	X	X	X	PS	PT1	PX1	PT0	PX0	**IP**
End. Bit	BF	BE	BD	BC	BB	BA	B9	B8	B8H

- PS (Priority Serial) - Nível de prioridade para o canal serial;
- PT1 (Priority Timer 1) - Nível de prioridade para o temporizador 1;
- PX1 (Priority External 1) - Nível de prioridade para a interrupção externa 1;

- PT0 (Priority Timer 0) - Nível de prioridade para o temporizador 0;
- PX0 (Priority External 0) - Nível de prioridade para a interrupção externa 0.

Os pinos externos de interrupção do 8051 podem ser sensíveis a nível ou borda do sinal que representa o evento externo. Se desejarmos acionar as referidas interrupções (INT0 e INT1) por nível ou borda, opta-se pelos bits IT0, IT1, IE0 e IE1, caso se escolha a interrupção por nível programa-se o bit 0 e por borda o bit 1 (Nicolosi, 2004). Estes bits estão alocados dentro de uma palavra chamada TCON, conforme ilustração da Tabela 4.

Tabela 4 – Bits que compõem o TCON

Nomes	*	*	*	*	IE1	IT1	IE0	IT0	**TCON**
End Bit	*	*	*	*	8B	8A	89	88	88H

*O símbolo * implica que o bit é existente, mas será tratado no item 3.4 deste capitulo.*

- IT0 igual à "0" : ativa a interrupção com nível lógico baixo no pino INT0;
- IT0 igual à "1" : ativa a interrupção com borda de descida no pino INT0;
- IT1 igual à "0" : ativa a interrupção com nível lógico baixo no pino INT1;
- IT1 igual à "1" : ativa a interrupção com borda de descida no pino INT1;
- IE X : fica em "1" quando for detectado uma borda de descida (pedido de interrupção). É resetado após o atendimento da rotina de tratamento da interrupção.

Na Tabela 5, mostrada a seguir, se encontram as principais interrupções do 80c552.

Tabela 5 – Principais Registradores do Microcontrolador 80c552		
Registrador	**Descrição**	**Endereçamento**
IE	Habilita interrupções	Bit a bit
IP	Define a prioridade das interrupções	Bit a bit
TMOD	Define o modo de operação do temporizador/contador	Byte
TCON	Controle das interrupções externas e do temporizador	Bit a bit
TH0	Registrador do temporizador/contador 0	Byte
TL0	Registrador do temporizador/contador 0	Byte
TH1	Registrador do temporizador/contador 1	Byte
TL1	Registrador do temporizador/contador 1	
SCON		
PCON	Configura power-control e porta serial (UART)	Byte
SBUF	Registrador para acessar e ler porta serial (UART)	Byte

3.3.2 Interrupções no projeto.

No projeto, o leitor de código de barras possui um conector do tipo PS2, idêntico ao conector de um teclado normal, na Figura 3.8, é demonstrada a forma que o leitor foi ligado ao CW552 e os pinos utilizados do PS2. O pino 5 (CLK) do conector PS2 foi conectado na porta INT0 do 80C552. Cada vez que uma interrupção é gerada pelo leitor de código de barras, significa que dados estão sendo enviados pelo pino 1 (DATA) do conector. O pino 1 de dados foi conectado à entrada do pino T0 do microcontrolador, onde os dados são recebidos. No microcontrolador a cada sinal de nível alto na porta INT0, a porta T0 é monitorada, verificando se dados estão chegando, e a cada bit recebido, ele é armazenado para posterior tratamento da informação.

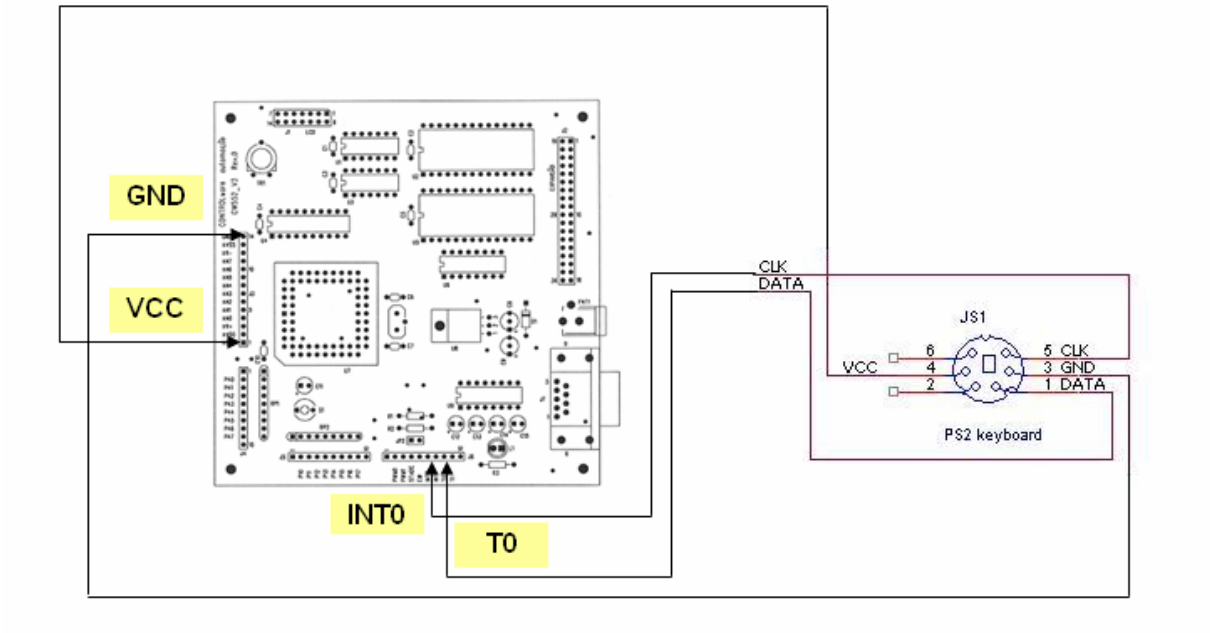

Figura 3.8 – Conector PS2 conectado ao CW552

3.4 Contadores e Temporizadores

O 80c552 possui internamente dois Contadores/Temporizadores (TIMER/COUNTER) denominados como T0 E T1. Ambos podem ser configurados para operar como temporizador ou contador de eventos, individualmente (Nicolosi, 2004).

Na função de *temporizador (timer)*, um registro será incrementado a cada ciclo de máquina. Considerando que cada ciclo de máquina consiste em 12 períodos do clock, a taxa de contagem será de 1/12 da freqüência do clock. Na função de *contador (counter)*, um registro será incrementado em resposta a uma transição de "1" para "0" de seu correspondente pino de entrada externa, T0 e T1. Nesta função, os pinos externos (T0 e T1) são amostrados a cada ciclo de máquina. Quando uma amostragem indicar um nível alto em um ciclo de máquina e um nível baixo no próximo ciclo, o contador será incrementado. A máxima taxa de contagem será de 1/24 da freqüência do clock, visto que são necessários dois ciclos de máquina para o reconhecimento de uma transição de "1" para "0" (Nicolosi, 2004).

Os dois Contadores/Temporizadores podem ser programados em quatro modos possíveis, a saber:

- **MODO 0:** Contador com capacidade máxima de 13 bits;

- **MODO 1:** Contador com capacidade máxima de 16 bits, utilizado no projeto;
- **MODO 2:** Contador com capacidade máxima de 8 bits e auto-reload;
- **MODO 3:** Contador misto.

Para programá-los, temos dois Registradores de Funções Especiais chamados TMOD e TCON, o TMOD é o registro de controle de modo Temporizador/Contador, é neste registro que é feita a seleção de função Temporizador ou Contador e a seleção do modo de operação (modo 0, 1, 2 ou 3) e o TCON é um registro acessado pelo endereço 88H e é bit endereçável (Nicolosi, 2004). Nas Tabelas 6 e 7, são descritas as funções de cada bit de ambos os registradores.

Tabela 6 – Bits que compõem o TMOD.

Nomes	GATE.1	C/T.1	M1.1	M0.1	GATE.0	C/T.0	M1.0	M0.0	**TMOD**
End Bit	*	*	*	*	*	*	*	*	89H

- **C/T.x:** seleciona a função, TEMPORIZADOR (timer) ou CONTADOR (counter), será selecionado como TEMPORIZADOR se este bit estiver em "0" e "1" a operação será como CONTADOR.
- **GATE.x:** quando GATE.x e Trx (um bit que compõe o TCON) for igual à 1, o temporizador irá operar somente enquanto o pino INTx for igual à 1 (controle por circuito). Quando GATE.x for igual à 0, o temporizador irá operar somente quando TRx for igual à 1 (controle por software).
- **M1.x e M0.x:** Bits de seleção de modo de operação.

Tabela 7 – Bits que compõem o TCON.

Nomes	TF1	TR1	TF0	TR0	*	*	*	*	**TCON**
End Bit	8F	8E	8D	8C	*	*	*	*	88H

*O símbolo * implica que o bit é existente, mas foi tratado no item 3.1.7 deste capitulo.*

- **TFx:** bit de overflow do temporizador, ativado pelo circuito quando ocorrer um overflow no temporizador, gerando um pedido de interrupção. É ressetado pelo hardware após o terminada a rotina de interrupção.

- **TRx:** bit de controle de operação do temporizador. É o bit que liga e desliga o C/T. Para ligar o temporizador, o software deverá setar este bit, e para desligar deverá ressetá-lo.

3.4.1 Modo 1 (16 Bits)

Este modo funciona com uma contagem de 16 bits, sendo utilizado os registros TH1 e TL1 ou TH0 e TL0 para formar estes 16 bits. Os registradores TH e TL funcionam como byte mais e menos significativos do temporizador, desta forma podemos contar de 0000h até FFFFh (65536 contagens), vide a Figura 3.9, gerando um bit TF(0xFFFF) e indicando que chegou a contagem máxima. Este bit TF pode ser usado para monitoração da interrupção, permitindo ativar ou desligar o timer do microprocessador (Nicolosi, 2004).

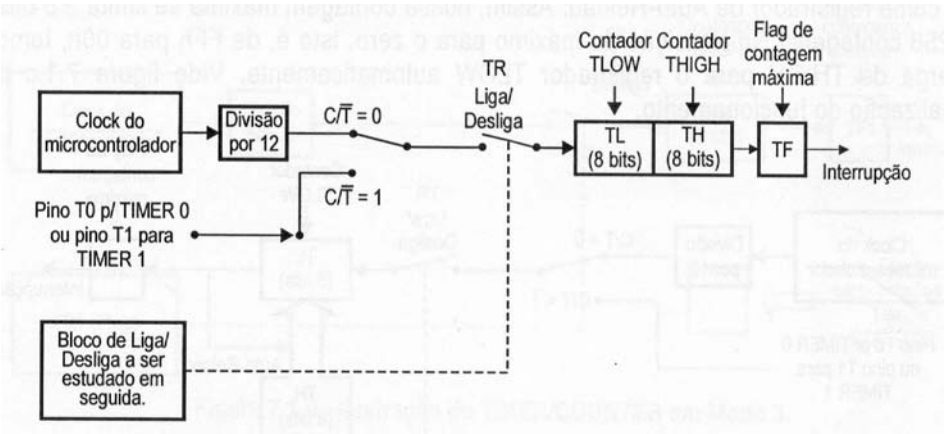

Figura 3.9 – Timer/Counter Modo 1

CAPÍTULO 4 – MODELO DESENVOLVIDO

O projeto desenvolvido está dividido em duas etapas: a primeira etapa apresenta o protótipo de acesso, onde são apresentados os componentes que o compõe e o funcionamento de cada um. Na segunda etapa é apresentado o sistema web, arquitetura do sistema, funcionalidades e telas. Junto com a apresentação de ambas as etapas são descritas as tecnologias e ferramentas utilizadas para seu desenvolvimento.

Os componentes que fazem parte do projeto foram escolhidos visando à qualidade, simplicidade, rapidez e baixo custo. São ferramentas consolidadas do mercado e que possuem vasta documentação. Na Tabela 8, tem-se a relação de todas as ferramentas e tecnologias utilizadas no decorrer do projeto e o custo de cada. Ressaltando que esses preços servem como base para quem venha a utilizar algum desses componentes ou ferramentas.

