

4 MODELAGEM DA APLICAÇÃO J2ME

O Sistema de Apoio Policial tem como objetivo facilitar o trabalho da Polícia Militar. Utilizando tecnologias recentes descritas nos capítulos anteriores, este sistema propõe uma solução para o problema de acesso às informações de veículos¹ para a Polícia. Através de um celular com suporte à Java e acesso à web (e a um banco de dados), os policiais obterão com maior rapidez as informações de que necessitam.

4.1 Artefatos de Requisitos

Basicamente, o sistema possuirá um software que através do Login/Senha fornecidos se conecta a web e ao banco de dados. Com a autenticação feita, o policial escolherá o tipo de consulta que será feita: Placa ou Chassi. Após a escolha ele deverá digitar ou o número do chassi (caso for consulta por chassi) ou o número da placa (caso seja consulta por placa). Em seguida, a consulta será feita no banco de dados e na tela retornarão as informações relacionadas àquela placa (ou chassi). Terminada a pesquisa, a conexão e a aplicação são encerradas pelo policial ou uma nova consulta poderá ser realizada. As Tabelas 4.1 e 4.2 mostram as funções e os atributos do sistema.

¹ Entenda-se por veículos motos, carros, caminhões, enfim, meios de transporte terrestre.

Tabela 4.1 Funções para o Sistema de Apoio Policial.

Ref#	Função	Categoria
R1.1	Iniciar a Aplicação	Evidente
R1.2	Efetuar Login	Evidente
R1.3	Conectar-se à web	Escondida
R1.4	Validar Login através do banco de dados	Escondida
R1.5	Escolher tipo de consulta	Evidente
R1.6	Consultar Placa	Evidente
R1.7	Consultar Chassi	Evidente
R1.8	Fazer a busca no banco de dados	Escondida
R1.9	Retornar Consulta	Evidente
R1.10	Encerrar Conexão ou Realizar Nova Consulta	Evidente

Tabela 4.2 Atributos para o Sistema de Apoio Policial.

Atributos	Detalhes e Restrições de Contorno	
Tempo de Resposta	Restrição de Contorno	Durante a busca das informações no Banco de Dados, o retorno destas está previsto para dez segundos, porém devido a fatores já comentados em capítulos anteriores, esse valor poderá variar para mais ou para menos.
Metáfora de Interface	Detalhe	Utilização de Formulários simples e práticos.
Plataforma do S.O.	Detalhe	Sistema Operacional Nativo do Celular

4.2 Diagramas de Caso de Uso

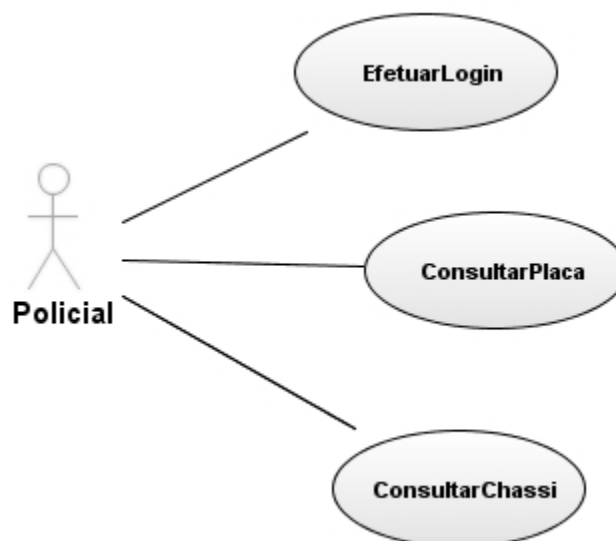


Figura 4.1 Diagrama de Caso de Uso – NetBeansIDE5.5

O público que utilizará este sistema é variado, alguns leigos e outros não e, portanto é necessário levar esse item em consideração, buscando desenvolver uma aplicação de fácil usabilidade para o usuário final. Desta forma serão utilizadas mensagens e telas simples com formulários, tanto para digitação, quanto para o retorno das informações.

No Caso de Uso EfetuarLogin, o Policial, devido à necessidade de buscar mais informações, inicia a aplicação no celular, sendo mostrado em seu display a tela inicial onde deverá digitar o Login/Senha e apertar o botão conectar. Na ação de apertar o botão conectar, o software mostrará mensagem automática de conexão à rede e as informações dos campos serão validadas através da conexão ao banco de dados. Se o Login/Senha existir no banco de dados, uma tela de confirmação será retornada. Caso Login/Senha estejam incorretos ou não existam no banco de dados, uma tela de erro será retornada ao usuário.

