Centro Universitário de Brasília
FATECS - Faculdade de Tecnologia e Ciências Sociais Aplicadas

Luiza Rocha Troncoso Gonçalves

SISTEMA DE COMPRA DE INGRESSOS DE CINEMA USANDO SMS

Brasília
2009

SISTEMA DE COMPRA DE INGRESSOS DE CINEMA USANDO SMS

por

Luiza Rocha Troncoso Gonçalves

Monografia apresentada à Banca examinadora
da FATECS – Faculdade de Tecnologia
e Ciências Sociais Aplicadas do UNICEUB
– Centro Universitário de Brasília como
requisito parcial para a obtenção de
Certificado de Conclusão de Curso de
Graduação, na área de Engenharia
de Computação.

Orientador: Prof. Gleyson Silva

Brasília
2009

SUMÁRIO

RESUMO

O acesso a shows, teatros, jogos e cinemas depende da aquisição de ingressos, os quais podem ser adquiridos por diversas formas. No caso dos cinemas, essa aquisição pode se dar por meio dos clássicos guichês de atendimento, via WEB ou por intermédio de guichês eletrônicos, os chamados totens. Enquanto a compra pela internet depende do acesso a um computador, as demais formas de aquisição de ingresso demandam o enfrentamento de longas filas.

Este trabalho tem por objetivo a construção de um protótipo que viabilize a compra de ingressos de cinema usando o serviço SMS, acessível à maioria das pessoas. Adicionalmente, a solução proposta pretende trazer benefícios às casas de cinema por meio da redução dos seu custos operacionais pela diminuição do número de funcionários que trabalham nos postos de atendimento ao público.

Constitui este trabalho a construção de um protótipo que viabiliza a compra de ingressos de cinema através do serviço de mensagens curtas (*Short Message Service* – SMS). A aplicação recebe a mensagem SMS e, após uma análise léxica e sintática, é efetivada a compra e devolvida uma resposta via SMS para o usuário. Todas essas operações são devidamente registradas.

O programa foi desenvolvido na linguagem de programação C# (C Sharp) utilizando-se da IDE Microsoft *Visual Studio* 2008. O Banco de Dados escolhido foi o Microsoft *SQL Server Express Edition* que utiliza comandos SQL e é o responsável pelo armazenamento dos dados. Para conectar-se ao banco de dados foi utilizada a ferramenta Microsoft *SQL Server Management Studio* 2008. A comunicação entre o celular e o servidor é provida pelo modem, que utiliza *AT Command* para isso.

Palavras – chave: Tecnologia GSM, serviço de mensagens curtas, celular.

ABSTRACT

The access to concerts, theaters, and sport events rely on a ticket purchasing process, which may happen in several ways. In the case of cinemas, the acquisition can occur through the traditional service counters, WEB or through electronic counters, called totems. While the purchase by the Internet depends on access to a computer, the other ways of purchasing tickets visit the face of long queues.

This paper aims at building a prototype that allows the purchase of tickets to the movie theaters using the SMS service which is a very popular tool. The intention of the proposed solution is to bring benefits for movie theaters by reducing their operational costs in decreasing the number of employees.

This work consists on the construction of a prototype system that enables the purchase of tickets through the short message service (Short Message Service - SMS). The application receives the SMS message, and after a lexical and syntactic analysis of it and because treatment is effective to purchase and return an answer via SMS to the user. All these operations are properly recorded.

The program was developed in the programming language C# using the Microsoft Visual Studio 2008 IDE. The database chosen was Microsoft SQL Server Express Edition that uses SQL commands and is responsible for storage of data. To connect to the database used was the Microsoft SQL Server Management Studio 2008. Communication between the mobile and the server is provided by modem, using AT Commands for it.

Key – words: GSM technology, short message service, mobile phones.

LISTA DE FIGURAS

LISTA DE TABELAS

ÍNDICE DE SIGLAS E ABREVIATURAS

AT	Hayes AT Commands
AuC	Authentication Centre
BSC	Base Station Controller
BSS	Base Station System
BTS	Base Transceiver Station
C#	C Sharp
CLR	The Common Language Runtime
CR	Carriage Return
EC	Echo Canceler
EDGE	Enhanced Data rates for GSM Evolution
EIR	*Equipment Identify Register*
ESME	External Short Message Entity
FCL	The Framework Class Libraries
G24	TCP/IP Wireless Data Modem
GPRS	General Packet Radio Service
GSM	Global System for Mobile communications
HLR	Home Location Register
IIS	Internet Information Service
IMEI	International Mobile Station Equipment Identity
IMSI	International Mobile Subscriber Identity
ISDN	Integrated Service Digital Network
IWF	InterWorking Function
Ki	Subscriber Authentication Key
LAI	Location Area Identity
LF	Line Feed
ME	Mobile Equipment
MS	Mobile Station
MSC	Mobile services Switching Centre
MSDN	Microsoft Development Network
MSISDN	Mobile Station Integrated Services Digital Network
NSS	Network Switching System

PDU	Protocol Data Unit
SIM	Subscriber Identity Module
SM – AL	Short-Message-Application-Layer
SM – LL	Short-Message-Link-Layer
SM – RL	Short-Message-Relay-Layer
SM – TL	Short-Message-Transfer-Layer
SME	Short Message Entity
SMS – GMSC	SMS gateway MSC
SMS – IWMSC	SMS InterWorking MSC
SMS	Short Message Service
SMSC	Short Message Service centre
SQL	Structured Query Language
TE	Terminal Equipment
TMSI	Temporary Mobile Subscriber Identity
TPDU	Transfer Protocol Data Unit
TP-MTI	TP-Message-Type-Indicator
TP-SRI	TP-Status-Report-Indication
UMTS	Universal Mobile Telecommunications System
USB	Universal Serial Bus
VLR	Visitor Location Register

1 INTRODUÇÃO

O serviço de mensagens curtas (*Short Message Service* – SMS) é um serviço simples que permite a troca de mensagens. A implantação do serviço de compra de ingresso via SMS visa explorar uma utilidade que advém desse serviço.

O presente projeto visa desenvolver um sistema para viabilizar a compra de ingresso usando SMS. Esse sistema vai verificar se tem disponibilidade de assento para a sessão especificada pelo usuário e enviar para este uma resposta via SMS. A aplicação vai gerar um registro das operações para que futuramente possa ser feita uma integração com o cinema e posterior débito em conta do cliente.