Tabela 8 – Custo do Projeto

Componentes e Ferramentas	Custo	Observação
Eclipse 3.1	R$ 0,0	Utilizado no desenvolvimento do Java e C.
MYSQL 4.0	R$ 0,0	Banco de dados do projeto.
Visio 2003	R$ 101,08	Utilizado na criação do fluxograma do projeto e modelagem do sistema. *Fonte do preço: www.submarino.com.br*
Servidor WEB – Tomcat 5.0	R$ 0,0	Servidor WEB do projeto
Framework Struts	R$ 0,0	Padrão de desenvolvimento do Sistema WEB.
Leitor de Código de Barras	R$ 85,90	*Fonte do preço: www.mercadolivre.com.br*
Teclado Alfanumérico	R$ 30,00	Foi adaptado um teclado

		telefônico. *Fonte do preço: www.mercadolivre.com.br*
Kit de Desenvolvimento CW552	R$ 500,00 [1]	*Fonte do preço: http://www.persocom.com.br/controlware/kitcw552.htm*
Conversor USB/Serial	R$ 60,00	*Fonte do preço: www.mercadolivre.com.br*
Cartão Magnético / Código de Barras	R$ 29,00	Preço de cada cartão. *Fonte do preço: http://www.pneutronic.com/loja.phtml*
Custo Total	**R$ 815,98**	

4.1 Modelagem de Dados

Um banco de dados é uma coleção de registros armazenados em um arquivo físico, ele é usado para armazenar informações estruturadas em tabelas e permitir a fácil recuperação desses dados. O papel de recuperação ou armazenamento dos dados é realizado pelo SGBD, sendo assim, é este gerenciador que fica responsável por fornecer as seguintes funções: permitir que os aplicativos acessem as informações armazenadas no banco, proteger a integridade dos dados e definir perfis de acesso ao banco, autorizando somente pessoas cadastradas a manipular os dados.

Neste projeto é utilizado uma base de dados simples, pois não é necessário realizar grandes consultas ou cadastros, como acontece em grandes sistemas. O Banco de dados utilizado é composto por quatro tabelas, conforme modelagem de dados mostrada na Figura 4.1, e tanto o sistema web quanto o protótipo de acesso utilizam as mesmas. O banco de dados escolhido no projeto foi o MYSQL 4, uma base gratuita e capaz de suportar o volume de dados proposto.

[1] Em um projeto real de controle de acesso o preço cairia bastante, visto que se utilizaria apenas o necessário na montagem elétrica do projeto.

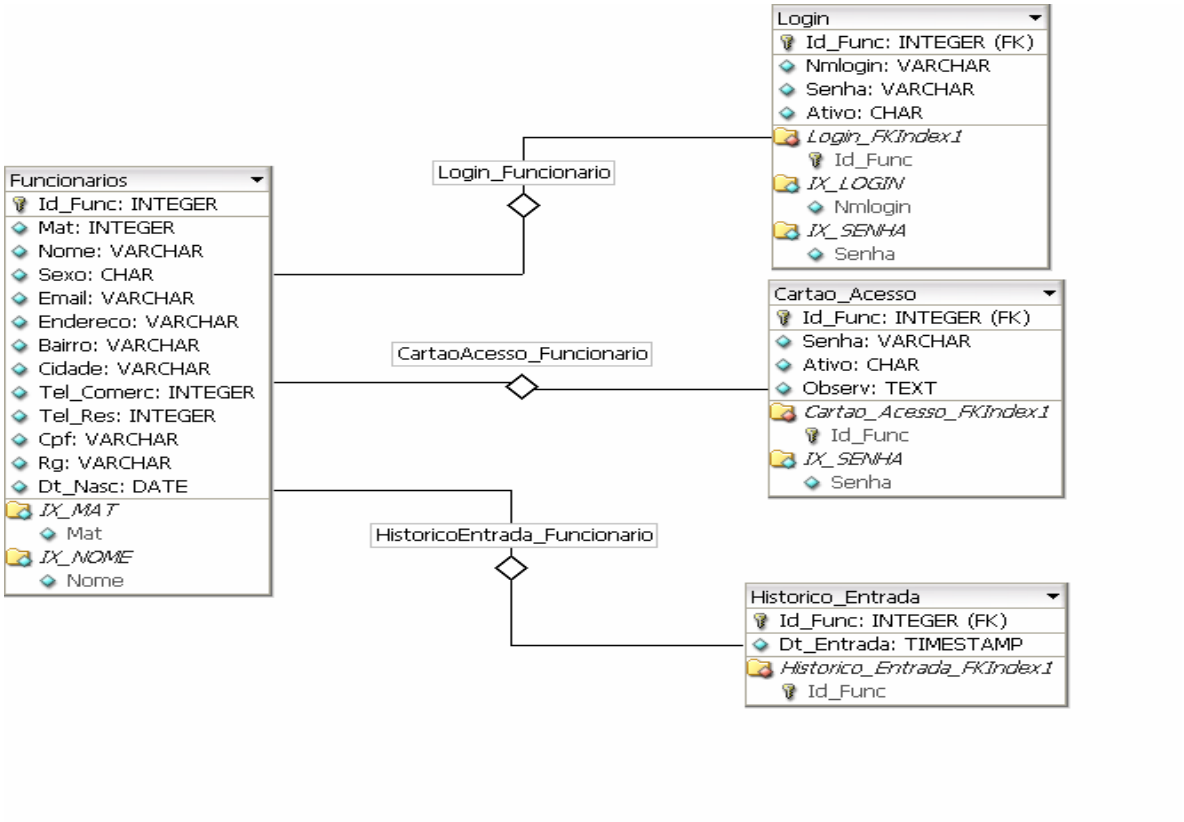

Figura 4.1 – Modelagem de Dados

A Tabela de Funcionários é a tabela principal desta modelagem, ela contém a chave principal do sistema (Id_Func – PRIMARY KEY) que é utilizada nas demais tabelas. Note que os Campos “Mat” e “Nome” estão indexados, isto foi feito pelo fato de serem campos de busca dentro do sistema, ou seja, quando é realizada uma consulta dentro do banco de dados pela matricula ou nome, faz com que ele vá diretamente a matricula ou nome escolhido, ao invés de percorrer todos os registros da tabela, ocasionando um ganho de desempenho da aplicação. O mesmo foi realizado nas Tabelas de Login e Cartao_Acesso, onde os campos indexados também serão campos de busca dentro do sistema.

Ressaltando que no início deste documento, foi dito que a empresa possui uma base de dados com as informações de cada funcionário. No projeto a base de dados utilizada é apenas uma base de “teste”, criada somente para fins acadêmicos.

4.2 Protótipo de Acesso

O desenvolvimento do protótipo de acesso integra os seguintes componentes e tecnologias: leitor de código de barras, teclado alfanumérico, microcontrolador 80c552 (Integrado no CW552), computador com interface serial (RS232) e programação na linguagem C. Entre os componentes citados, o microcontrolador tem o papel principal, pois é nele que serão armazenadas todas as regras do código desenvolvido em C, como:

- Comunicação entre o leitor de código de barras e portas lógicas do microcontrolador;
- Recepção dos dados digitados pelo funcionário no teclado alfanumérico;
- Mensagens no display para o funcionário, informando se o mesmo foi autorizado ou não a acessar o ambinete desejado.

Na Figura 4.2 é apresentada à fotografia do protótipo desenvolvido. Pode-se observar que o Kit CW552 está conectado ao computador. E essa comunicação é feita pela porta serial. O computador funciona como servidor, pois nele se encontra o banco de dados, servidor web e a aplicação rodando à espera dos dados enviados pelo protótipo de acesso.

Figura 4.2 – Fotografia do Projeto Desenvolvido

O código desenvolvido para monitorar a porta serial está dividido em duas fases:

1º Fase: Código desenvolvido em C, se encontra a codificação dos componentes que integram o CW552, que são o teclado alfanumérico e o leitor de código de barras, a cada vez que se passa o cartão e se digita a senha são enviados esses dados para a porta serial, ao final de cada componente lido foi enviado um caractere de identificação, no caso do código de barras foi à letra “L” e no caso do teclado a letra “U”. Isto foi feito para facilitar a identificação de cada componente. As funções responsáveis por enviar e receber os dados na porta serial são chamadas de ***putchar()*** e ***getchar()***, cujo código é apresentado a seguir.

```
void putchar (unsigned char a)  // manda caractere na porta serial
{
 while(!TI);  // espera ate completar ultima transmissao
 SOBUF=a;     // manda caractere para buffer transmissao
 TI=0;        // limpa o flag de transmissao
}

unsigned char getchar (void)  // rotina que espera caractere na porta serial
{
 while(!RI);     // espere receber caractere no buffer de recepcao
 RI=0;           // limpa flag de recepcao
 return(SOBUF);  // retorna caractere recebido
}
```

2º Fase: Código desenvolvido em Java, depois de iniciar o protótipo de acesso, ou seja, após ser compilado e executado o código em C, habilita-se a monitoração da porta serial utilizando a codificação do Java. Em primeiro lugar é iniciado o método ***ObterIdDaPorta()***, responsável por identificar a porta o qual está sendo realizada a comunicação serial. Em seguida chama-se o método ***AbrirPorta()***, responsável pela configuração da porta serial como: a taxa de transmissão ***baudrate***, o clock utilizado ***timeout***, o número de bits dos dados em um caractere ***data*** e por fim o número de bits de parada ***stop*** que definem o fim de um caractere. Logo em seguida, chama-se os métodos ***LerDados()*** e ***EnviarUmaString()***, responsáveis por enviar e receber os caracteres do protótipo de acesso. Os métodos citados anteriormente são representados na Figura 4.3.

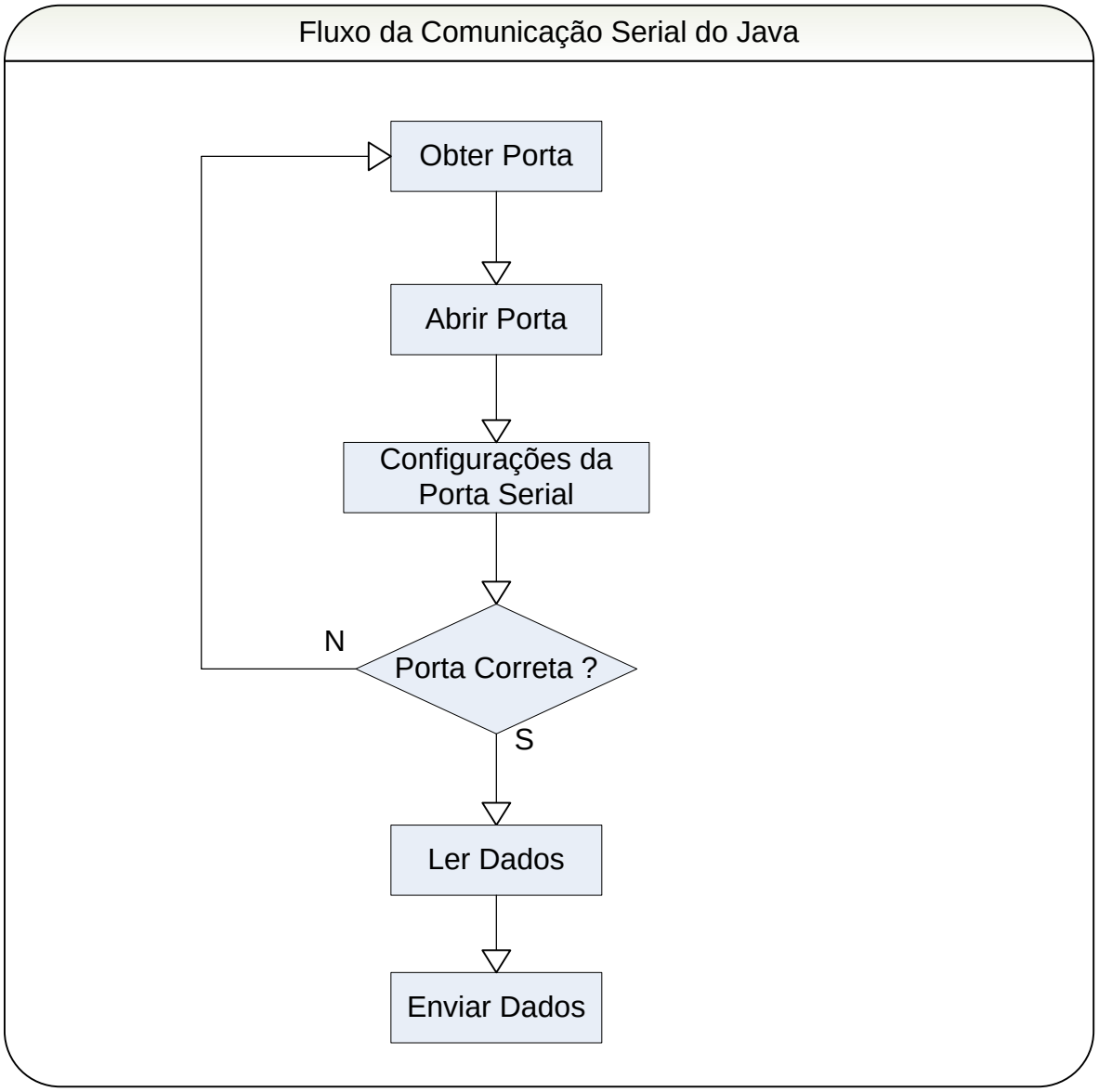

Figura 4.3 – Fluxo da Comunicação Serial do Java

O código-fonte do C e das classes do Java, são apresentados completos nos Apêndices B e C.