Caso de Uso	<u>EfetuarLogin</u>
Ator	Policial
Propósito	Iniciar a Aplicação. Digitar Login/Senha. Conectar-ser à web e validar Login através do banco de dados.
Descrição	O Policial, através do teclado do celular entra na aplicação. Na tela inicial estarão os campos de Login/Senha para serem preenchidos com o nome de usuário e senha do Policial. Ao apertar o botão conectar, o software emite mensagem de conexão a web, o policial deverá apertar o botão indicado para confirmar. Em seguida o programa conecta-se ao banco de dados e valida os campos Login/Senha, permitindo acesso ao sistema. É retornada então uma tela para escolher o tipo de consulta caso a validação esteja correta. Caso esteja incorreta, uma mensagem de erro é retornada e uma nova autenticação deverá ser feita.
Tipo	Principal e Essencial
Referência	R1.1, R1.2, R1.3, R1.4.

Típica Seqüência de Eventos

<u>Ação do Ator</u>	<u>Resposta do Sistema</u>
1. Este caso de uso começa quando o Policial percebe a necessidade de obter mais informações sobre determinado automóvel.	
2. O Policial inicia a aplicação J2ME através do Celular.	3. É mostrada a tela inicial no display com os campos Login/Senha para serem preenchidos.
4. O Policial digita então seu nome de usuário e senha e aperta o botão conectar.	5. A aplicação se encarrega de mostrar a mensagem no display de confirmação de conexão à web.
6. O policial deve apertar o botão indicado para confirmar conexão à web.	7. Quando conectado, o programa validará os campos do formulário no banco de dados, verificando se o Login/Senha existem e se estão corretos. Caso esteja tudo certo, uma tela é retornada no display. Caso contrário uma mensagem de erro é retornada .
8. Se Login/Senha corretos, o policial passa para a próxima etapa. Caso haja problemas na autenticação, o Policial deverá digitar novamente seu nome de usuário e senha.	

No Caso de Uso ConsultarPlaca o Policial apenas digitará a placa do automóvel que deseja consultar. Quando clicar no botão buscar, o sistema fará a verificação de existência da placa no banco de dados. Se a placa existir, as informações no banco de dados retornarão na tela do celular em alguns segundos, caso não exista, uma mensagem de erro será retornada.

Caso de Uso	<u>ConsultarPlaca</u>
Ator	Policial
Propósito	Consultar um automóvel pelo número da placa.
Descrição	O Policial digita no campo Placa, o número da placa do automóvel. Após digitar deverá apertar o botão consultar. Nesta ação uma busca será feita no banco de dados. Se a placa digitada existir no banco, as informações referentes àquele veículo daquela placa serão retornadas em uma nova tela. Em seguida, o Policial poderá escolher entre fazer uma nova consulta apertando o botão indicado ou então encerrar a aplicação apertando o botão sair. Se a placa digitada não constar no banco, uma mensagem de erro é retornada.
Tipo	Principal e Essencial
Referência	R1.6, R1.8, R1.9, R1.10

Típica Seqüência de Eventos

<i>Ação do Ator</i>	<i>Resposta do Sistema</i>
1. Digitar o número da placa e apertar o botão consultar.	2. Realiza a busca no banco de dados verificando se a placa existe. Caso exista, é retornada uma nova tela com as informações do veículo daquela placa. Caso não exista, é retorna uma mensagem de erro.
3. Caso a consulta tenha retornado um erro, o Policial deverá tentar novamente o mesmo procedimento. Porém, caso a consulta tenha sido realizado com sucesso, o policial neste momento poderá encerrar a aplicação ou realizar uma nova consulta.	

No Caso de Uso ConsultarChassi o Policial apenas digitará o chassi do automóvel que deseja consultar. Quando clicar no botão buscar, o sistema fará a verificação de existência do chassi no banco de dados. Se o chassi existir, as informações no banco de dados retornarão na tela do celular em alguns segundos, caso não exista, uma mensagem de erro será retornada.

Caso de Uso	<u>ConsultarChassi</u>
Ator	Policial
Propósito	Consultar um automóvel pelo número do chassi.
Descrição	O Policial digita no campo Chassi, o número do chassi do automóvel. Após digitar deverá apertar o botão consultar. Nesta ação uma busca será feita no banco de dados. Se o chassi digitado existir no banco, as informações referentes àquele veículo daquele chassi serão retornadas em uma nova tela. Em seguida, o Policial poderá escolher entre fazer uma nova consulta apertando o botão indicado ou então encerrar a aplicação apertando o botão sair. Se o chassi digitado não constar no banco, uma mensagem de erro é retornada.
Tipo	Principal e Essencial
Referência	R1.7, R1.8, R1.9, R1.10

Típica Seqüência de Eventos

<i>Ação do Ator</i>	<i>Resposta do Sistema</i>
1. Digitar o número do chassi e apertar o botão consultar.	2. Realiza a busca no banco de dados verificando se o chassi existe. Caso exista, é retornada uma nova tela com as informações do veículo daquele chassi. Caso não exista, é retorna uma mensagem de erro.
3. Caso a consulta tenha retornado um erro, o Policial deverá tentar novamente o mesmo procedimento. Porém, caso a consulta tenha sido realizado com sucesso, o policial neste momento poderá encerrar a aplicação ou realizar uma nova consulta.	