Abaixo, a Figura 1.1 ilustra a arquitetura simplificada do projeto:

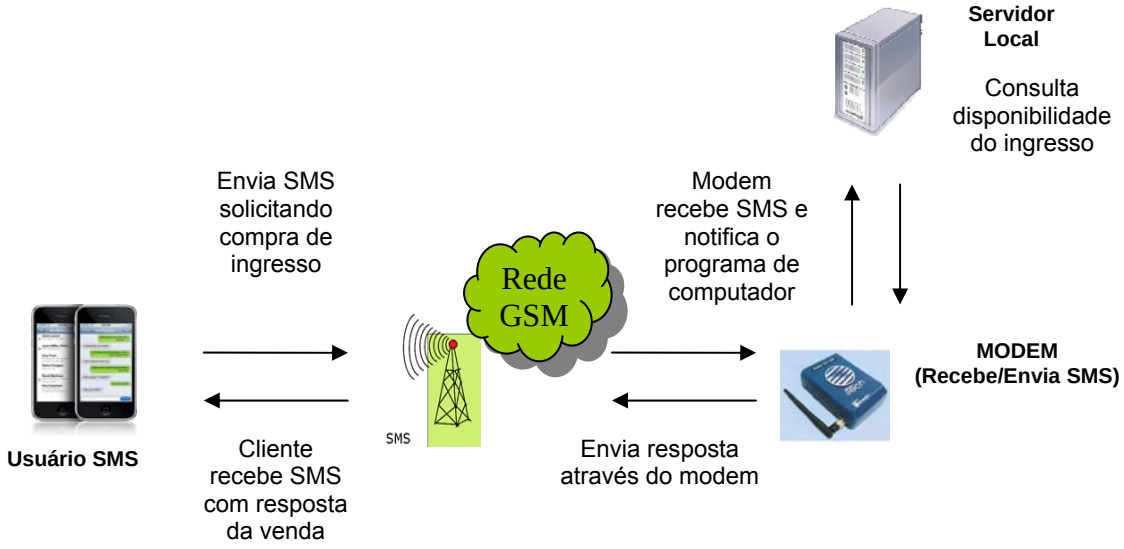


Figura 1.1 – Arquitetura simplificada do projeto.
Fonte (AUTORA DO PROJETO, 2009)

1.1 Justificativa

A motivação para a realização deste projeto surgiu com a grande acessibilidade que os usuários têm ao celular. Com o intuito de aproveitar essa facilidade, surgiu a idéia de se desenvolver uma aplicação que faz a interligação e permite a troca de dados entre um modem e um sistema de venda de ingressos de cinema.

Esse projeto visa permitir que os usuários utilizem melhor o seu tempo, evitando desperdiçá-lo em filas de espera.

A ferramenta demonstra a viabilidade da compra de ingressos de cinema usando o serviço SMS. Vale ressaltar, no entanto, que o protótipo desenvolvido no projeto não é uma solução completa de *mobile payment* tendo em vista que o pagamento dessa compra é simulado.

1.2 Objetivo Geral

O principal objetivo deste projeto é oferecer uma base tecnológica que demonstre a viabilidade da implementação futura de uma solução de *Mobile Payment.* O projeto prevê uma solução integrada entre o aparelho celular do usuário, utilizando a tecnologia do Sistema Global para Comunicações Móveis (*Global System for Mobile Communications* - GSM) e um modem de dados para receber e enviar mensagens, interligado com o servidor local.

O resultado esperado é receber uma resposta no celular via SMS correspondente à compra efetuada.

1.3 Objetivos Específicos

De modo a alcançar os objetivos gerais foram escolhidos os seguintes objetivos específicos:

- desenvolver um sistema para receber o SMS, tratar, efetivar a venda, devolver SMS e registrar em *log* essas operações;

- demonstrar que por meio do serviço de SMS da telefonia móvel o usuário envia um SMS para o número fixo do modem e recebe uma resposta via SMS referente a compra efetuada;

- desenvolver uma aplicação WEB que permitirá cadastrar os filmes assim como visualizar os registros das operações efetuadas.

1.4 Estrutura da Monografia

Este trabalho está organizado da seguinte forma: o capítulo 2 apresenta o problema a ser tratado, o capítulo 3 apresenta o referencial teórico abrangendo as tecnologias utilizadas no projeto; o capítulo 4 discorre detalhadamente como o projeto foi modelado, incluindo a integração entre os módulos; o capítulo 5 mostra os testes efetuados após a implementação do sistema, os resultados obtidos e a finalização do programa desenvolvido; por fim, no capítulo 6 são apresentadas as conclusões, dificuldades encontradas e as sugestões de trabalhos futuros.

2 APRESENTAÇÃO DO PROBLEMA

Com a solução proposta, pretende-se permitir que os usuários do sistema utilizem melhor o seu tempo, sem precisar ficar fisicamente localizados em filas para compra de ingresso, o que beneficiaria clientes que desejam agilidade e praticidade.

Atualmente, a primeira opção para a compra de ingressos de cinema, e a mais clássica, é aquela em que o cliente se dirige aos guichês de atendimento para efetuar a compra.

A segunda opção pode ser acessando uma página WEB pelo computador ou pelo celular através da tecnologia *General Packet Radio Service* (GPRS) para reservar um ingresso como oferecido pelo site do Cinemark, por exemplo. A tecnologia GPRS, por sua vez, é mais restrita porque, além de requerer suporte na rede e no próprio aparelho celular, trata-se de um serviço que demanda maior custo quando comparado com o SMS, que é oferecido em praticamente todos os modelos de celulares, mesmo os mais simples. Portanto, pode-se afirmar que, além de mais prática, a solução aqui proposta é mais abrangente, pois qualquer pessoa que possua um aparelho celular poderia fazer uso dela.

Já a terceira opção permite aos usuários comprar o ingresso de cinema pelos terminais totem de autoatendimento conveniados do Visa Electron, o qual se mostra ineficiente em diversas ocasiões, especialmente em dias de maior movimento quando a quantidade física de terminais se mostra ineficiente e devido a dificuldade em se manter funcionando esses aparelhos, pois é notado com uma certa freqüência que se encontra indisponíveis.

Nessa conjuntura, a solução aqui proposta beneficiará os clientes que preferem pagar a tarifa cobrada para o envio de SMS em troca do conforto de não enfrentar filas.

Esse sistema trará benefícios tanto para os clientes quanto para as casas de cinema. Para os clientes, permitirá otimizar o tempo disponível. Por outro lado, as casas de cinema poderão oferecer um diferencial competitivo em relação a outras que não disponham desse tipo de serviço. Tal diferencial, além de atrair a clientela, traz a perspectiva da redução do custo operacional de funcionamento dos cinemas, tendo em vista a possibilidade de diminuição do número de funcionários que trabalham nos postos de atendimento ao público.