4.2.1 Teclado Alfanumérico

Neste projeto foi utilizado um teclado numérico semelhante a um teclado de telefone. Na Figura 4.4 é apresentada a configuração interna do teclado, sendo ele de quatro linhas por três colunas. Quando uma tecla é pressionada, a linha e a coluna adjacentes à tecla são curto-circuitadas, dessa maneira pode-se identificar qual tecla foi pressionada, ou seja, todo o teclado funciona como uma matriz 4 x 3.

Além das dez teclas numéricas (de zero a nove), o teclado possui também as teclas '*', que cancela todo o fluxo do protótipo e '#', que confirma a seqüência numérica digitada pelo usuário e a envia para a porta serial.

Para saber exatamente qual linha e coluna se encontram a tecla pressionada, logo a seguir é apresentado um trecho do código de varredura do teclado, desenvolvido em C. A porta escolhida para varredura do teclado foi à porta P1 do microncontrolador.

```
/*
 saida   pinos linha                    entrada pino  coluna
 pl0 =    tek1    [123]   linha1        pl1 =    tek2    [369#]  coluna1
 pl2 =    tek3    [456]   linha2        pl3 =    tek4    [2580]  coluna2
 pl4 =    tek5    [789]   linha3        pl5 =    tek6    [147*]  coluna3
 pl6 =    tek7    [*0#]   linha4
*/
void scan(void)
{
 unsigned char a;
 tec_ok=1;
 linha1=0; linha2=1; linha3=1; linha4=1;
 a=P1;
 a=a & 0x2a;
 if (a!=0x2a) tec_buf= 0x100|a;
 else
 {
  linha1=1; linha2=0; linha3=1; linha4=1;
  a=P1;
  a=a & 0x2a;
  if (a!=0x2a) tec_buf= 0x200|a;
  else
  {
   linha1=1; linha2=1; linha3=0; linha4=1;
   a=P1;
   a=a & 0x2a;
   if (a!=0x2a) tec_buf=0x300|a;
   else
   {
    linha1=1; linha2=1; linha3=1; linha4=0;
    a=P1;
    a=a & 0x2a;
    if (a!=0x2a) tec_buf=0x400|a;
    else tec_ok=0;
   }
  }
 }
}
```

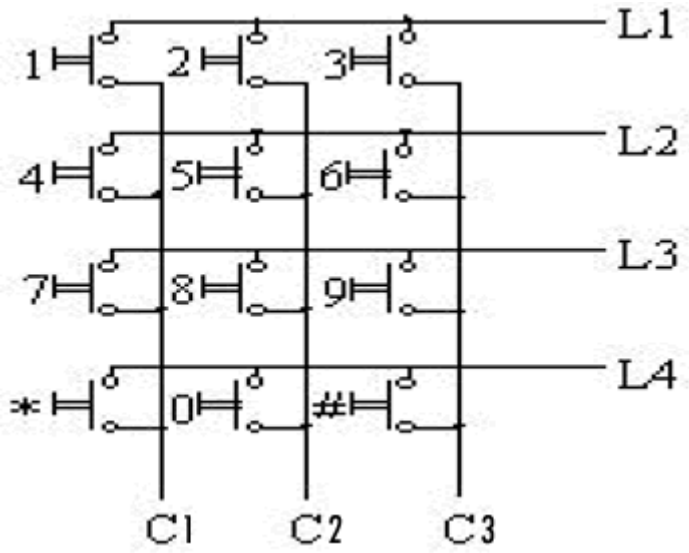

Figura 4.4 – Configuração Interna do Teclado

4.2.2 Leitor Código de Barras

O leitor de código de barras do projeto possui uma entrada PS2, que funciona da seguinte maneira: cada vez que o cartão magnético é passado no leitor é gerada uma interrupção. E essa interrupção é monitorada pelo pino 1 de sincronismo (*Clock*), que por sua vez envia os dados para o pino 2 de dados (*Data*), ou seja, enquanto estiver ocorrendo a interrupção, quer dizer que estão sendo enviados dados para a porta serial. Os outros pinos que compõem o conector PS2 são os pinos de alimentação (*Vcc*) e o de aterramento (*Gnd*), conforme Figura 4.5. A maneira que cada pino foi ligado ao microcontrolador é apresentada no item 3.1.9.

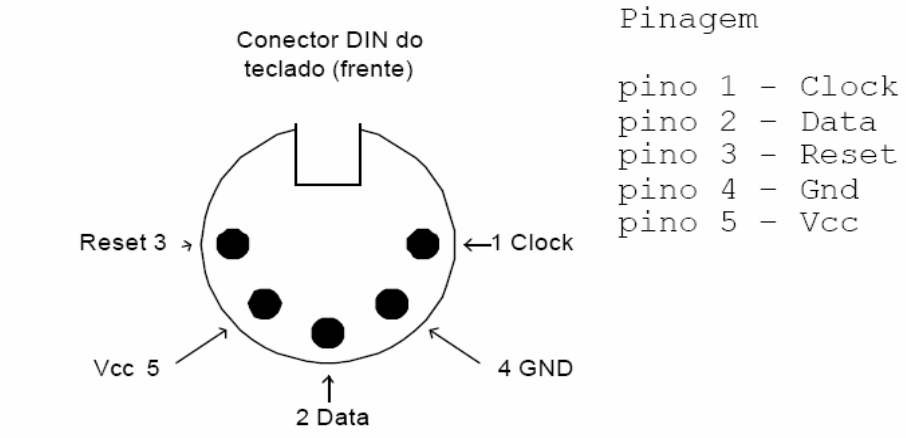

Figura 4.5 – Conector PS2 do Leitor de Código de Barras

Para entender melhor o funcionamento da leitura do código de barras, mostrarei o código em C responsável pela leitura do mesmo. Note que para se ler o dado é necessário saber o endereçamento exato de cada pino do microcontrolador.

```
#define kclock          INT0    // kit INT0 - pino ps2 clock
#define kdata           T0      // kit T0   - pino ps2 data


unsigned int scancode(void)
{
  unsigned int k;
  unsigned int i;
  k=0;
  for (i=0;i!=11;i++)
  {
    while(kclock==1);
    if (kdata==1) k=k+0x400;//
    k = k >> 1;
    while(kclock==0);
   }
  k = k & 0x00FF;
  return(k);
}

unsigned int scantec(void)
{
 unsigned int k1,l,m;
 k1=scancode();
 l=scancode();
 m=scancode();
// if ((k1==num_enter)||(k1==num_enter2)) {m=scancode();l=scancode();}
 return (k1);
}


char gettec(char c)
{
  char s;
  switch(c)
  {
    case 0xe0: s='E'; break;  case 0x5a: s='E'; break;
    case 0x45: s='0'; break;  case 0x16: s='1'; break;  case 0x1E: s='2'; break;
    case 0x26: s='3'; break;  case 0x25: s='4'; break;  case 0x2E: s='5'; break;
    case 0x36: s='6'; break;  case 0x3D: s='7'; break;  case 0x3E: s='8'; break;
    case 0x46: s='9'; break;
    default: s='N';
  }
  return(s);
}
```

Observa-se que na função ***scancode()*** existe um círculo em vermelho, é exatamente neste ponto que está sendo verificado se existe alguma interrupção, caso exista os dados serão armazenados na variável “K” da função. Nas outras funções ***scantec()*** é responsável pela chamada da função ***scancode()*** e a ***gettec()***

é responsável pelo reconhecimento dos números contidos no código de barras do cartão.

4.2.3 Padrão do Código de Barras

A tecnologia de código de barras foi desenvolvida visando à captura automática dos dados evitando assim os erros de digitação por parte do usuário. Para cada leitor existe um padrão de código de barras, os leitores ópticos devem estar habilitados para interpretar o seu tipo específico.

O padrão utilizado no Brasil é o EAN – “European Article Number”. A EAN é uma organização sediada em Bruxelas, na Bélgica, à qual as organizações membros EAN são filiadas. A EAN Internacional, junto com a UCC, gerencia o sistema EAN UCC (EAN, 2007).

Existem muitos tipos de codificação para gerar os símbolos e números das barras, os mais conhecidos: Padrão 25, Padrão 2 de 5 intercalado, Padrão 39, Padrão 39 com dígito de verificação, Padrão EAN 8, Padrão EAN 13 e outros.

O padrão de codificação utilizado no projeto foi o padrão 2 de 5 intercalado, ele é composto por uma seqüência de números, contendo a matricula do funcionário e por um números específicos da empresa.

4.3 Sistema WEB

O sistema web desenvolvido tem como principal objetivo monitorar constantemente os funcionários. Este sistema utiliza à mesma base de dados do protótipo de acesso, por esse motivo é capaz de interagir diretamente com este, bloqueando e desbloqueando funcionários a qualquer instante.

O nome dado ao sistema web foi *Sistema de Controle de Acesso – SCA*, ele compõe uma série de funcionalidades, entre elas: geração de relatórios, alteração de funcionários e bloqueio de cartões.

Para desenvolver o SCA foram utilizados os padrões de desenvolvimento do Java, dentre os vários padrões existentes optou-se pela utilização do padrão MVC, existente no Framework Struts (detalhes do funcionamento no item 2.7), DAO, MODEL, que são apresentados no decorrer deste Capítulo. Todos esses padrões

foram escolhidos, pelo fato de serem consolidados no mercado e por já possuir experiência pessoal.

A arquitetura do sistema, apresentada na Figura 4.6, é composta por três camadas: apresentação (WEB), negócio (Business) e integração ou recursos (Integration/Resources).