4.3 Diagrama de Classes

Usuario
<i>Atributos</i> público String login público String senha
<i>Operações</i> público Usuario() público String getLogin() público void setLogin(String login) público String getSenha() público void setSenha(String senha)

*Figura 4.2 Diagrama de Classe:
Classe Usuario – NetBeansIDE5.5*

Veiculo
<i>Atributos</i> público String idchassi público String idplaca público String marcaModelo público String proprietario público String cidade público String estado público String cor público String especieTipo público String combustivel público String anoModFab público String situacao
<i>Operações</i> público Veiculo() público String getIdchassi() público void setIdchassi(String val) público String getIdplaca() público void setIdplaca(String val) público String getMarcaModelo() público void setMarcaModelo(String val) público String getProprietario() público void setProprietario(String val) público String getCidade() público void setCidade(String val) público String getEstado() público void setEstado(String val) público String getCor() público void setCor(String val) público String getEspecieTipo() público void setEspecieTipo(String val) público String getCombustivel() público void setCombustivel(String val) público String getAnoModFab() público void setAnoModFab(String val) público String getSituacao() público void setSituacao(String val)

Figura 4.3 Diagrama de Classe: Classe Veiculo – NetBeansIDE5.5

4.4 Diagramas de Seqüência

Nos diagramas de seqüência a seguir são mostradas as trocas de mensagens entre o Policial e o Sistema de Apoio Policial.

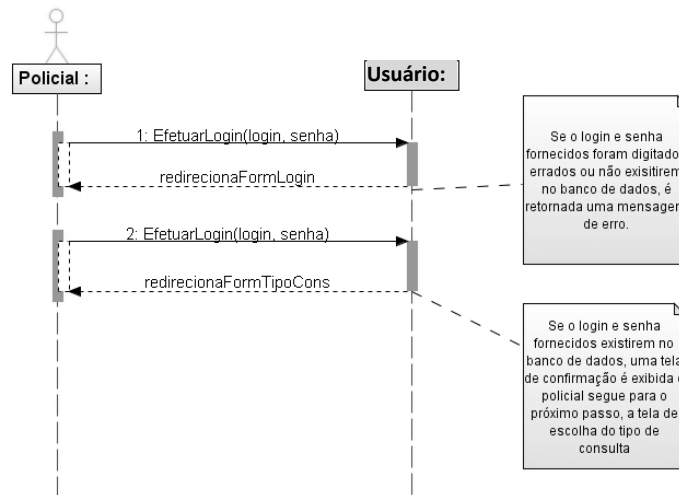


Figura 4.4 Diagrama de Seqüência: EfetuarLogin – NetBeansIDE5.5

Contrato	EfetuarLogin(login, senha)
Responsabilidades	Verificar se o usuário existe no banco de dados e liberar o acesso à consulta de placas e chassis. Se o usuário existe, retorna a tela para consulta. Se o usuário não existe, retorna mensagem de erro.
Tipo	Policiais
Referências	Funções R1.1, R1.2, R1.3, R1.4
Caso de Uso	EfetuarLogin
Notas	
Exceções	
Saída	
Pré-condições	O Policial já deve ter conhecimento do seu Login/Senha
Pós-condições	Policial conectado ao sistema para consultas.

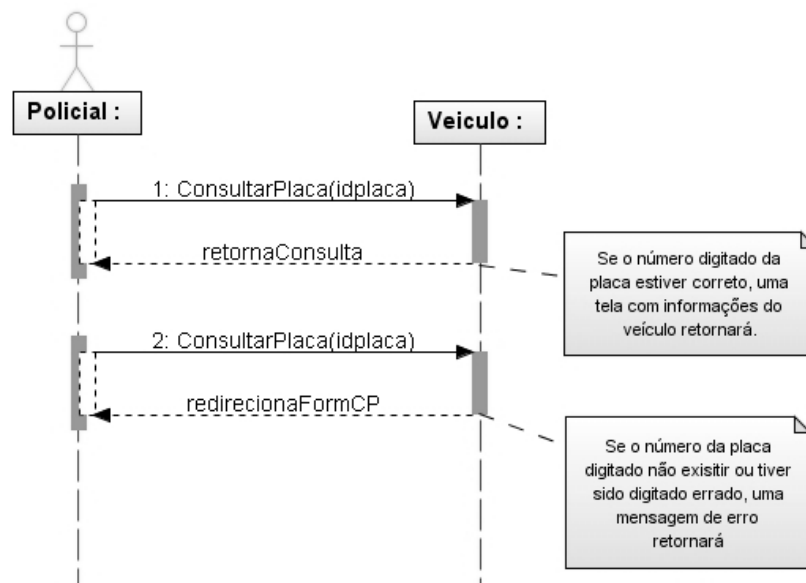


Figura 4.5 Diagrama de Seqüência: ConsultarPlaca – NetBeansIDE5.5

Contrato	ConsultarPlaca(idplaca)
Responsabilidades	Verificar se a placa existe no banco de dados. Retornar mensagem de erro caso não exista. Retornar informações do veículo caso exista.
Tipo	Veiculos
Referências	Funções R1.6, R1.8, R1.9, R1.10
Caso de Uso	ConsultarPlaca
Notas	
Exceções	
Saída	
Pré-condições	O Policial já deve ter estar logado no sistema e ter escolhido o tipo de consulta.
Pós-condições	Sair da aplicação. Realizar nova consulta. Refazer consulta.