É importante ressaltar que para completar o processo da compra utilizando SMS são relevantes as seguintes variáveis condicionantes: o local do evento, o dia e a hora da sessão, o pagamento do ingresso e o acesso ao cinema.

Essas variáveis são suficientes para determinar para qual filme o usuário deseja comprar os ingressos. Para efeitos deste trabalho, a determinação do responsável pela solicitação será feita exclusivamente pela utilização do número de telefone do celular emissor da mensagem. Por conseguinte, a tarifação poderia ser efetuada mediante convênio com a operadora de telefonia celular.

2.1 Mobile Payment

Conforme especificado em (WNEWS, 2008), o *Mobile Payment* é todo tipo de operação que envolva um dispositivo sem fio para iniciar, ativar ou confirmar um pagamento. A idéia é conseguir realizar o pagamento com o celular, que tende a ser usado como uma carteira eletrônica em substituição aos cartões de débito e crédito. Pode-se afirmar, então, que *Mobile Payment* é a validação de um pagamento via aparelho celular.

O objetivo do presente projeto é viabilizar uma forma de pagamento remoto. Esse pagamento é independente da localidade do usuário no momento da compra, porque as transações são efetuadas a partir do envio de mensagens de texto.

A solução proposta no projeto utiliza o serviço de mensagens curtas e é compatível com a maioria dos modelos de aparelhos celulares. Para o cliente, o custo adicional da compra é o preço que cada operadora de telefonia móvel cobra por torpedo enviado, que é em média R$ 0,31.

A transação pelo celular envolve o aparelho GSM da autora do projeto, a rede oferecida pelo serviço de telefonia da operadora TIM, um modem de dados e um servidor local.

3 REFERENCIAL TECNOLÓGICO

3.1 Padrão GSM

O padrão GSM, de acordo com (SVERZUT, 2008), passou a ser adotado no Brasil no ano de 2002. Para esse padrão, a largura de faixa de cada canal usada é de 200kHz e a faixa de operação é de 1,8 (GHz). A seguir, a figura 3.1 ilustra o padrão GSM:

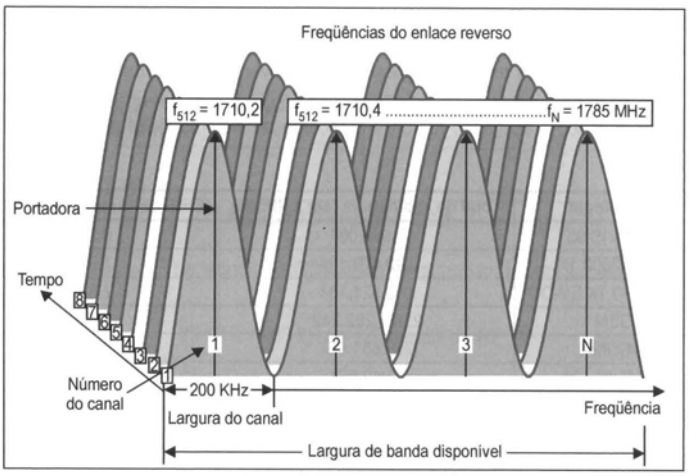


Figura 3.1 – Padrão GSM.
Fonte (SVERZUT, 2008)

Para o referido autor, três são as fases da evolução da tecnologia GSM:

Fase 1: desenvolveu os serviços básicos da tecnologia GSM:

- telefonia (Voz);
- chamadas de emergência (911 nos EUA, 112 na Europa);
- SMS : ponto a ponto e ponto multiponto;
- dados assíncronos (0.3 – 9.6 Kbps);
- dados síncronos (0.3 – 9.6 Kbps);
- transmissão de pacotes assíncronos.

Fase 2: segunda fase desenvolveu mais serviços, tais como:

- telesserviços;
- voz a meia taxa (*half rate*);
- melhorias no serviço SMS;
- serviços de dados;

- pacotes com transmissão síncrona e dedicada a taxas entre 2.4 e 14.4 Kbps;
- serviços adicionais: identificação do chamador (A), chamadas restritas por número, chamada em espera, teleconferência e grupo de usuários (*Voice Group Call Service*), entre outros.

Fase 2^{+}: foi introduzido o serviço de dados por pacotes a altas taxas de transmissão GPRS na rede GSM.

A figura 3.2 ilustra os serviços associados com as respectivas fases:

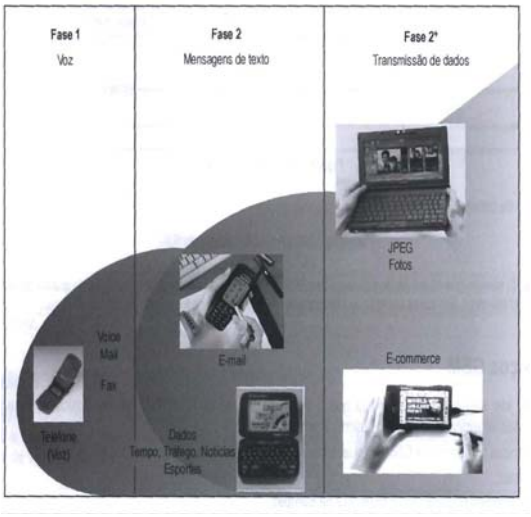


Figura 3.2 – Fases da tecnologia GSM.
Fonte (SVERZUT, 2008)

O sistema GSM é o mais utilizado no mundo. Segundo dados recentemente divulgados pela ANATEL, a participação da tecnologia GSM no Brasil já alcança 89,02%, enquanto a CDMA e TDMA juntas não chegam a 7%.

Uma das vantagens da utilização desse padrão tecnológico para este projeto é que ele trabalha com o SIM card (conhecido popularmente como “chip”), o qual armazena o perfil do assinante conectado na rede GSM e registra as mensagens recebidas semelhante a um hard disk.

A escolha deste padrão para a realização do protótipo apresentado neste trabalho teve como fundamento principal as observações de dados mercadológicos, uma vez que seria interessante, do ponto de vista da aplicabilidade deste projeto, trabalhar com a tecnologia dominante. Considerou-se também a maior

disponibilidade de produtos com essa tecnologia no mercado. Todavia, cabe ressaltar que, conceitualmente, a ideia aqui apresentada pode ser empregada por qualquer outra tecnologia digital celular que permita o envio e recebimento de mensagens SMS.