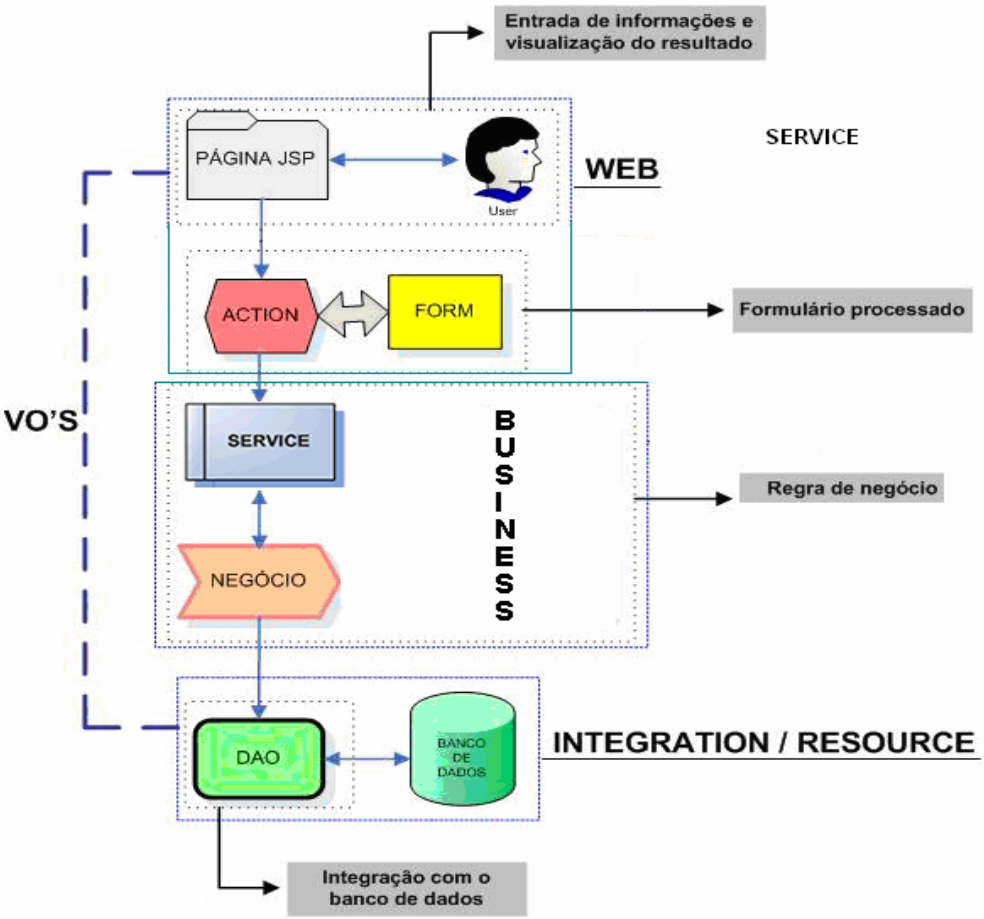

Figura 4.6 – Arquitetura do Sistema

A camada de apresentação (web) é a porta de entrada do usuário para o sistema, pois é através dela que este irá manipular os dados no servidor, por esse motivo essa fase requer um cuidado muito grande, pois a única coisa que o usuário verá do aplicativo serão as telas projetadas, se as mesmas não tiverem amigáveis pode-se perder todo o projeto, por melhor e mais avançado que ele esteja.

A camada de negócio (business) é o cerne dessa arquitetura, pois é nela que serão realizadas todas as regras de negócios pertinentes ao sistema, ou seja, todas as regras e processamento específicos de cada aplicação são executados nesta

parte. Esta camada não tem a obrigação de conhecer como ou onde os dados sobre os quais ela atua são armazenados, quem tem a responsabilidade de armazenar e recuperar os dados é um SGBD.

A última camada da arquitetura (Integration/Resources) é relativa aos dados armazenados fisicamente, é sem dúvida uma parte importante do sistema, pois se mal projetada o desempenho da aplicação pode ficar seriamente comprometida e até mesmo comprometer a integridade dos dados.

A seguir, é apresentado o diagrama de pacotes e classes do sistema, toda essa modelagem foi desenvolvida em UML, uma linguagem padronizada para modelagem de sistemas orientados a objetos.

4.3.1 Diagrama de Pacotes

No diagrama de pacotes é demonstrada as camadas de desenvolvimento do projeto. O projeto foi montado usando a tecnologia cliente-servidor, na qual o usuário por meio de ações (pacote cliente), solicita ao servidor a execução da atividade desejada. Essa transação está mostrada na Figura 4.7.

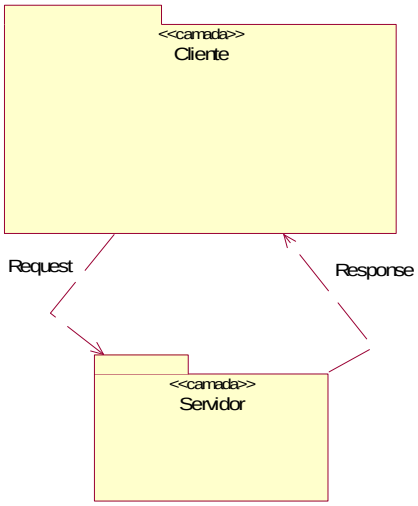

Figura 4.7 – Diagrama de Pacote do Sistema

Conforme representado na Figura 4.8, a camada Cliente possui as páginas "*.jsp*" que são compostas basicamente por páginas HTML, imagens, arquivos Javascripts(js) utilizados para validar dados na tela, arquivos CSS utilizados para configuração visual dos elementos dispostos nas telas.

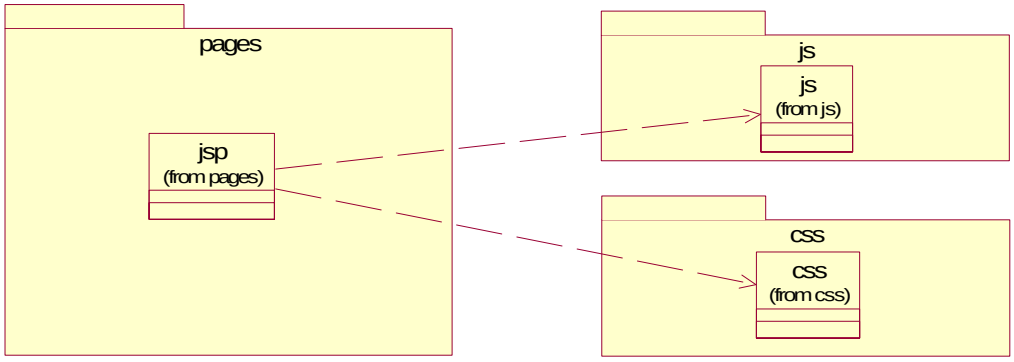

Figura 4.8 – Diagrama de Pacote Cliente

O pacote Servidor está subdivido em camadas que foram projetadas para reduzir a dependência entre as partes lógicas no desenvolvimento do software e permitir que as modificações evolutivas sejam feitas com baixo custo. Na Figura 4.9 mostra-se a integração das camadas.

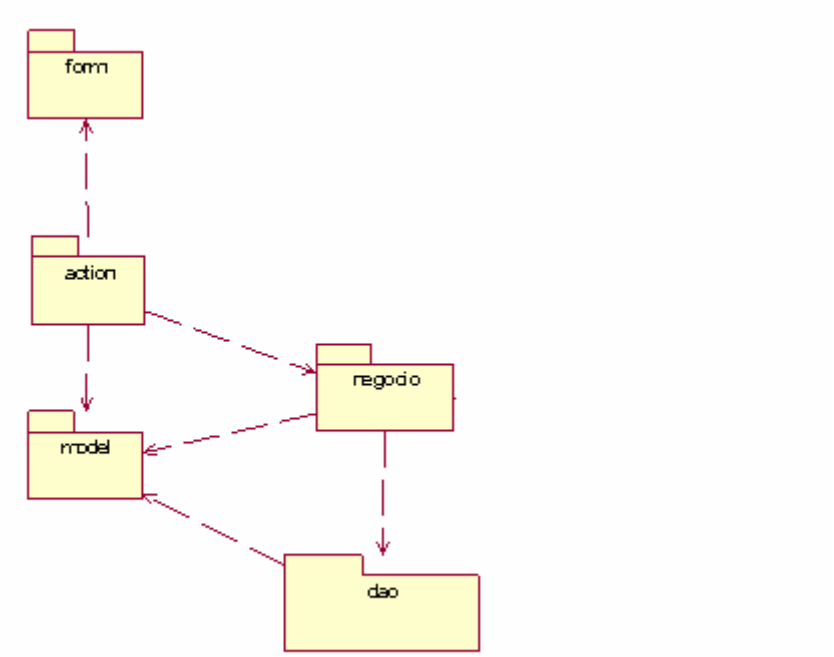

Figura 4.9 – Diagrama de Pacote Servidor

4.3.2 Diagrama de Classes

Neste diagrama é representado o modelo da estrutura estática das classes que compõem o projeto, permitindo visualizar o modelo da estrutura de um sistema, demonstração das classes e dos relacionamentos. Na Figura 4.10 é apresentada à

integração com o Struts e estabelece a representação macro das classes solicitadas na realização de ação do sistema.

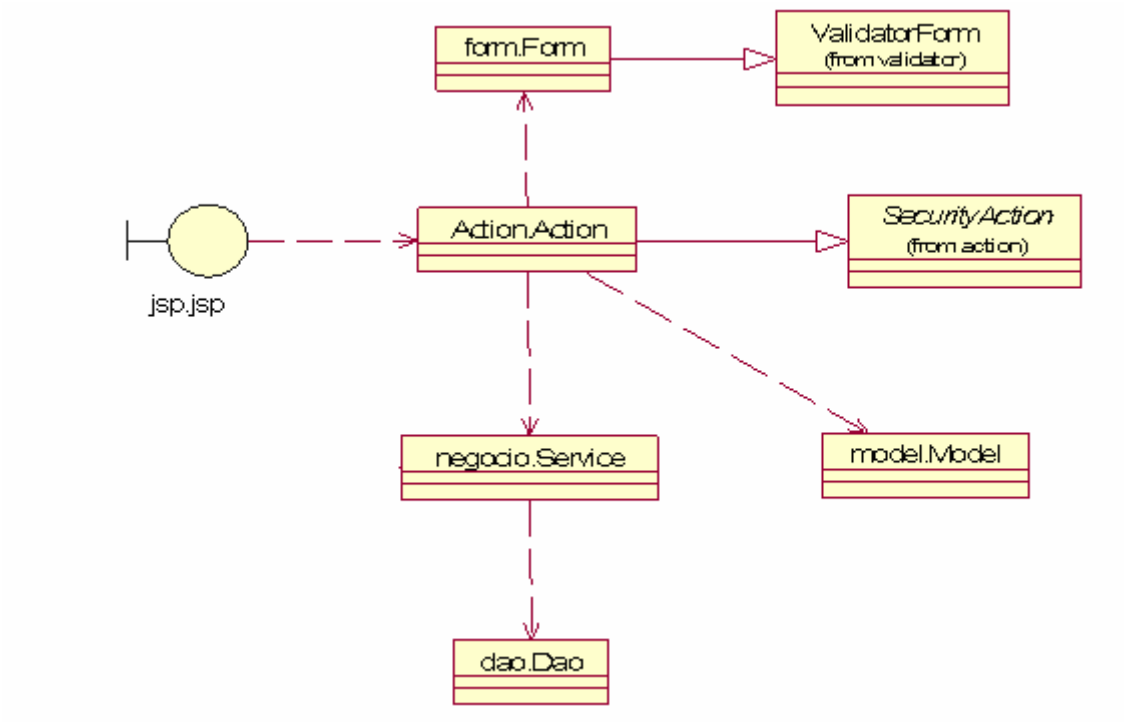

Figura 4.10 – Diagrama de classe macro das ações

A camada Form é a representação das variáveis usadas nas páginas do pacote cliente que será populada através da Action. Essa camada possui uma associação de herança em relação à classe ValidatorForm do Framework Struts utilizada para validar as variáveis conforme mostrada na Figura 4.11.

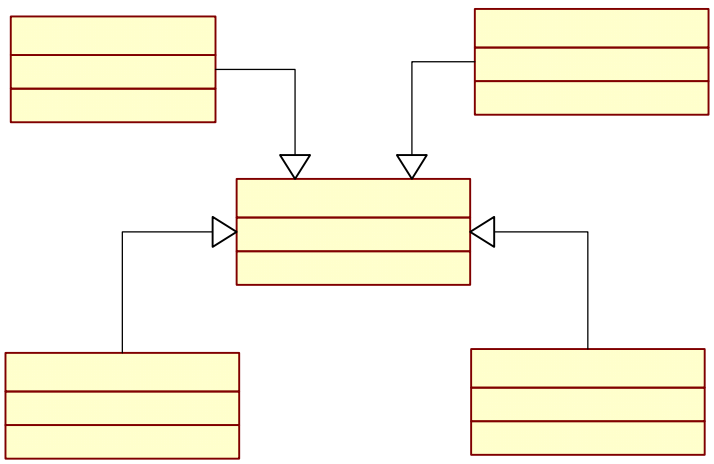

Figura 4.11 – Diagrama de Classes do Pacote Form

A camada Action possui como característica fundamental a interpretação dos comandos do cliente transformando-os em ações do sistema. Conforme a Figura 4.12 se observa que a arquitetura faz uso das classes SecurityAction e CrudAction.