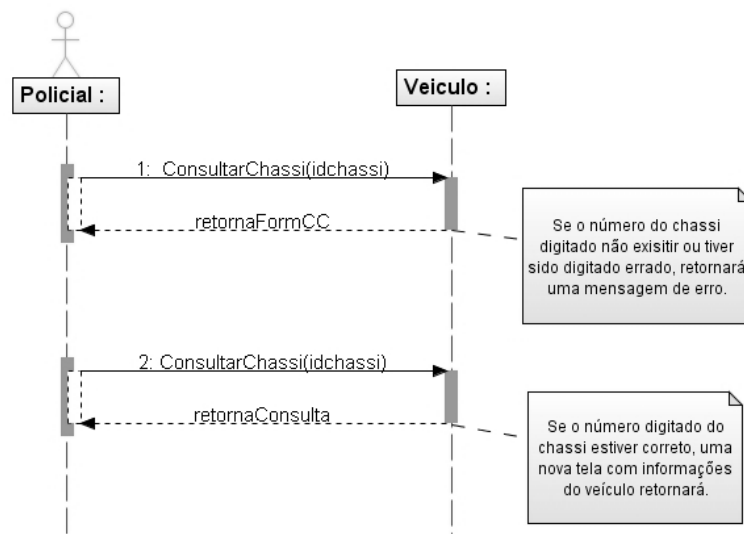


Figura 4.6 Diagrama de Seqüência: ConsultarChassi – NetBeansIDE5.5

Contrato	ConsultarChassi(idchassi)
Responsabilidades	Verificar se o chassi existe no banco de dados. Retornar mensagem de erro caso não exista. Retornar informações do veículo caso exista.
Tipo	Veiculos
Referências	Funções R1.7, R1.8, R1.9, R1.10
Caso de Uso	ConsultarChassi
Notas	
Exceções	
Saída	
Pré-condições	O Policial já deve ter estar logado no sistema e ter escolhido o tipo de consulta.
Pós-condições	Sair da aplicação. Realizar nova consulta. Refazer consulta.