3.1.1 Arquitetura GSM

Segundo (VIEIRA, 2008) compõe a arquitetura física da rede GSM a estação móvel (*Mobile Station* - MS), o sistema de estação base (*Base Station System* – BSS) e a central de comutação celular (*Mobile Services Switching Centre* – MSC). Forma a MS o equipamento móvel (*Mobile Equipment* – ME) e o módulo de identidade do assinante (*Subscriber Identity Module* - SIM), que armazena a identidade internacional do assinante móvel (*International Mobile Subscriber Identity* - IMSI). A estação transceptora base (*Base Transceiver Station* - BTS), controlada pelo controlador de estação base (*Base Station Controller* - BSC) compõe o BSS. Por sua vez, várias BTS podem ser controladas por um BSC.

Conforme (VIEIRA, 2008) a comutação das chamadas e a interligação da rede móvel celular com outras redes são feitas pela MSC. As interfaces “Um” entre a MS e a BTS, “Abis” entre BTS e a BSC e “A” entre a BSC e a MSC podem ser observadas na figura 3.3.

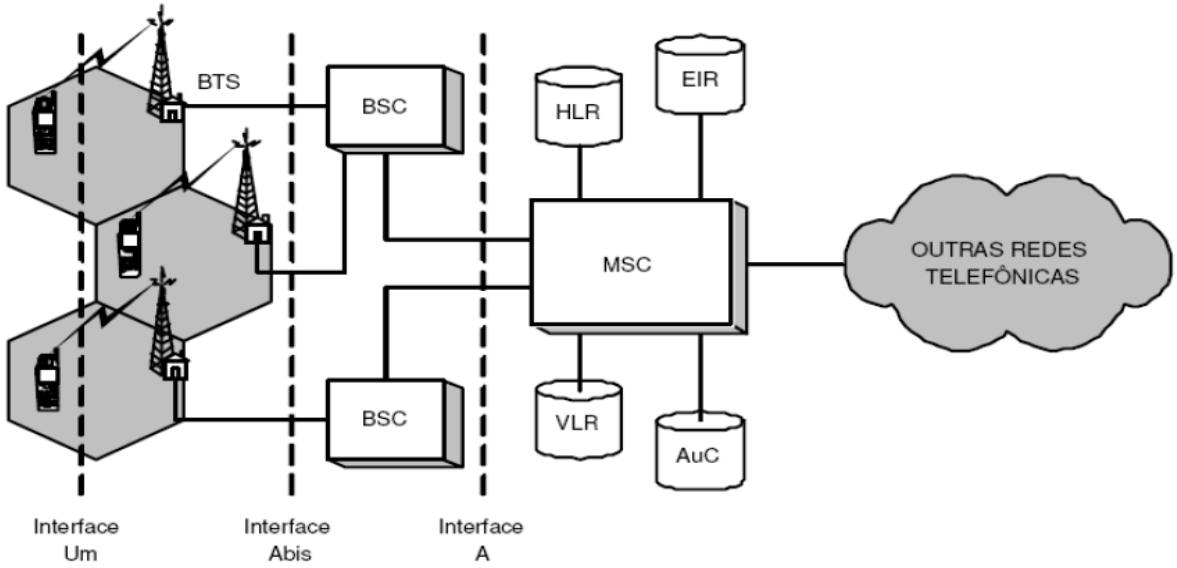


Figura 3.3 – Arquitetura do sistema GSM
Fonte (VIEIRA, 2008)

Para (SVERZUT, 2008), em uma rede GSM, o sistema de comutação de rede é responsável pelas funções de comutação, controle e gerenciamento da mobilidade e da base de dados dos assinantes.

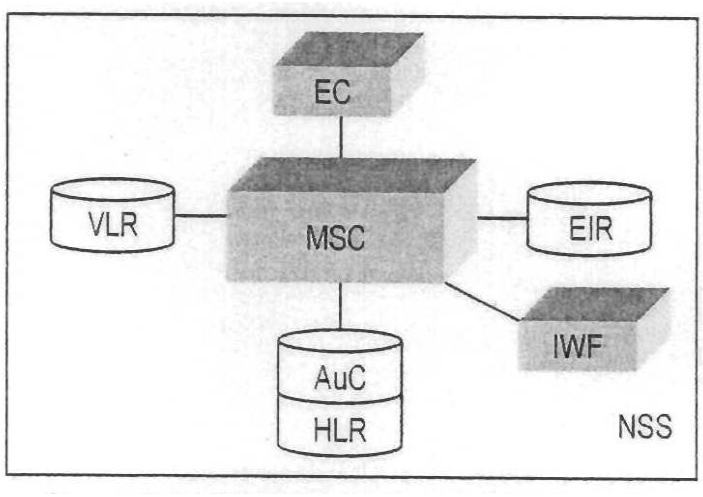


Figura 3.4 - Sistema de comutação de rede
Fonte (SVERZUT, 2008)

A figura 3.4 ilustra os componentes do sistema de comutação de rede (*Network Switching System* – NSS) e a seguir são descritos:

- central de comutação móvel, MSC;
- registro de localização local (*Home Location Register* - HLR);
- centro de autenticação (*Authentication Centre* - AuC);
- registro de localização de visitante (*Visitor Location Register* - VLR);
- registro de identidade de equipamento (*Equipment Identify Register* - EIR);
- função de interfuncionamento (*InterWorking Function* - IWF);
- supressor de eco (*Echo Canceler* - EC).

De mesma opinião (VIEIRA, 2008) considera as bases de dados descritas anteriormente essenciais para se estabelecer os procedimentos de gerência da rede. O HLR, por sua vez, armazena informações administrativas, ou perfil, de cada assinante registrado na rede GSM correspondente, bem como sua localização atual. O VLR é aquele responsável por armazenar informações sobre usuários de outra operadora que por acaso estejam conectados na rede. Dessa forma, essas informações são repassadas pelo HLR da rede de origem da MS visitante.

Ainda com base em (VIEIRA, 2008) as informações necessárias para a autenticação do usuário são armazenadas no AuC contida no SIM de cada MS. Já o EIR armazena o *International Mobile Station Equipment Identity* – IMEI, espécie de número de série único, presente em todos os MS e necessário durante o acesso à rede. Pode-se dizer, portanto, que uma MS só tem acesso à rede com um IMSI e um IMEI válidos.

3.2 Serviço SMS

O serviço SMS permite que os usuários, através de estações móveis ou por outros dispositivos conectados em rede, enviem mensagens de textos para uma *Short Message Entity* (SME).

O usuário para enviar um SMS insere o número do celular do destinatário, escreve o texto da mensagem em uma interface no aparelho celular e envia para o serviço de rede da telefonia celular. Para esse protótipo, por exemplo, é preciso enviar uma mensagem com o seguinte texto: “COMPRAR 01 2045 2”.