A classe SecurityAction é utilizada com a finalidade de implementar condições de verificação de segurança, enquanto a classe CrudAction é utilizada para mapear as funções de acordo com a ação. Além disso, é na CrudAction que são implementadas as funcionalidades de inclusão e alteração. Todas as actions do sistema devem herdar da classe CrudAction que por sua vez herda da SecurityAction.

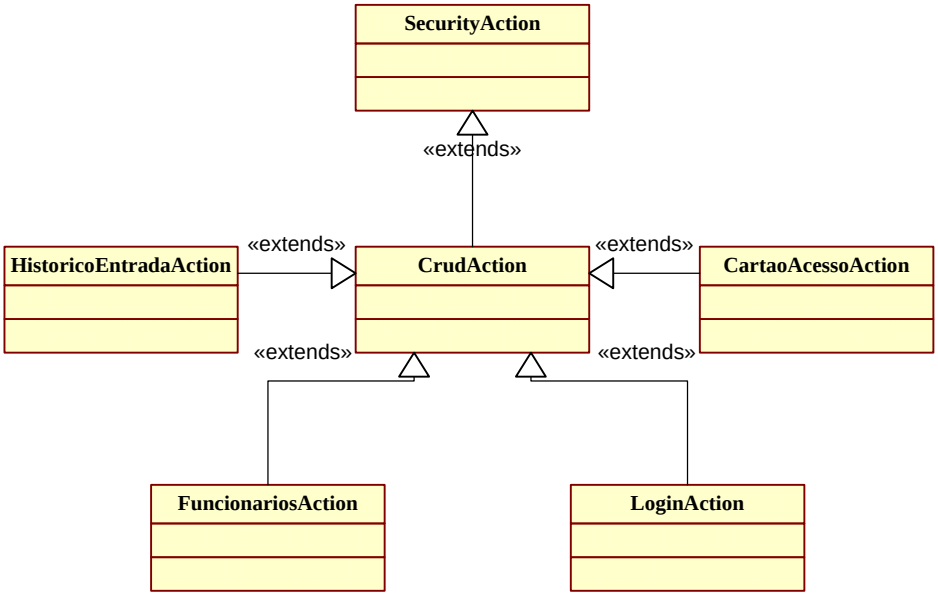

Figura 4.12 – Diagrama de Classes do Pacote Action

A camada model é responsável por encapsular os valores de retorno dos componentes de negócio, ou seja, ele armazena todos os dados necessários em um único objeto, resultando em uma única chamada, ao invés de chamar cada dado individualmente, isso acarreta em um ganho de desempenho da aplicação. Podemos citar como exemplo os valores de retorno de um método DAO. As classes do pacote model são mostradas na Figura 4.13.

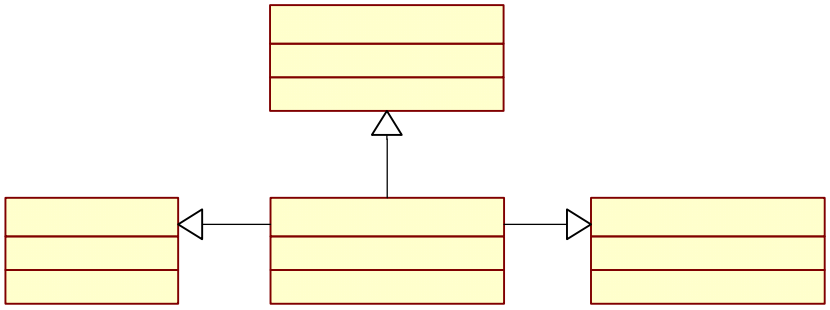

Figura 4.13 – Diagrama de Classes do Pacote Model

As classes da camada de Serviço (Negócios) colaboram entre si para executar todas as regras de negócio do sistema conforme representado na Figura 4.14. A visão externa desses componentes é representada por meio de interfaces. Cada aplicação possui sua classe referente que herda da classe CrudServiceImpl, que possui todos os métodos básicos já implementados.

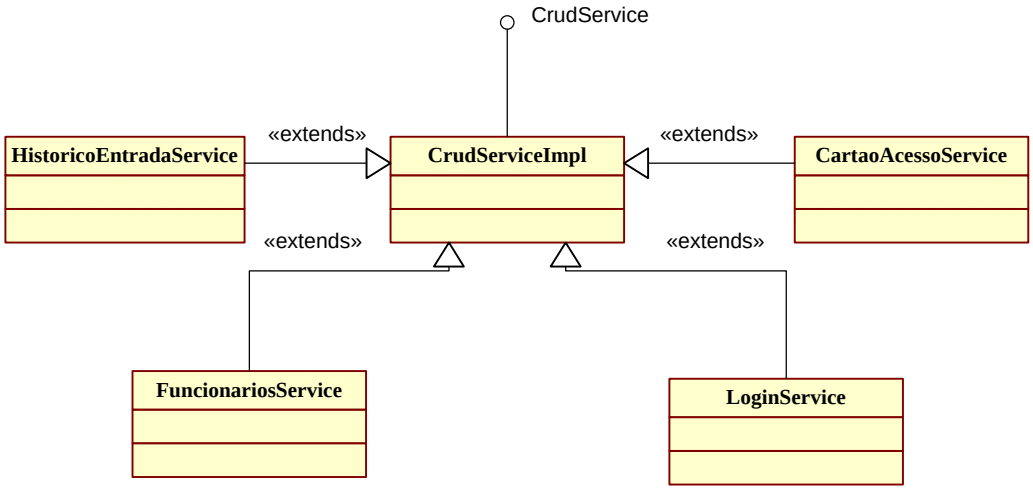

Figura 4.14 – Diagrama de Classes do Pacote Negócio

A camada DAO (Data Access Objetcs), Figura 4.15, é responsável pela integração com o Banco de Dados. Todas as classes herdam da Classe CrudDaoImpl os métodos create, update e restore já implementados com as transações necessárias, necessitando apenas criar as queries que serão utilizadas para alterações, consultas ou inserções no Banco.

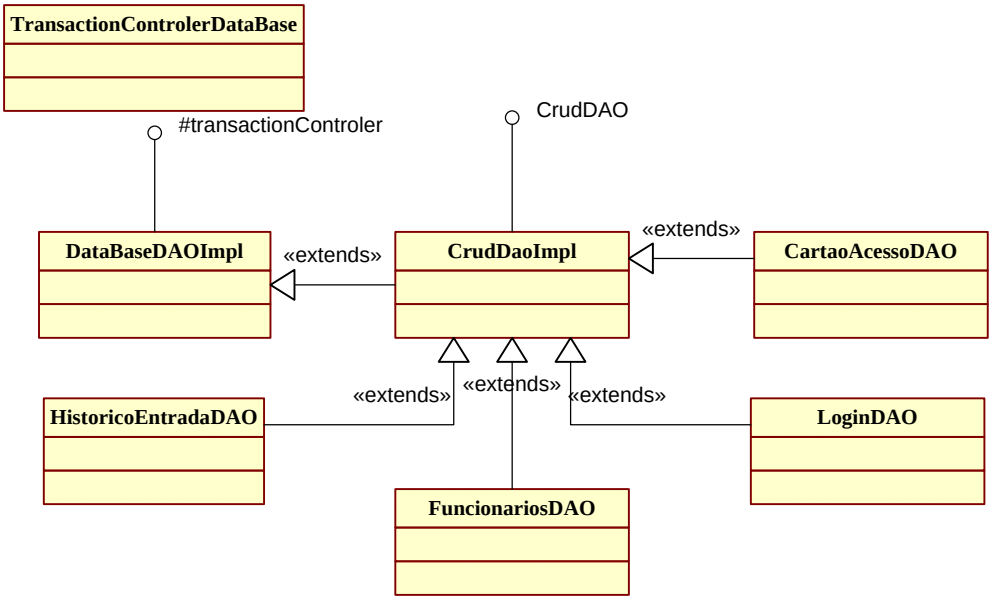

Figura 4.15 – Diagrama de Classes do Pacote Negócio DAO.

4.3.3 Telas e Funcionalidades

Acesso ao Sistema

Na Figura 4.16, é mostrado o fluxo inicial do sistema. A tela de login do usuário visa oferecer maior segurança aos dados e registros armazenados. Os funcionários só terão acesso aos dados, se tiverem sido previamente cadastrados. Quando o usuário for autorizado, é mostrado a tela principal do sistema com os itens do menu e sua foto no cartão de acesso.

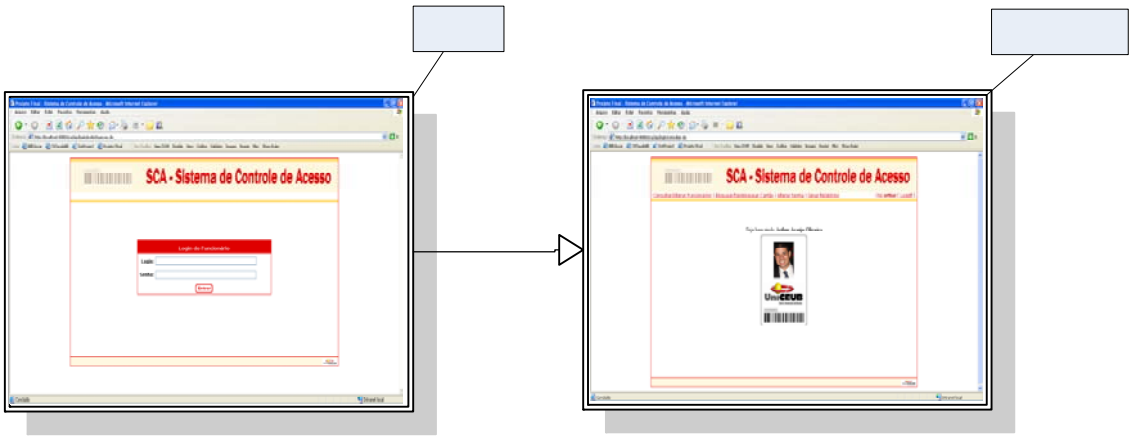

Figura 4.16 – Fluxo Inicial do Sistema

Opções do Menu SCA:

1) Consultar/Alterar Funcionários;
2) Bloquear/Desbloquear Cartão;
3) Alterar Senha;
4) Gerar Relatório;

Consultar/Alterar Funcionários

Na Figura 4.17 é mostrada às telas referentes aos funcionários. A tela de consulta é constituída por nome e matrícula, parâmetros usados para pesquisar os funcionários. A pesquisa é bastante flexível e rápida, você pode digitar apenas o início do nome ou matricula que o sistema encontrará o funcionário. Caso encontre, é mostrada uma lista com o nome e matricula de cada um. Para alterar, excluir ou verificar os dados do funcionário você clicará no nome do funcionário que mostrará todas as informações referentes ao mesmo.