4.5 Diagrama de Atividade

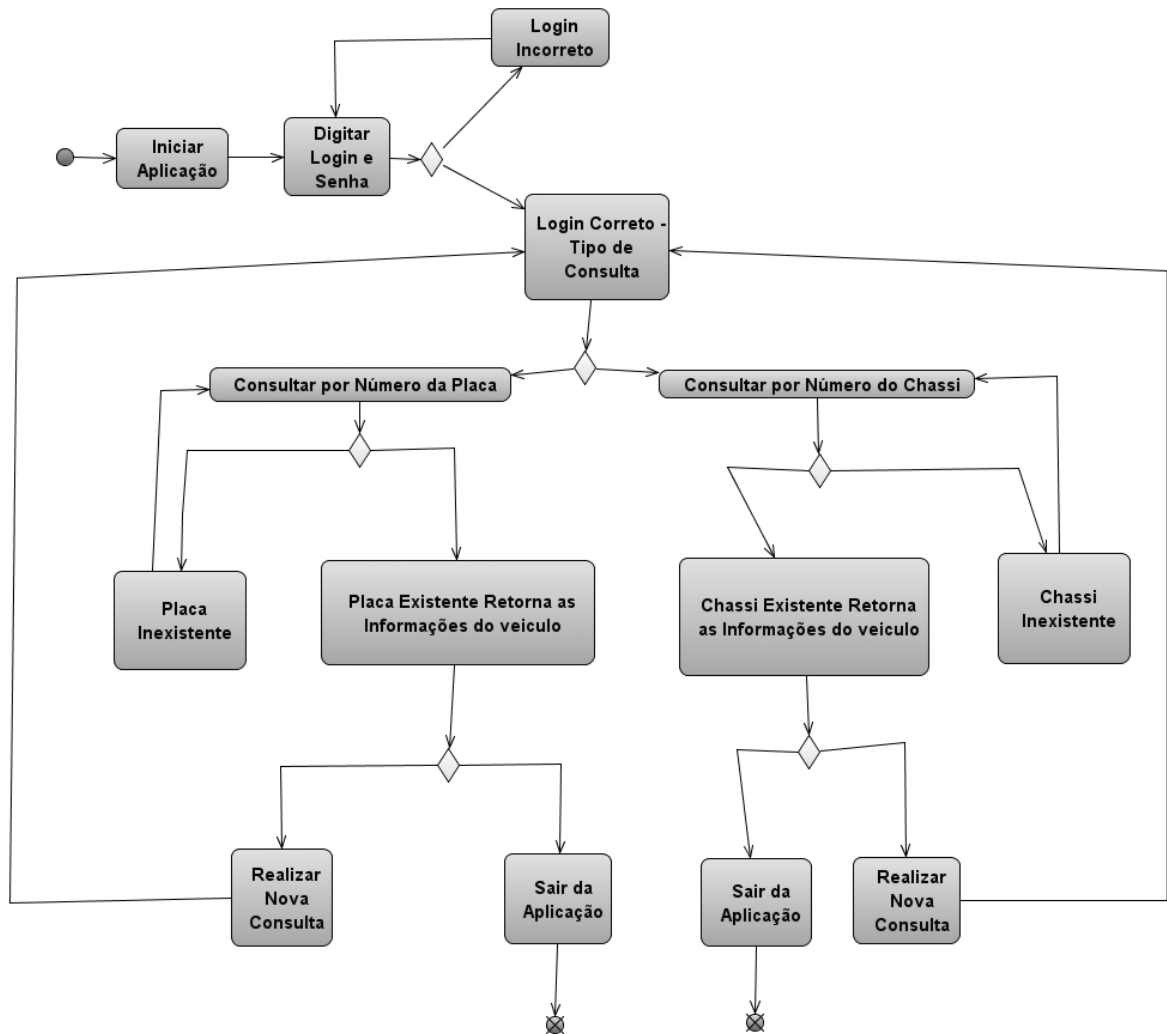


Figura 4.7 Diagrama de Atividade – NetBeansIDE5.5

4.6 Diagrama de Implementação

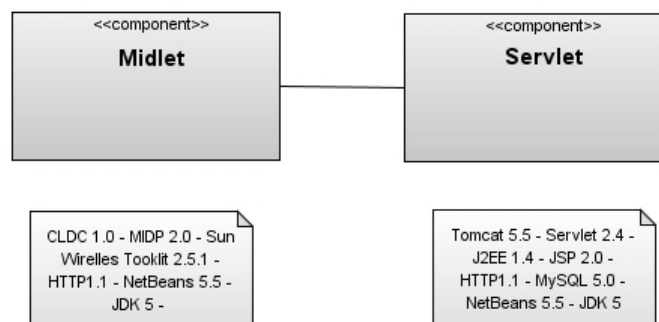


Figura 4.8 Diagrama de Componente – NetBeansIDE5.5

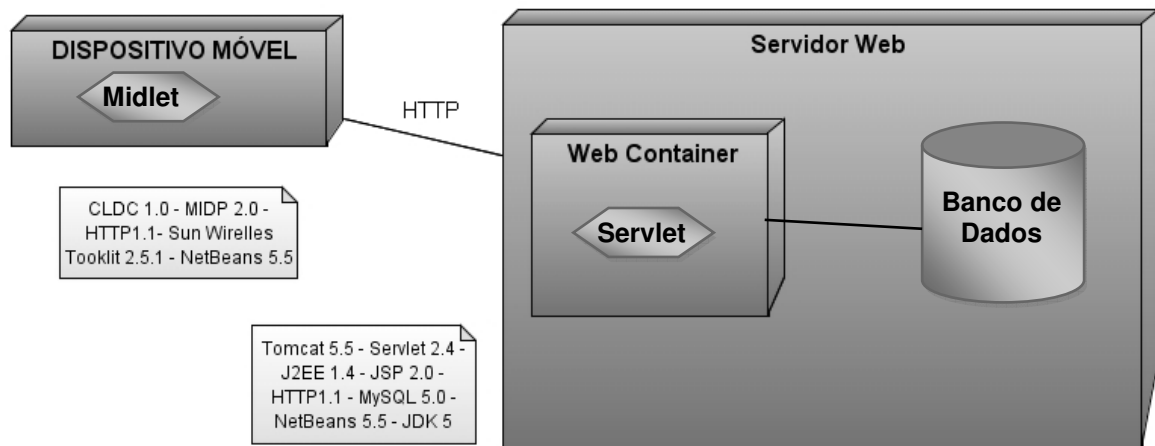


Figura 4.9 Diagrama de Implantação – NetBeansIDE5.5

4.7 Código

Esta seção abordará comentários sobre o código desenvolvido para a aplicação do servidor e também para a aplicação cliente.