Essa mensagem para chegar ao destinatário é então transportada por uma ou mais redes móveis. No caso, o chip usado no modem G24 é da rede TIM, logo usuários dessa mesma rede poderão, eventualmente, conseguir receber uma resposta via SMS mais rápida.

Quando o destinatário está disponível a mensagem é entregue diretamente, caso contrário, a mensagem é mantida pelo serviço de rede da telefonia por um determinado período até que o destinatário se torne disponível.

Segundo (LE BODIC, 2003) esse serviço envolve uma pessoa, um destinatário, uma ou mais redes de telefonia celular para transportar a mensagem e outros intermediários, tais como provedores de serviços.

3.2.1 Arquitetura SMS na rede GSM

(LE BODIC, 2003) lembra a necessidade de serem adicionados à arquitetura da rede GSM, GPRS, UMTS (*Universal Mobile Telecommunications System*) dois elementos para o serviço de SMS. O primeiro deles é o centro de SMS (*Short Message Service Centre* – SMSC). Esse elemento desempenha um papel chave na arquitetura do SMS. A sua função é retransmitir as mensagens de texto entre os aparelhos de celular e o *store-and-forwarding* dessas mensagens quando o receptor está indisponível. O SMSC pode fazer parte da rede de telefonia celular ou ser uma entidade de rede independente.

Operadores da rede de telefonia celular geralmente possuem contratos comerciais recíprocos que permitem a troca de mensagens de texto entre as redes. Desse modo, uma mensagem enviada por um usuário conectado a uma rede A pode ser entregue ao destinatário associado a uma rede B de telefonia. Essa troca de mensagens entre assinantes de diferentes redes, por vezes, localizados em até diferentes países é uma das características do SMS.

O segundo elemento é o Email gateway que permite a interoperabilidade entre E-mail e SMS através de uma conexão com o SMSC e a Internet. Essa operação , no entanto, não se insere no escopo desse projeto.

Para (LE BODIC, 2003), além desses dois elementos é importante ressaltar a entidade de mensagens curtas (SME). Essa é uma entidade que pode receber ou enviar mensagens. Mencionada entidade pode ser localizada na rede fixa, em uma estação móvel, ou em outro centro de serviço. Quando uma SME é um servidor ligado a uma SMSC diretamente ou via gateway, denomina-se entidade móvel externa (*External Short Message Entity* – ESME).

A figura 3.5 mostra o dois elementos: SMSC e o Email gateway adicionados a arquitetura GSM:

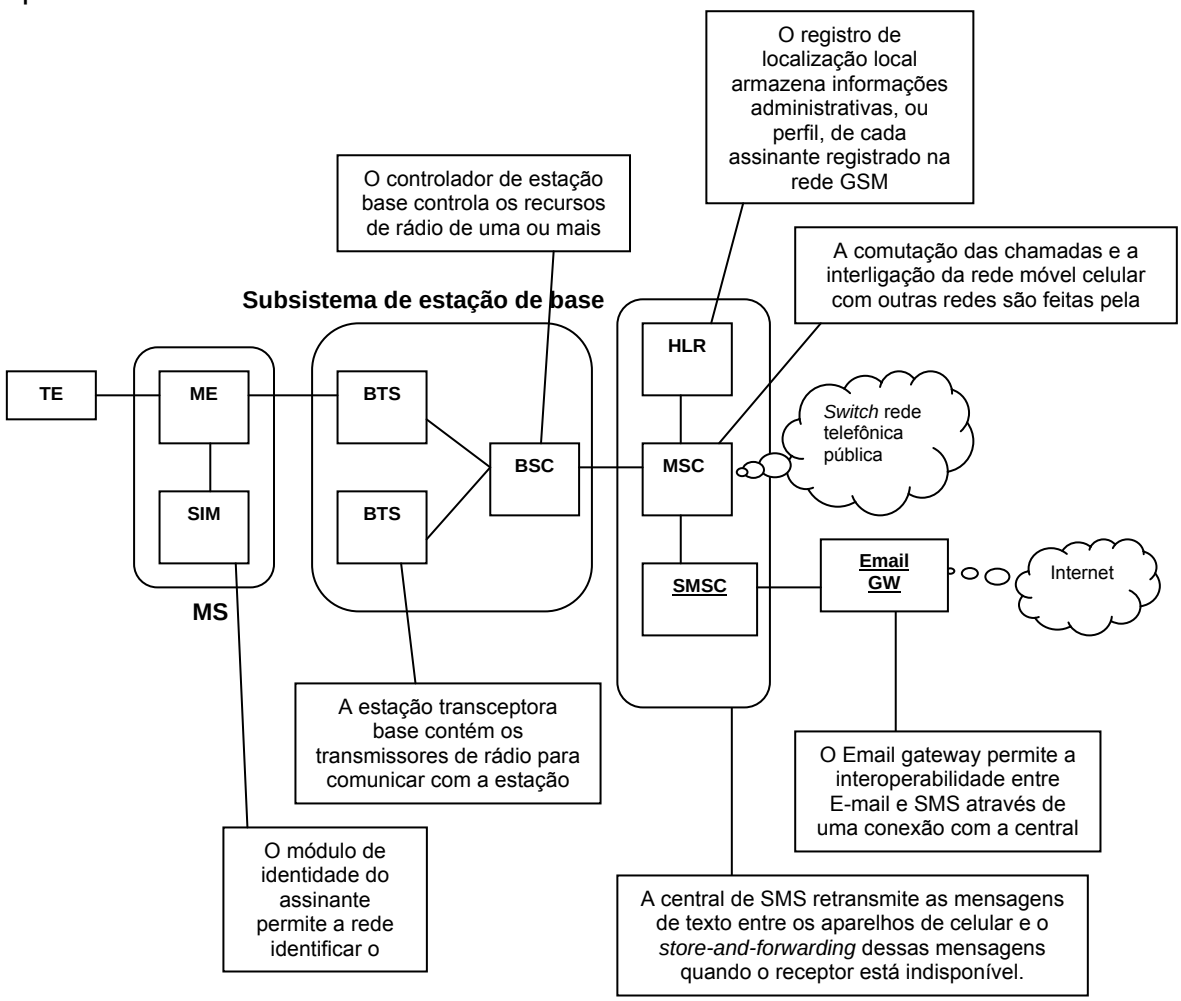


Figura 3.5 - Arquitetura SMS na rede GSM.
Fonte (LE BODIC, 2003).
Tradução (AUTORA DO PROJETO)

3.2.2 Características Básicas

Segundo (LE BODIC, 2003), duas são as características básicas do SMS: o envio e a recepção de mensagens de texto. Mensagem originada é aquela criada no aparelho celular e enviada para SMSC. Essas mensagens são endereçadas para outro SME. Mensagens terminadas são as mensagens entregue pela SMSC para as estações móveis.