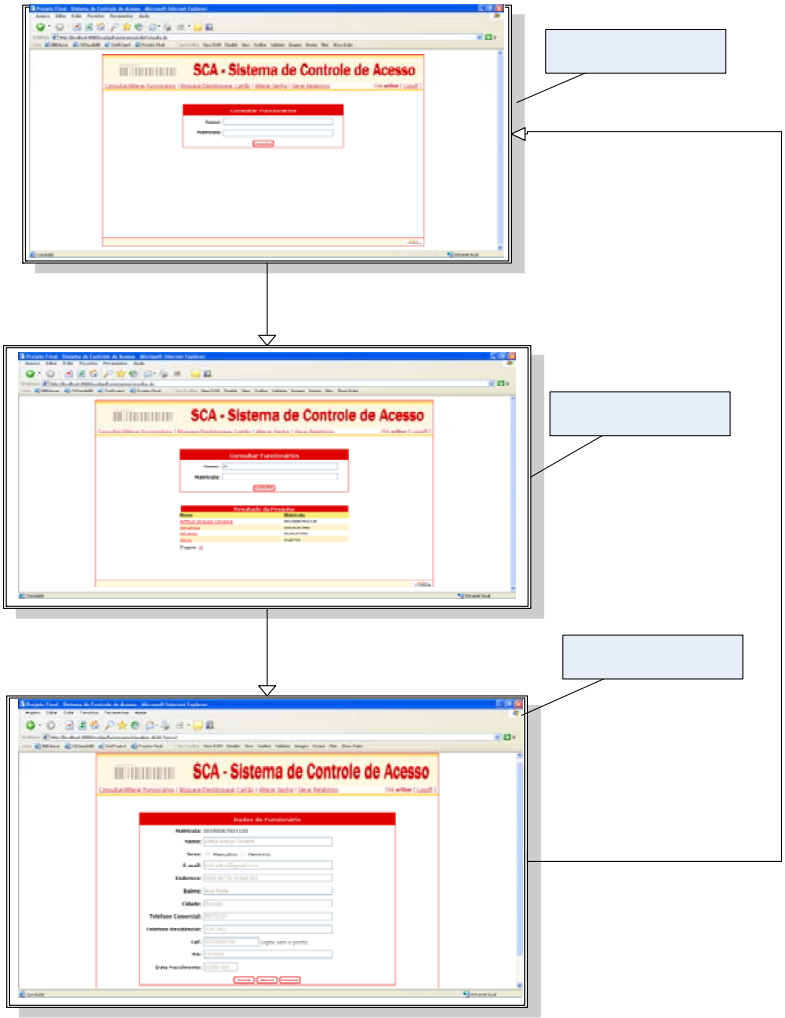

Figura 4.17 – Fluxo do Funcionário no Sistema

Bloquear/Desbloquear Cartão

Na Figura 4.18, é mostrada às telas referentes aos cartões de acesso. O usuário poderá bloquear ou desbloquear o cartão em caso de perda ou em virtude do desligamento do funcionário da empresa. Para realizar o bloqueio ou desbloqueio do cartão o usuário realizará o mesmo procedimento do consultar/alterar funcionários, consultando o cartão, caso exista clicará no nome do funcionário, logo após será mostrado o status do cartão e um campo de observação do mesmo.

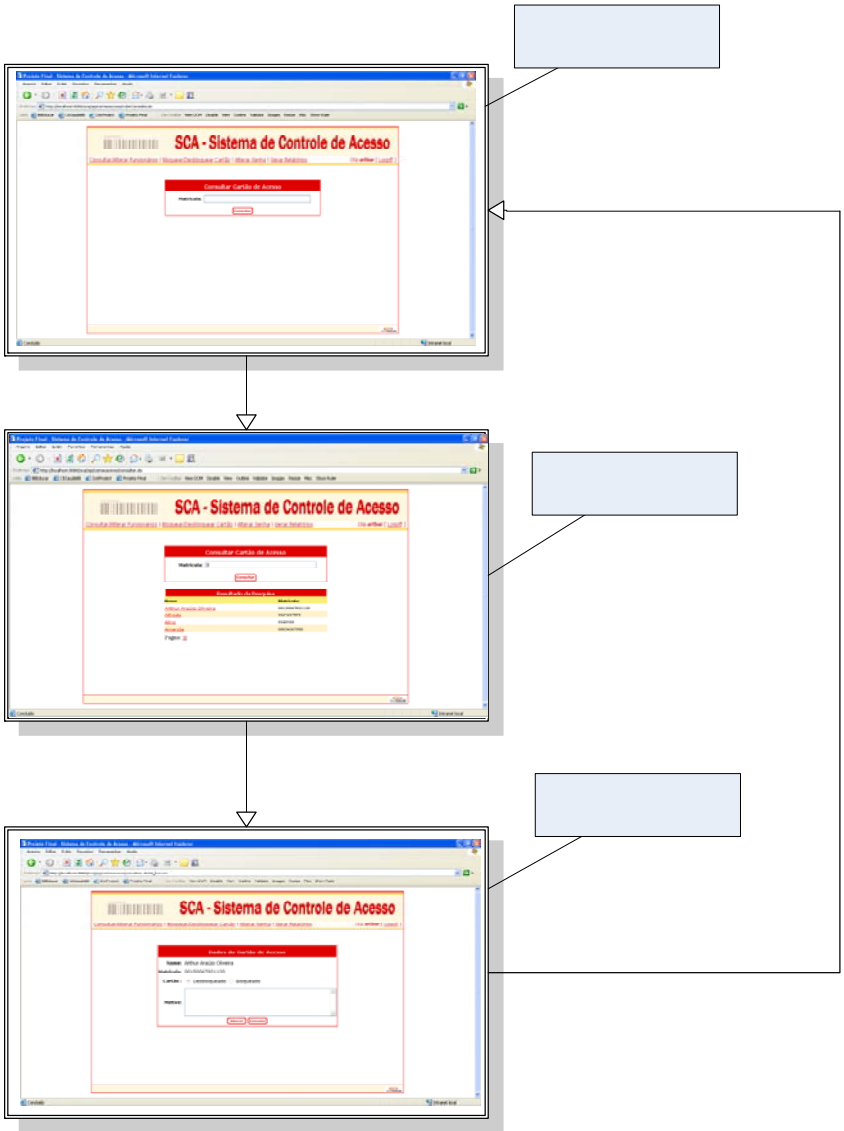

Figura 4.18 – Fluxo do Cartão de Acesso no Sistema

Alterar Senha

Na Figura 4.19 é mostrada a tela de alteração de senha do usuário no sistema. A qualquer momento o usuário pode trocar sua senha, para isso o mesmo tem que digitar sua senha atual e digitar a nova, caso a senha antiga não for valida o sistema o alertará que a senha não foi alterada.

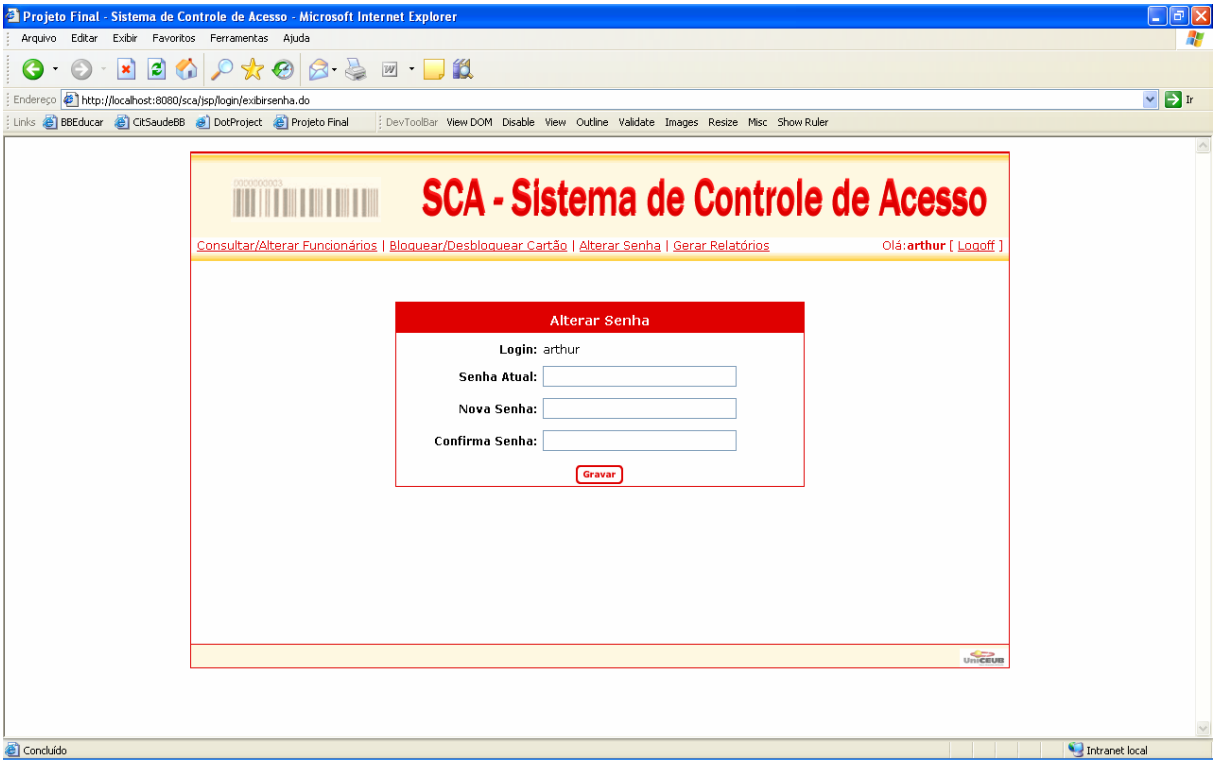

Figura 4.19 – Alterar Senha

Gerar Relatório

Na Figura 4.20 é mostrada a tela para gerar o relatório de acessos do funcionário. O relatório será constituído pelo nome do funcionário, matrícula, data e hora do local acessado, conforme Figura 4.21.