Como mostrado nos diagramas de componentes e de implantação, de um lado, o cliente, foi utilizado um programa que simula um dispositivo móvel para a configuração CLDC 1.0 e o perfil MIDP 2.0. O Sun Wirelles Toolkit 2.5 é de uso obrigatório para outros programas, como o Netbeans, para desenvolvimento e simulação das aplicações móveis. Do outro lado foi utilizado um servidor web para aplicações Java, o web container Tomcat5 (suporte à Servlet 2.4, JSP 2.0, versão J2EE 1.4) trabalhando junto com o banco de dados MySQL 5.

A aplicação do lado cliente (aplicação móvel) foi desenvolvida no NetBeans5.5. A figura 4.12 mostra o flow design da aplicação desenvolvida através da opção Visual Midlet do NetBeans.

O Flow Design funciona como um fluxo da aplicação, tornando a programação mais produtiva. A aplicação cliente possui então, quatro formulários: login, escolha do tipo de consulta, consultar por placa e consultar por chassi. No formulário principal, foram incluídos dois botões, um para *sair* da aplicação e outro para *logar*. No formulário para escolha do tipo de consulta, existem quatro botões: *placa* – para ir ao formulário da consulta por placa, *chassi* – para ir ao formulário da consulta por chassi, *sair* – para sair da aplicação e *voltar* – para voltar à tela de login.

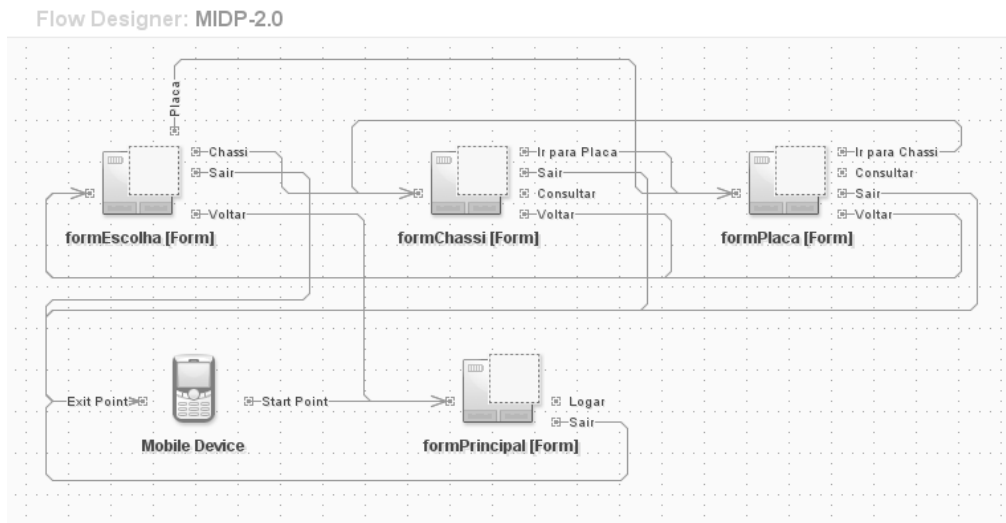


Figura 4.10 Flow Design – NetBeansIDE5.5

Quando o usuário escolher placa, o formulário para placa será chamado e mostrado no display. Este possui quatro botões: *sair* – para sair da aplicação, *voltar* – para voltar à tela de escolha do tipo de consulta, *consultar* – para fazer a busca no banco e *ir para chassi* – para escolher o formulário chassi. O formulário para o chassi é chamado quando o usuário clica no botão consultar por chassi. Assim como o formulário placa, este também possui quatro botões: *sair* – para sair da aplicação, *voltar* – para voltar à tela de escolha do tipo de consulta, *consultar* – para fazer a busca no banco e *ir para placa* – para escolher o formulário placa.

No lado servidor, um servlet foi desenvolvido utilizando também o NetBeans 5.5. O servlet recebe o pedido do cliente através de uma string url e a partir dela é possível tratar a comunicação com o banco de dados e também o envio da resposta para o cliente, tudo através da conexão HTTP.

De forma resumida, a seguir, é apresentado o que ocorre nesta conversa entre cliente e servidor nos códigos fontes de cada um.

Ao clicar no botão logar, a classe `ThreadHttpConnection` é chamada. No primeiro bloco `try/catch` ocorre a seguinte situação: se houver um erro de conexão HTTP ou no servidor, este erro será retornado ao cliente no display. Caso a conexão seja estabelecida, o servidor fará a comparação da string `command` enviada ao final da url para saber o que fazer.