Mensagens originadas ou terminadas podem ser entregues ou submetidas durante uma chamada de voz ou mesmo em progresso de uma conexão de dados. Dessa forma, são entregues ou enviadas através dos canais de sinalização da rede GSM e também por canais da rede GPRS.

3.2.3 Camadas e Protocolos do SMS

A pilha de protocolo do SMS, de acordo com (LE BODIC, 2003), é composta de quatro camadas:

- camada de aplicação
- camada de transferência
- camada inter-redes (relay)
- camada de enlace

A camada de aplicação (*Short-Message-Application-Layer* SM – AL) é implementada na SME por meio de um aplicativo capaz de enviar, receber e interpretar o conteúdo da mensagem de texto.

Na camada de transferência (*Short-Message-Transfer-Layer* SM – TL), a mensagem é considerada uma seqüência de octetos que contém informações da mensagem tais como, sua extensão, emissor ou receptor da mensagem e data da mensagem.

A camada inter-redes (*Short-Message-Relay-Layer* SM – RL) permite transportar uma mensagem entre vários elementos de rede. Um elemento da rede pode armazenar a mensagem, por determinado período, se o elemento seguinte estiver indisponível. O MSC nessa camada lida com duas funcionalidades adicionais à central de comutação. A primeira função chamada SMS gateway MSC (SMS – GMSC) consiste em receber uma mensagem do SMSC, interrogar o HLR para obter as informações de roteamento e também entregar a mensagem para a rede do destinatário. A segunda função SMS InterWorking MSC (SMS – IWMSC) resume-se em receber uma mensagem de uma rede de telefonia móvel e enviá-la para o SMSC.

A camada de enlace (*Short-Message-Link-Layer* SM – LL) trata da transmissão de mensagens no nível físico. Desse modo, a mensagem é protegida para lidar com o baixo nível de erros do canal.

A pilha de protocolo do SMS pode ser vista na figura 3.6. Para transportar uma mensagem, a aplicação mapeia o conteúdo da mensagem e associa um protocolo (*Transfer Protocol Data Unit* – TPDU) na camada de transferência. Um TPDU é um conjunto de parâmetros. O parâmetro protocolo de transferência possui um prefixo TP e indica o tipo da mensagem, por exemplo, se foi ou não solicitado

relatório de entrega pelo emissor e entre outras. Exemplo: *TP-Message-Type-Indicator* (TP-MTI) e *TP-Status-Report-Indication* (TP-SRI), respectivamente.

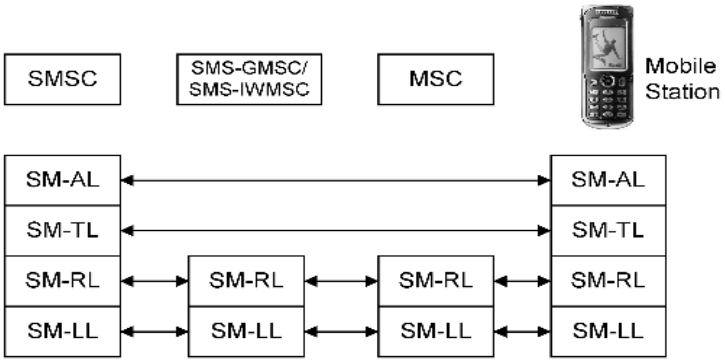


Figura 3.6 – Pilha de protocolo SMS
Fonte (LE BODIC, 2003)

A troca de mensagens, na camada de transferência, é feita em duas ou três etapas. O processo dividido em três etapas é ilustrado na figura 3.7. Etapa 1: a mensagem é submetida para o SMSC uma vez criada pelo emissor. O SMSC verifica com a rede de telefonia móvel se o emissor está apto ou tem permissão para enviar mensagens. Etapa 2 : O SMSC entrega a mensagem para o receptor SME. Quando não ocorre a entrega da mensagem porque o receptor está indisponível, o SMSC armazena temporariamente a mensagem até o receptor ficar disponível ou até o prazo de validade da mensagem expirar. Etapa 3: Após a entrega da mensagem ou a exclusão desta da rede, um relatório de entrega pode ser enviado para o emissor, nos casos em que ele solicitar no envio da mensagem.

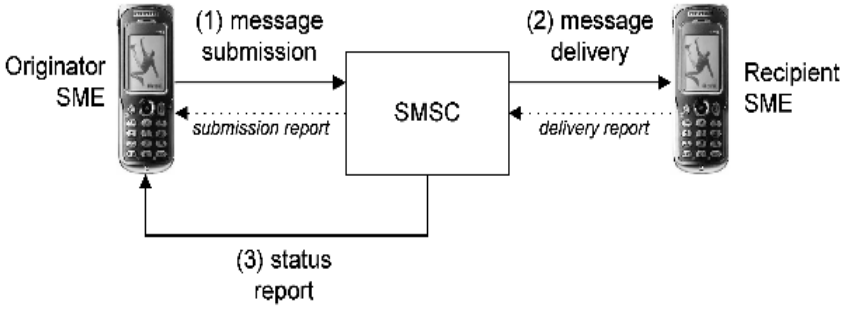


Figura 3.7 – Troca de SMS entre duas SME.
Fonte (LE BODIC, 2003)

De acordo com (LE BODIC, 2003), a tecnologia GSM/GPRS permitiu às operadoras de telefonia móvel fazerem acordos comercias que permitem o intercâmbio de mensagens entre diferentes redes.