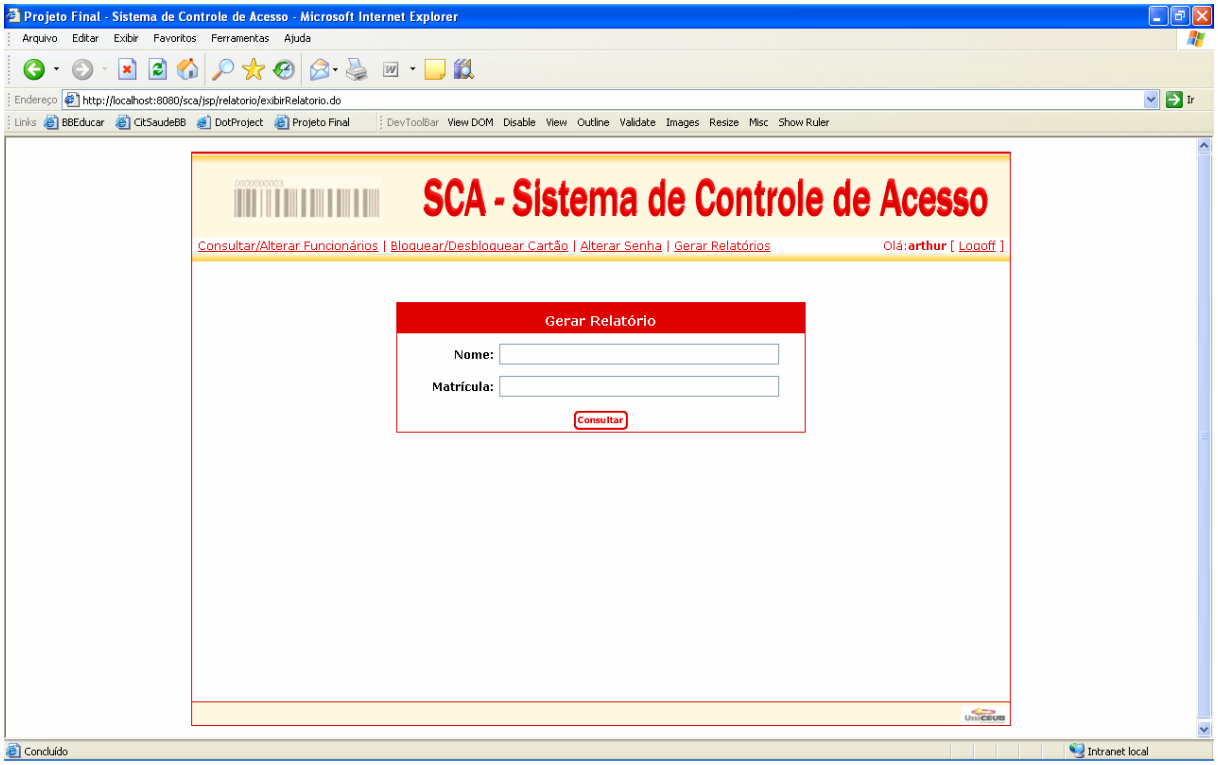

Figura 4.20 – Tela de Consulta para Gerar Relatório

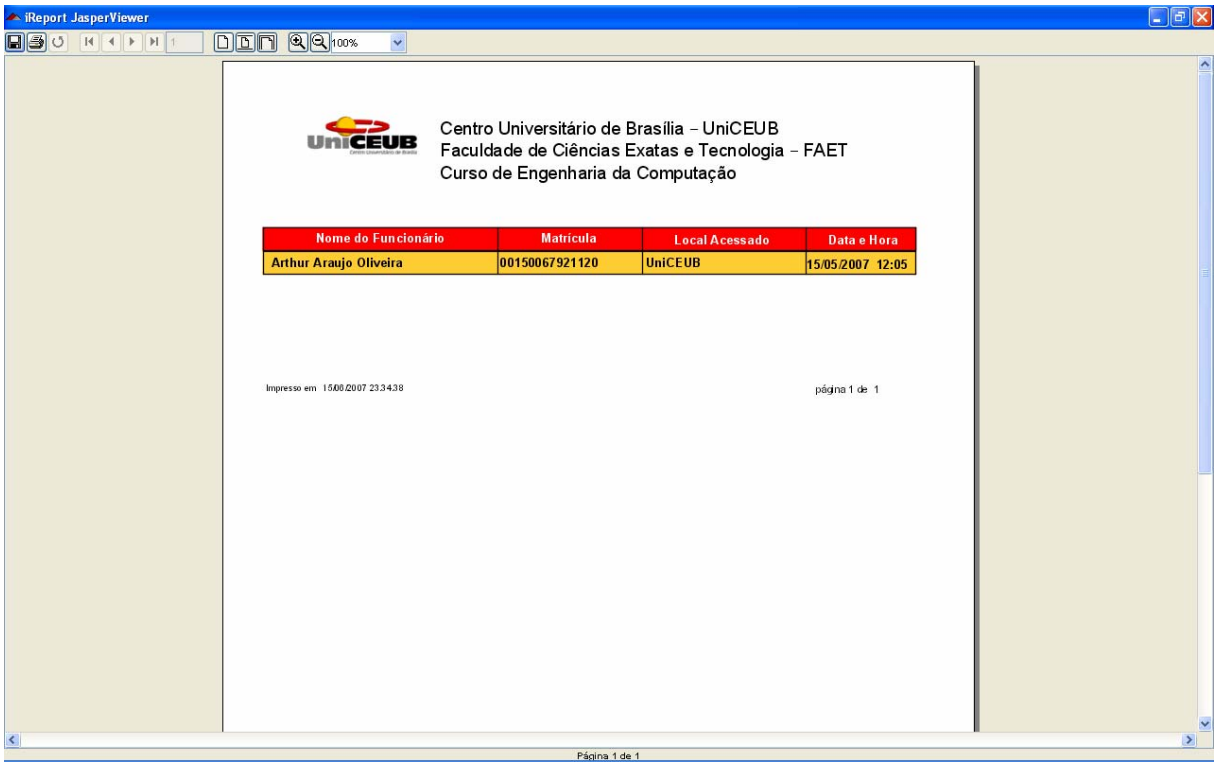

Figura 4.21 – Relatório Gerado

CAPÍTULO 5 – CONCLUSÃO

5.1 Resultados e Considerações Finais

De acordo com o proposto pelo projeto, o desenvolvimento do software e da parte física foram concluídas com êxito. O enriquecimento teórico e prático bem como todo o conhecimento obtido durante o curso de graduação, se mostrou eficaz ao final do desenvolvimento.

As maiores dificuldades encontradas durante o desenvolvimento do projeto foram: a integração dos componentes com o Kit de desenvolvimento CW552, a transmissão e recepção de dados na porta serial e a configuração da comunicação serial com o Java. Esses obstáculos foram ocasionados pelo pouco conhecimento na linguagem C e endereçamentos do microcontrolador 80c552, juntamente pela falta de experiência na configuração da porta serial no Java.

Por fim, espero que este projeto possa ser utilizado como referência aos alunos do curso de Engenharia da Computação que tiverem interesse em realizar projetos sobre controle de acesso, sistemas web e microcontroladores.

5.2 Propostas Futuras

Como propostas futuras pode-se citar a criação de um controle de acesso e um sistema web utilizando um microcontrolador PIC, como por exemplo o PIC 18 e PIC 24, que fornecem os serviços dos protocolos TCP/IP, utilizados em um sistema web. Neste novo sistema web criado pode-se acrescentar novos módulos, como: geração do código de barras, perfil de acesso e segurança.

REFERÊNCIAS BIBLIOGRÁFICAS

EAN **[Home Page].** 2007. Disponível em: http://www.eanbrasil.org.br. Acessado em: 1 Junho de 2007.

Eclipse **[Home Page].** 2006. Disponível em: http://www.eclipse.org/downloads/index.php. Acessado em 10 out. 2006.

ELS, Rudi van**. Sistema de desenvolvimento para microcontroladores CW552**, CONTROLWARE Automação Comercial, Versão 7, Brasília 2001.

FURGERI, Sérgio. **Java 2 Ensino Didático**. São Paulo: Editora Érica, 2002.

Guedes, Gilleanes T. A. **Guia de Consulta Rápida UML**. São Paulo: Novatec Editora Ltda, 2005.

Java Free **[Home Page].** 2006. Disponível em: http://www.javafree.org/content/view.jf?idContent=22. Acessado em: 20 out.2006.

Linha de Código **[Home Page].** 2007. Disponível em: http://www.linhadecodigo.com.br/artigos?id_ac=655. Acessado em: 9 maio de 2007.

Manzano, José Augusto N.G. **MySQL 5 interativo. Guia Básico de Orientação e Desenvolvimento**. São Paulo: Editora Érica 1ª edição, 2007.

Mundo Java **[Home Page].** 2007. Disponível em: http://www.mundojava.com.br/NovoSite/14materiacapa.shtml. Acessado em: 9 maio de 2007.

Nicolosi, Denys E. C. **Microcontrolador 8051 com Linguagem C.** São Paulo: Editora Érica 1ª edição, 2005.

Nicolosi, Denys E. C. **Microcontrolador 8051 detalhado.** São Paulo: Editora Érica 5ª edição, 2004.

Portal Java **[Home Page].** 2006. Disponível em: http://www.portaljava.com.br/home//modules.php?name=Content&pa=showpage&pid=51. Acessado em: 14 set. 2006.

Sun Microsystens **[Home Page].** 2007. Disponível em: http://java.sun.com/products/javacomm/index.jsp. Acessado em: 15 Jan. 2007.

VASCONCELOS, Laércio. **Hardware Total**. São Paulo: Makron Books, 2002.

Apêndice A – Configuração Serial do Java

A SUN fornece como download gratuito a API de comunicação serial e paralela na URL do Java:

http://java.sun.com/products/javacomm/index.jsp

Basta baixar a API e realizar os procedimentos de instalação. Após baixar a API, descompactá-la, você terá:

- Copiar o arquivo win32com.dll para o diretório C:\JavaSDK\JRE\BIN (isto é, o diretório onde o J2SDK foi instalado no seu PC).
- Copiar o arquivo comm.jar para o diretório C:\JavaSDK\ JRE\BIN \LIB.
- Copiar o arquivo javax.comm.properties para o diretório C:\JavaSDK\ JRE\BIN\LIB.
- Em seguida configure o CLASSPATH para que ele reconheça o arquivo comm.jar.

Apêndice B – Código-Fonte – Classe SComm.java

```
package br.com.projetofinal.sca.util;

import java.io.InputStream;
import java.io.OutputStream;

import javax.comm.CommPortIdentifier;
import javax.comm.SerialPort;
import javax.comm.SerialPortEvent;
import javax.comm.SerialPortEventListener;

public class SComm implements Runnable,SerialPortEventListener{

   //  propriedades
   private String Porta;
   public String Dadoslidos;
   public int nodeBytes;
   private int baudrate;
   private int timeout;
   private CommPortIdentifier cp;
   private SerialPort porta;
   private OutputStream saida;
   private InputStream entrada;
   private Thread threadLeitura;

   private boolean Escrita;

   //  construtor default paridade : par
   //  baudrate: 9600 bps stopbits: 2 COM 4
   public SComm() {
       Porta = "COM4";
       baudrate = 9600;
       timeout = 1000;
   }

  //  um Objeto ComObj é passado ao construtor
  //  com detalhes de qual porta abrir
  //  e informações sobre configurações
  public SComm( String p , int b , int t ){
      this.Porta = p;
      this.baudrate = b;
      this.timeout = t;
  }

  //habilita escrita de dados
  public void HabilitarEscrita(){
      Escrita = true;
  }

  //habilita leitura de dados
  public void HabilitarLeitura(){
      Escrita = false;
  }
```

```
 //  Obtém o ID da PORTA
 public void ObterIdDaPorta(){
     try {
         cp = CommPortIdentifier.getPortIdentifier(Porta);
         if ( cp == null ) {
             System.out.println("A " + Porta + " nao existe!" );
             System.exit(1);
         }
     } catch (Exception e) {
             System.out.println("Erro durante o procedimento. STATUS" + e );
             System.exit(1);
     }
 }

//  Abre a comunicação da porta
public void AbrirPorta(){
    try {
        porta = (SerialPort)cp.open("SComm",timeout);
        System.out.println("Porta aberta com sucesso!");

        //  configurar parâmetros
        porta.setSerialPortParams(baudrate, SerialPort.DATABITS_8,
                SerialPort.STOPBITS_2,SerialPort.PARITY_NONE);
    } catch (Exception e) {
        System.out.println("Erro ao abrir a porta! STATUS: " + e );
        System.exit(1);
    }
}

//  função que envie um bit para a porta serial
public void EnviarUmaString(String msg){
    if (Escrita==true) {
        try {
            saida = porta.getOutputStream();
            System.out.println("FLUXO OK!");
        } catch (Exception e) {
            System.out.println("Erro.STATUS: " + e );
        }
        try {
            System.out.println("Enviando um byte para " + Porta );
            System.out.println("Enviando : " + msg );
            saida.write(msg.getBytes());
            Thread.sleep(100);
            saida.flush();
        } catch (Exception e) {
            System.out.println("Houve um erro durante o envio. ");
            System.out.println("STATUS: " + e );
            System.exit(1);
        }
    } else {
        System.exit(1);
    }
}
```

```
//leitura de dados na serial
public void LerDados(){
    if (Escrita == false){
        try {
            entrada = porta.getInputStream();
            System.out.println("FLUXO OK!");
        } catch (Exception e) {
            System.out.println("Erro.STATUS: " + e );
            System.exit(1);
        }
        try {
            porta.addEventListener(this);
            System.out.println("SUCESSO. Porta aguardando...");
        } catch (Exception e) {
            System.out.println("Erro ao criar listener: ");
            System.out.println("STATUS: " + e);
            System.exit(1);
        }
        porta.notifyOnDataAvailable(true);
        try {
            threadLeitura = new Thread(this);
            threadLeitura.start();
        } catch (Exception e) {
            System.out.println("Erro ao iniciar leitura: " + e );
        }
    }//if
}

//  método RUN da thread de leitura
public void run(){
    try {
        Thread.sleep(5000);
    } catch (Exception e) {
        System.out.println("Erro. Status = " + e );
    }
}

//  função que fecha a conexão
public void FecharCom(){
    try {
        porta.close();
        System.out.println("CONEXAO FECHADA>>FIM..");
    } catch (Exception e) {
        System.out.println("ERRO AO FECHAR. STATUS: " + e );
        System.exit(0);
    }
}
//zerar variaveis
public void reset(){
    teclado = new StringBuffer();
    dadosLidos = new StringBuffer();
    porta.removeEventListener();
}

//  Acessores
public String obterPorta(){
    return Porta;
}
public int obterBaudrate(){
    return baudrate;
}
```

```
//  gerenciador de eventos de leitura na serial
//int count = 0;
StringBuffer dadosLidos = new StringBuffer();
StringBuffer teclado = new StringBuffer();

public void serialEvent(SerialPortEvent ev){
    switch (ev.getEventType()) {
    case SerialPortEvent.BI:
    case SerialPortEvent.OE:
    case SerialPortEvent.FE:
    case SerialPortEvent.PE:
    case SerialPortEvent.CD:
    case SerialPortEvent.CTS:
    case SerialPortEvent.DSR:
    case SerialPortEvent.RI:
    case SerialPortEvent.OUTPUT_BUFFER_EMPTY:
        break;
    case SerialPortEvent.DATA_AVAILABLE:
        byte[] bufferLeitura = new byte[20];
        try {
            while ( entrada.available() > 0 ) {
                nodeBytes = entrada.read(bufferLeitura);
            }
            String[] binarios = new String[nodeBytes];

            for (int i=0; i < nodeBytes; i++) {
                //converte os bytes lidos em binario
               binarios[i] = Integer.toBinaryString(bufferLeitura[i]);
               int teclas = bufferLeitura[i];
               teclado.append(teclas);
            }// for

  String Dadoslidos = new String(bufferLeitura);
  dadosLidos.append(Dadoslidos.trim());

} catch (Exception e) {
    HabilitarEscrita();
    EnviarUmaString("<");
    System.out.println("Erro durante a leitura: " + e );
}

//verifica codigo de barras
if(dadosLidos.indexOf("L") > -1){
    System.out.println(" >>>>> Codigo de barras lido");
    String matricula = dadosLidos.toString();
    matricula = matricula.replaceAll("L","");
    System.out.println(" >>>>> Matricula = " + matricula);
    HabilitarEscrita();
    EnviarUmaString(">");
    bufferLeitura = null;
    Dadoslidos = new String();
    reset();
    HabilitarLeitura();
    LerDados();
}
```

```
        //verifica teclado
        if(dadosLidos.indexOf("U") > -1){
            System.out.println(" >>>>> Teclado Alfanumerico lido");
            String senha = teclado.toString().substring(0,6);
            System.out.println(" >>>>> Senha = " + senha);
            HabilitarEscrita();
            EnviarUmaString("+");
            bufferLeitura = null;
            Dadoslidos = new String();
            reset();
            HabilitarLeitura();
            LerDados();
        }
        break;
    }

}

  //  função que fecha a conexão
  public void FecharCom(){
      try {
          porta.close();
          System.out.println("CONEXAO FECHADA>>FIM..");
      } catch (Exception e) {
          System.out.println("ERRO AO FECHAR. STATUS: " + e );
          System.exit(0);
      }
  }
  //zerar variaveis
  public void reset(){
      teclado = new StringBuffer();
      dadosLidos = new StringBuffer();
      porta.removeEventListener();
  }

  //  Acessores
  public String obterPorta(){
      return Porta;
  }
  public int obterBaudrate(){
      return baudrate;
  }

}
```