APLICAÇÃO CLIENTE

```
//início da classe
1 private class ThreadHttpConnection extends Thread {

//a linha 2 envia a string url de conexão web via http (método post) para o servidor
2 public String url = "http://localhost:8084/ServerSAP/ServerSAP?command=fazerlogin";
:
:
:
```

APLICAÇÃO DO SERVIDOR

```
//início do método POST
protected void doPost(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException {

//abre o envio e recebimento de dados
    dis = new DataInputStream( ( InputStream ) request.getInputStream() );
    dos = new DataOutputStream( ( OutputStream ) response.getOutputStream() );

// define uma string chamada command para comparar a string que está chegando do cliente e com ela saber o
que fazer em seguida
    String command = request.getParameter( "command" );
    System.out.println( "\n String command recebida... " + command );

// se o comando que está chegando do cliente estiver vazio
    if ( ( "" ).equals( command ) ) {

// imprime no console
        System.out.println( "\n String command vazia" );
    }

// se o comando que está chegando do cliente é o comando fazer login
    else if ( ( "fazerlogin" ).equals( command ) ) {

:
:
:
```

Se a string enviada pelo cliente for a que corresponde a *fazerlogin*, então, o servlet fará o gerenciamento de sessão e buscará os dados no banco.

O gerenciamento de sessão deve sempre ser utilizado em aplicações deste tipo. Isto porque o protocolo HTTP não tem estado e também não possui nenhuma conexão persistente entre o cliente e o servidor. Portanto, a interação entre eles poderá ocorrer se o servidor reconhecer que os pedidos estão vindo do mesmo cliente. Neste trabalho utilizou-se a reescrita de url para sanar este problema. O servlet criará então apartir da string url fornecida uma nova sessão e um ID de sessão correspondentes para poder controlar os pedidos do cliente, isto é também conhecido como controle de sessão. Com isto, um cabeçalho é enviado

ao cliente e nele é indicado o novo url reescrito. Ao receber a resposta do servlet, o cliente procura por um cabeçalho indicando o novo url e após este passo, cada pedido do cliente, usando este novo url reescrito, faz com que seja possível para o servlet reconhecer quem iniciou o pedido (MUCHOW, 2004).

O código a seguir, desenvolvido para esta aplicação, mostra como o servlet e o midlet faz este tratamento.

```

APLICAÇÃO DO SERVIDOR
:
:
:
//inicia o bloco try/catch que trata da requisição do cliente
    try {
// obtém informações da sessão
        HttpSession session = request.getSession( true );
        System.out.println( "\n Session... " + session );
// se for uma sessão nova, reescreve o URL do cliente
        if ( session.isNew() ) {
// obtém o URL
            String incomingURL = request.getRequestURL().toString();
            System.out.println( "\n incomingURL... " + incomingURL );
// codifica e adiciona o ID de sessão no URL
            String URLwithID = response.encodeURL(incomingURL);
            System.out.println( "\n URLwithID... " + URLwithID );
// devolve um cabeçalho para o cliente, com o novo URL
            response.setHeader( "Custom-newURL", URLwithID );
            System.out.println( "\n URLwithID... " + URLwithID );
        }
:
:
:

```

Com a conexão estabelecida, o cliente envia os dados de login para que o servidor busque no banco de dados e verifique sua existência.

```

APLICAÇÃO CLIENTE
:
:
:
//inicio da thread
public void run() {
    HttpURLConnection http = null;

```

```

//bloco try/catch: envia dados para o servidor
try {

//abre a conexão http
    http = (HttpConnection)Connector.open(url, Connector.READ_WRITE);

//informações de cabeçalho exigido pelo método POST (headers ou cabeçalho normalmente tem um nome e um
valor que estão sempre no formato string)
    http.setRequestProperty("User-Agent", "Profile/MIDP-2.0 , Configuration/CLDC-1.0");
    http.setRequestProperty("Content-Language", "en-US");

//método POST
    http.setRequestMethod(HttpConnection.POST);

//abre a saída de dados
    dos = http.openDataOutputStream();

//envia a string login
    dos.writeUTF(login.trim());

//força o envio dos dados
    dos.flush();

//imprime no console a url e o login enviados ao servidor
    System.out.println( "\n Enviando String url: " + url );
    System.out.println( "\n Enviando String Login: " + login );
    System.out.println( "\n Conectando com o servidor para fazer o login" );
}
:
:
:

```

O servidor recebe a string e faz a busca no banco.

APLICAÇÃO DO SERVIDOR

```

:
:
:
//o servidor lê a string que veio do cliente
String login = dis.readUTF();

// escreve no console a string login que veio do cliente
    System.out.println( "\n String login recebida... " + login );

//bloco try/catch: busca dos dados no banco e imprime tudo no console
try {

// prepara a Statement
    ps = this.con.prepareStatement(sqla);
    System.out.println( "\n prepare statement... " + sqla );

```

```

// seta o valor "?" da String sql
ps.setString( 1, login );
System.out.println( "\n string login para consulta... " + login );

//executa a query de consulta
rs = ps.executeQuery();
System.out.println( "\n rs... " + rs );
System.out.println( "\n ps... " + ps );
}
:
:
:

```

Após a consulta, o servlet fará a verificação de null. Mesmo que o conteúdo de loginResult seja null, este resultado será enviado ao cliente, que se encarregará do tratamento adequado. Com o IF, é possível identificar quando ocorre o null na busca.