São seis os tipos de operação que podem ocorrer entre uma SME e um SMSC na camada de transferência. Cada transação corresponde a um tipo de TPDU. A figura 3.8 ilustra os tipos de transação:

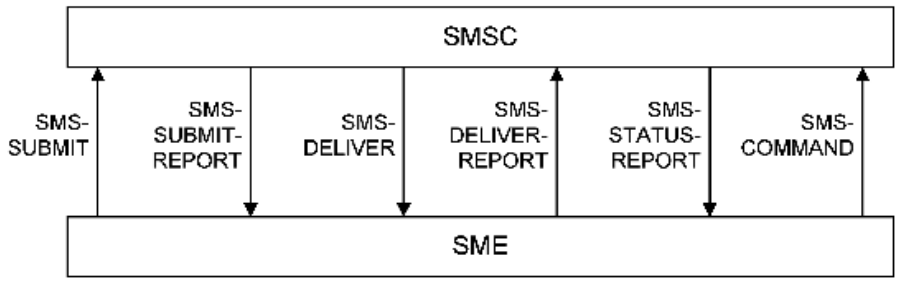


Figura 3.8 – Tipos de transações entre uma SME e o SMSC.
Fonte (LE BODIC, 2003)

- *SMS-SUBMIT*: quando uma SME envia uma mensagem para SMSC.
- *SMS-SUBMIT-REPORT*: recebida pelo SMSC e reconhecida retorna um relatório para SME.
- *SMS-DELIVER*: quando um SMSC entrega a mensagem para SME.
- *SMS-DELIVER-REPORT*: retorna para SMSC um relatório uma vez recebida a mensagem pela SME.
- *SMS-STATUS-REPORT*: corresponde a transação que retorna do SMSC um relatório para a SME.
- *SMS-COMMAND*: é um pedido da SME, via de regra uma SME externa, que solicita para o SMSC executar um comando específico.

Pode-se considerar que as aplicações de SMS são diretamente baseadas na camada de transferência.

3.2.4 Controle do SMS via G24

Uma MS conecta-se a um terminal externo (*Terminal Equipment* – TE) via interface assíncrona, por meio de um cabo serial, infravermelho ou qualquer outro tipo de conexão conforme ilustrado na figura 3.9. A comunicação entre MS e o TE pode ser de três formas:

- Modo em bloco (*Block mode*): protocolo de comunicação binário que inclui proteção contra erros. É aconselhável usar esse modo quando a ligação entre a MS e o TE não é confiável.
- Modo texto (*Text mode*): protocolo baseado em caracteres adequados para aplicações de alto nível. Esse protocolo é baseado em comandos AT. As iniciais AT indicam atenção. Esse prefixo é usado para iniciar uma linha de comando AT a ser enviado do TE para o MS no modo texto.
- Modo protocolo de data (*Protocol Data Unit* – PDU): é protocolo baseado em caracteres hexadecimais que transfere os comandos entre a MS e o TE. Esse modo é indicado para *drives* de baixo nível, que não interpretam o conteúdo dos comandos.

Figura 3.9 – MS conectado com TE
Fonte (LE BODIC, 2003)

Vale ressaltar que, independentemente do modo de conexão, o TE está no controle das operações SMS. De forma análoga, pode-se dizer que o TE funciona como “mestre” e a MS como “escravo”.

3.3 Linguagens de Programação

Linguagem de Programação é o conjunto de regras lógicas utilizadas para programar as ações em um computador. O programador representa suas idéias no código fonte e posteriormente esse mesmo código fonte é traduzido para código de máquina para ser executado pelo processador.

Para o projeto, o *Visual Studio* foi a interface de desenvolvimento escolhida em virtude de ser um ambiente integrado.

3.3.1 C Sharp (C#)

Conforme descrito em (Wille, 2001) a linguagem de programação C# deriva do C e C++. São elementos do C#: modernidade, orientação a objetos, compatibilidade e flexibilidade. Para o referido autor este é um sistema de tipos unificados que permite ao programador visualizar cada tipo como um objeto, seja um tipo primitivo ou uma classe completa.

Essa linguagem faz parte da plataforma .NET desenvolvida pela Microsoft.

3.3.2 ASP.NET

Segundo a MSDN, ASP.NET é a tecnologia para desenvolver aplicativos *WEB* utilizando a plataforma NET da Microsoft. O ambiente .NET Framework é um ambiente de desenvolvimento que permite várias linguagens de programação e bibliotecas trabalharem em conjunto para criar as aplicações que são mais fáceis de construir, gerir, implementar e integrar com outros sistemas de rede ou como aplicações autônomas. Esse ambiente consiste basicamente de dois elementos abaixo descritos:

- CLR (*The Common Language Runtime*) - um ambiente de execução que independe de linguagem, através de uma interação com coletâneas de bibliotecas.
- FCL (*The Framework Class Libraries*) – coletâneas de biblioteca orientada a objeto que permite a interação.

O .NET Framework é uma plataforma única para desenvolvimento e execução de sistemas e aplicações. Todo e qualquer código escrito e gerado pelo .NET poderá ser executado em qualquer dispositivo que possua o framework instalado porque um componente do IIS (Internet Information Service) o interpreta e possibilita a exibição de paginas WEB dinâmicas.

O ASP.NET herda todas as características do Framework e pode ser escrita em diversas linguagens. A linguagem de programação escolhida neste projeto foi C#.

3.3.3 XMLNuke

Conforme descrição contida no site XMLNUKE, este é um sistema de gerenciamento e também um framework de desenvolvimento para WEB que permite armazenar, processar e exibir conteúdos para sites. É preciso utilizar o conjunto de classes de abstração do XML existentes no Framework, sendo que o próprio XMLNuke se encarrega de montar o documento XML pronto para ser transformado. Isso ocorre porque o sistema permite manipular as informações através de documentos XML. Esses documentos são transformados através de instruções XSL e exibidos para o usuário em qualquer navegador WEB. Vale afirmar que as funcionalidades existentes no XMLNuke são executadas em sua totalidade utilizando apenas o mecanismo principal do XMLNuke e um serviço http.

3.3.4 *Hayes AT Commands*

Comandos AT constituem o método padrão de configuração e comunicação com modems. É um conjunto composto por uma série de comandos curtos de texto que combinam para realizar as operações.

O usuário pode criar dois tipos de conexões para estabelecer uma sessão de *AT Commands* com o G24:

- Ligação RS232
- Conexão USB

O usuário pode usar tanto uma conexão RS232 ou USB, mas não ambas simultaneamente. Para o projeto foi escolhido ligar o servidor com o G24 usando uma conexão USB.

Conforme o manual do modem G24 da Motorola (MOTOROLA, 2006), os comandos AT são utilizados para solicitar serviços do modem G24, tais como:

- Fazer chamadas: discar, atender e desligar
- Enviar / receber SMS
- Permitir Auto Resposta
- Permitir consultar a qualidade do sinal GSM.

Os comandos AT são usados para configurar o modem para receber e enviar SMS através de redes celulares de tecnologia GSM.

3.4 Notações

Uma notação é um conjunto de símbolos que se usa para representar uma idéia. O projeto faz uso de duas notações a seguir descritas. A primeira é uma gramática e a segunda é um padrão para reconhecer as combinações de caracteres.