Apêndice C – Código-Fonte – Principal_Acesso.c

```
#include <reg552.h>
#include "rot_lcd.c"

/*  saida   pinos linha
 pl0 =    tek1    [123]   linhal
 pl2 =    tek3    [456]   linha2
 pl4 =    tek5    [789]   linha3
 pl6 =    tek7    [*0#]   linha4

 entrada pino  coluna
 pl1 =    tek2    [369#]  colunl
 pl3 =    tek4    [2580]  colun2
 pl5 =    tek6    [147*]  colun3
*/

sbit at 0x90 P1_0;
sbit at 0x91 P1_1;
sbit at 0x92 P1_2;
sbit at 0x93 P1_3;
sbit at 0x94 P1_4;
sbit at 0x95 P1_5;
sbit at 0x96 P1_6;
sbit at 0x97 P1_7;
sbit at 0xC0 P4_0;
sbit at 0xC1 P4_1;
sbit at 0xC2 P4_2;
sbit at 0xC3 P4_3;

bit tec_ok, tec_liv;
bit bit_reset=0;

#define num_enter       0xe0
#define num_enter2      0x5a
#define w_en            P4_0   // Pino 3 LCD   HLCD
#define s_data          P4_2   // Pino 2 4094  CLOCK
#define s_clock         P4_1   // Pino 3 4094  DATA
#define s_strobe        P4_3   // Pino 1 4094  STROBE
#define baud            0xFD   // 100h-(11590000/(12*32*9600))
#define kclock          INT0   // kit INT0 - pino ps2 clock
#define kdata           T0     // kit T0   - pino ps2 data
#define linhal          P1_0
#define linha2          P1_2
#define linha3          P1_4
#define linha4          P1_6
#define colunal         P1_1
#define coluna2         P1_3
#define coluna3         P1_5

unsigned int tec_buf;
unsigned int tec;

char codbar[14],senha[6];
char c;

int conth;
int c_mat; //contador do tam da matriz.

void tec4x3(void);
```

```
void putchar (unsigned char a)  // manda caractere na porta serial
{
 while(!TI);  // espera ate completar ultima transmissao
 SOBUF=a;     // manda caractere para buffer transmissao
 TI=0;        // limpa o flag de transmissao
}

unsigned char getchar (void)  // rotina que espera caractere na porta serial
{
 while(!RI);     // espere receber caractere no buffer de recepcao
 RI=0;           // limpa flag de recepcao
 return(SOBUF);  // retorna caractere recebido
}

void scan(void)
{
 unsigned char a;
 tec_ok=1;
 linha1=0; linha2=1; linha3=1; linha4=1;
 a=P1;
 a=a & 0x2a;
 if (a!=0x2a) tec_buf= 0x100|a;
 else
 {
  linha1=1; linha2=0; linha3=1; linha4=1;
  a=P1;
  a=a & 0x2a;
  if (a!=0x2a) tec_buf= 0x200|a;
  else
  {
   linha1=1; linha2=1; linha3=0; linha4=1;
   a=P1;
   a=a & 0x2a;
   if (a!=0x2a) tec_buf=0x300|a;
   else
   {
    linha1=1; linha2=1; linha3=1; linha4=0;
    a=P1;
    a=a & 0x2a;
    if (a!=0x2a) tec_buf=0x400|a;
    else tec_ok=0;
   }
  }
 }
}

unsigned int scancode(void)
{
  unsigned int k;
  unsigned int i;
  k=0;
  for (i=0;i!=11;i++)
  {
    while(kclock==1);
    if (kdata==1) k=k+0x400;//
    k = k >> 1;
    while(kclock==0);
   }
  k = k & 0x00FF;
  return(k);
}
```

```
unsigned int scantec(void)
{
 unsigned int k1,l,m;
 k1=scancode();
 l=scancode();
 m=scancode();
 return (k1);
}

char gettec(char c)
{
  char s;
  switch(c)
  {
    case 0xe0: s='E'; break;  case 0x5a: s='E'; break;
    case 0x45: s='0'; break;  case 0x16: s='1'; break;  case 0x1E: s='2'; break;
    case 0x26: s='3'; break;  case 0x25: s='4'; break;  case 0x2E: s='5'; break;
    case 0x36: s='6'; break;  case 0x3D: s='7'; break;  case 0x3E: s='8'; break;
    case 0x46: s='9'; break;
    default: s='N';
  }
  return(s);
}

unsigned char get_tec(void)
{
 unsigned char a;
 switch(tec_buf)
 {
  case 0x128:a='3';break;
  case 0x122:a='2';break;
  case 0x10a:a='1';break;
  case 0x228:a='6';break;
  case 0x222:a='5';break;
  case 0x20a:a='4';break;
  case 0x328:a='9';break;
  case 0x322:a='8';break;
  case 0x30a:a='7';break;
  case 0x428:a='#';break;
  case 0x422:a='0';break;
  case 0x40a:a='*';break;
 }
 tec_ok=0;
 return(a);
}
delay4x3(unsigned int ms)
{
  unsigned int tms=0;
  while(tms!=ms)
  {
    TL0=((65535-1000) & (23));
    TH0=((65535-1000)>>8);
    TR0=1;
    while (!TF0){}
    TF0=0;
    TR0=0;
    tms++;
  }
}
```

```
void enviaSenha(void)
{
   char s=0;
   putchar(senha[0]);
   putchar(senha[1]);
   putchar(senha[2]);
   putchar(senha[3]);
   putchar(senha[4]);
   putchar(senha[5]);
   putchar('U');

   wr_ctr_lcd(1);
   lcd_str(">>>> AGUARDANDO....");
   while(1)
   {
     while(RI)// verifica a se existe informação na porta serial
     {
       s=getchar();//recebe o dado da serial
       if (s=='+')///validado
       {
         wr_ctr_lcd(1);
         lcd_str(">>>> ENTRADA LIBERADA, SEJA BEM VINDO");
         delay4x3(6000);
         main();
       }
       else if (s=='!')///não validado
       {
         wr_ctr_lcd(1);
         lcd_str(">>>> ENTRADA NAO AUTORIZADA <<<<< ");
         delay4x3(6000);
         main();
       }
     }
   }

}
void pega_senha(unsigned char i)
{
  senha[conth]=i;

  if (conth != 6){
    conth++;
    wr_lcd('*');
  }

  if (conth > 6){
    conth=0;
    wr_ctr_lcd(1);
    lcd_str(" >>>>> SENHA INVALIDA <<<<< ");
    delay4x3(6000);
    main();
  }


}
```

```
void tec4x3(void)
{
 while(1)
  {
    scan();
    if (!tec_ok) tec_liv=1;
    if ((tec_ok)&(tec_liv))
    {
      tec_liv=0;
      c=get_tec();
      switch(c)
      {
        case '3':{pega_senha(3);break;}
        case '2':{pega_senha(2);break;}
        case '1':{pega_senha(1);break;}
        case '6':{pega_senha(6);break;}
        case '5':{pega_senha(5);break;}
        case '4':{pega_senha(4);break;}
        case '9':{pega_senha(9);break;}
        case '8':{pega_senha(8);break;}
        case '7':{pega_senha(7);break;}
        case '#':{enviaSenha();break;}//Enter e liberar entrada
        case '0':{pega_senha(0);break;}
        case '*':{main();break;}
      }

    }
    delay4x3(50);
  }
}
```

```
 void reset(void)
 {
   IE0=0;           // deshabilita todas interrupcoes
   TCON=0;          // controle das interrupçoes externas e do temporizador
   TMOD=0x21;       // configura timer1 no modo 2 e timer 0 no modo 1
   SOCON=0x52;      // configura comunicacao serial
   TH1=baud;        // configura velocidade de transmissao
   TR1=0;           // desliga timer 1 se estiver ligado
   TR1=1;           // liga timer 1
   kclock=1;
   kdata=1;
   bit_reset=1;
 }
```

```
void main(void)
{
  char s;
  unsigned int i;
  for (i=0;i!=15;i++)
    codbar[i]=0;

  if (bit_reset==0)
    reset();

  tec_ok=0;
  wr_ctr_lcd(1);
  lcd_str(">>>> PASSE O CARTAO DE ACESSO <<<<<");
  c_mat=0;
  conth=0;
```

```
goto_lcd(2,1);

while(1)
{
  if (kclock==0)
  {

    tec=scantec();
    s=gettec(tec);
    wr_lcd(s);
    putchar(s);
    codbar[c_mat]=s;//pega 14 posiçoes - matricula do Funcionário
    c_mat++;
    if (c_mat==14)
    {
       putchar('L');//Caracter para simbolizar enter do teclado
       c_mat=0;
    }
  }

  while(RI)// verifica a se existe informação na porta serial
  {
    c=getchar();//recebe o dado da serial

    if (c=='>'){
      wr_ctr_lcd(1);
    lcd_str(">>>> CARTAO AUTORIZADO <<<<");
    delay4x3(6000);
    wr_ctr_lcd(1);
        lcd_str("DIGITE A SENHA: '#'-Entrar '*'-Cancelar ");
        goto_lcd(2,1);
        tec4x3();
    }

       if (c=='<')
       {
        wr_ctr_lcd(1);
        lcd_str(">>>> ERRO INESPERADO,CONTATE O RH");
        delay4x3(6000);
        main();
       }else if (c=='-'){
         wr_ctr_lcd(1);
         lcd_str(">>>> CARTAO NAO CADASTRADO,CONTATE O RH");
         delay4x3(6000);
         main();
       }else if (c=='?'){
         wr_ctr_lcd(1);
         lcd_str(">>>> CARTAO BLOQUEADO,CONTATE O RH");
         delay4x3(6000);
         main();
       }


    }
  }
}
```

Apêndice D – Esquema Elétrico do Projeto