APLICAÇÃO DO SERVIDOR

```

:
:
:
:
//ser loginResult for igual a null então
if ( ( "" ).equals( loginResult ) ) {
    System.out.println( "\n if loginResult == null --> String loginResult tem o valor... " + loginResult );

    // retorna para o cliente o resultado da busca
    response.setContentType("text/plain");
    dos.writeUTF(loginResult); //saída de dados
    dos.writeUTF(senhaResult); //saída de dados
    System.out.println( "| Enviando loginResult: " + loginResult + " | Enviando senhaResult: " +
senhaResult );
    dos.flush(); //força o envio para o cliente
    dis.close(); // fecha a entrada de dados
    dos.close(); // fecha a saída de dados
}

//ser loginResult for diferente de null então
else {
    System.out.println( "\n if loginResult <> null --> String loginResult tem o valor... " + loginResult );

    // retorna para o cliente o resultado da busca
    response.setContentType("text/plain");
    dos.writeUTF(loginResult); //saída de dados
    dos.writeUTF(senhaResult); //saída de dados
    System.out.println( "| Enviando loginResult: " + loginResult + " | Enviando senhaResult: " +
senhaResult );
    dos.flush(); //força o envio para o cliente
    dis.close(); // fecha a entrada de dados
    dos.close(); // fecha a saída de dados
}

:
:
:

```

No cliente, primeiro é feito o controle de sessão:

```

APLICAÇÃO CLIENTE
:
:
:
//bloco try/catch: tratamento da resposta do servidor
try {

//abre a entrada de dados
    dis = http.openDataInputStream();

//bloco try/catch: gerenciamento de sessão
    try {
//obtem informações de cabeçalho
        String URLwithID = http.getHeaderField("Custom-newURL");
        System.out.println( "\n URLwithID: " + URLwithID );

//se o cabeçalho possui um URL reescrito, então:
        if(URLwithID != null) {
            url = URLwithID; //atualiza o URL para todos os pedidos futuros da servlet
            System.out.println( "\n Url: " + url );
        }
    }

//se qualquer erro ocorrer neste ponto, é mostrado um alert.
catch (IOException ex) {
    System.out.println( "\n Passei no catch Reescrita de URL..." + ex.toString() + " \n " );
    showAlert( " \n Foi encontrado o seguinte erro...\n" + ex.getMessage() + " \n " );
    getDisplay().setCurrent(get_alert1(), get_formPrincipal());
    ex.printStackTrace();
}

:
:
:

```

Em seguida é feita a leitura das strings que estão chegando do servidor. Se loginResult for igual a null, então o texto “Dados não encontrados” é setado no item “StringItem” do formulário, mostrando para o usuário que a busca retornou um erro. Caso loginResult for diferente de null, então um alert será mostrado com o resultado da busca.

```

APLICAÇÃO CLIENTE
:
:
:
try {
    loginResult = dis.readUTF();//entrada de dados
    senhaResult = dis.readUTF();//entrada de dados

```

```

        System.out.println( "\n String loginResult tem o valor... " + loginResult + " String senhaResult tem o
valor... " + senhaResult);

//se loginResult e senhaResult forem igual a null então imprime no console o que chegou e vai para o catch
    if ( "").equals(loginResult) && "").equals(senhaResult) ){
        System.out.println( "\n if loginResult == null --> String loginResult tem o valor... " + loginResult +
"String senhaResult tem o valor..." + senhaResult);
    }

//se loginResult e senhaResult forem diferente de null então
    else {
        try {
            System.out.println( "\n if loginResult < > null --> String loginResult tem o valor... " + loginResult
+ "String senhaResult tem o valor..." + senhaResult);
            alertResult = new Alert( "\n Seja Bem Vindo(a):\n ", "\n Bem Vindo(a): \n " + loginResult, null,
AlertType.CONFIRMATION) ;
            alertResult.setTimeout(Alert.FOREVER);
            getDisplay().setCurrent(alertResult, get_formEscolha());
            getDisplay().vibrate(1000);
            dos.close();//fecha a saída de dados
            dis.close();//fecha a entrada de dados
            http.close();//fecha a conexão http
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
}

//passa aqui quando loginResult e senhaResult for igual a null
catch (IOException ex) {
    System.out.println( "\n Passei no catch login result..." + ex.toString() + " valor de loginResult " +
loginResult);
    stringItemFormPrincipal.setLabel("Erro: ");
    stringItemFormPrincipal.setText("Dados não encontrados. Por favor, Faça seu login novamente!");
    getDisplay().setCurrent(get_formPrincipal());
    textFieldLogin.setString("");
    textFieldSenha.setString("");
    //ex.printStackTrace();
}

:
:
:

```

O código para consulta por placa e por chassi é semelhante, o conceito é o mesmo e, portanto não há necessidade de comentá-los. O código fonte completo da aplicação cliente e servidor encontra-se no CD que acompanha este trabalho.