3.4.1 Backus- Naur- Form

Segundo conceito fornecido pelo sítio NETPEDIA, BNF é uma metalinguagem usada para a definição da sintaxe de linguagens formais, tanto para o desenvolvedor da linguagem quanto para o usuário. Uma linguagem é difundida por um conjunto de instruções, em cada uma das quais um elemento da linguagem conhecido como uma metavariável, escrita entre os sinais "<" e ">", é definida em termos de símbolos reais (chamados terminais). Exemplo da sintaxe na forma Backus-Naur:

- "::=" significa "é definido como"
- "|" separa alternativas.

Já o WIKIPEDIA, define Backus- Naur Form como sendo uma notação amplamente usada nas gramáticas de computador para definir a sintaxe da linguagem de programação, conjunto de instruções e protocolos de comunicação.

BNF, através de uma gramática, foi a melhor forma encontrada para descrever os métodos e retornos do protocolo de comunicação que serão utilizados nesse projeto.

3.4.2 Expressões Regulares

Expressões regulares, de acordo com (JARGAS, 2008), são compostas de símbolos, caracteres com funções especiais, que, agrupados entre si e com caracteres literais, formam uma seqüência, uma expressão. Essa expressão é interpretada como uma regra, que indicará sucesso se uma entrada de dados qualquer combinar com essa regra, isto é, precisa obedecer rigorosamente as condições definidas.

No projeto, as variáveis são definidas no protocolo de funcionamento a ser discutido no capítulo 04 dessa monografia.

A forma escolhida para validar se o texto contém os caracteres esperados definidos no projeto é através das expressões regulares. Dessa forma, as referidas expressões serão muito utilizadas tanto para obter uma determinada seqüência de caracteres do texto, como para validar se os comandos AT foram executados com sucesso e, ainda, para se obter as respostas dos comandos AT.

3.5 Banco de Dados

Banco de dados é um conjunto de dados estruturados em tabelas composta por linhas e colunas. Esse banco de dados é acessado por meio de um software conhecido como Sistema Gerenciador de Banco de Dados (SGBD). O objetivo é armazenar os registros da aplicação desenvolvida.

O *SQL Server Management Studio* 2008 foi o SGBD escolhido para implementar o banco de dados do projeto devido a sua acessibilidade, portabilidade do arquivo e conhecimento prévio da linguagem de programação SQL.

O projeto prevê para registro no banco de dados dois casos:

- Caso 1: uma vez atendida a disponibilidade para a sessão de cinema especificada e confirmada a compra, o sistema registra na tabela dbo.*smslog* e dbo.registro_compra: a data e a hora que recebeu a mensagem, o número do celular que enviou o SMS, a mensagem que foi recebida pelo modem, a mensagem enviada pelo modem para o usuário, o nome do filme, o horário da sessão escolhida e a quantidade de ingressos que foram comprados pelo usuário.

- Caso 2: no caso de indisponibilidade ou comando desconhecido, registra na tabela dbo.*smslog*: o número que enviou o SMS, a data e a hora da solicitação, o texto da mensagem recebida pelo modem e o texto da resposta enviada pelo modem para o usuário.

Para qualquer caso, exposto acima, será feito um registro no banco de dados da data e hora do recebimento pelo modem do SMS, o número do celular que enviou o SMS para o modem e o status da mensagem enviada pelo modem para o usuário. O relatório poderá ser extraído através de qualquer consulta SQL ou através da interface WEB gerada pela solução Adm Cinema no Visual Studio. A página principal da interface web oferece para o usuário fazer a consulta por duas opções “Consultar Registro de Compra”, que mostra apenas as compras efetuadas com sucesso, ou pela opção “Ver Registro do SMS”, que mostra todas as operações ocorridas no sistema.

3.6 Componentes Físicos

3.6.1 Celular

O celular da autora do projeto usado para solicitar a compra de um ingresso via SMS é um iPhone 2G do modelo de 8GB, desenvolvido pela Apple Inc. O aparelho foi adquirido em outubro de 2007.

Na tabela 3.1 são descritas as especificações técnicas do aparelho:

Tabela 3.1 – Especificações técnicas do aparelho.
Fonte Apple Inc.

Fabricante	Apple
Tipo	Smartphone
Tela	3.5 pol, 320x480 px em 160 ppi
Câmera	2.0 megapixel
Sistema Operacional	iPhone OS (2.2.1)
Entrada	Tela sensível ao toque, Quatro botões físicos principais
Memória	8GB
Rede	GSM / UMTS
Conectividade	GSM 2.75G Quad band (850/900/1800/1900 MHz) Wi-Fi (802.11b/g)
Bateria	Bateria Li-Ion 3,7 volts
Dimensões	115×61×11.6 (mm)

3.6.2 Modem de dados

O modem de dados com tecnologia GSM/GPRS – EDGE (*Enhanced Data rates for GSM Evolution*) foi o recurso eleito para fazer a integração entre o celular e o servidor.

Conforme disposto pela Informat Technology, fornecedora do referido modem, o G24 TCP/IP Wireless Data Modem é um equipamento voltado para a transmissão de dados, através das redes celulares de tecnologia GSM/GPRS-EDGE. Este modem possui interface de comunicação padrão, que lhe permite ser conectado a diversos outros equipamentos dos quais se queira transmitir informações.

É equipado com microcontrolador Motorola HCS08 e possui memória disponível para rodar aplicações proprietárias. Este recurso, portanto, proporciona flexibilidade nas aplicações.

A descrição das características físicas e técnicas estão na Tabela 3.2:

Tabela 3.2 – Descrição das características físicas e técnicas do modem G24.
Fonte: Informat Technology.

Dimensões	100 x 70 x 30 mm
Peso	200 g
Conector RF de saída	SMA (fêmea) 50ohm GSM
Temperatura Operacional	-30 a +60 ºC
Temperatura Armazenamento	-40 a +85 ºC
Sistema de Operação	Quad band 850/900/1800/1900 MHz
Tensão	5 a 35 V
Potência de Saída	0.6W - 850 MHz 2W - 900 MHz 1W - 1800/1900 MHz
Leitor do SIM Card	Interno - chip SIM CR 1.8/3V SIM
SMS	MT/MO Modos Texto e PDU
Áudio Analógico	Full duplex I/O em conector opcional

3.6.3 SIM Cards

De acordo com (SVERZUT, 2008), o módulo de identificação do usuário (SIM) é um cartão inteligente (*smart card*) conectado internamente ao equipamento móvel. Esse cartão contém informações sobre a estação móvel, tais como:

- Identidade internacional do assinante móvel - IMSI;
- Identidade temporária do assinante móvel (Temporary Mobile Subscriber Identity - TMSI);
- Identidade da área de localização (*Location Area Identity* - LAI);
- Chave de autenticação do assinante (*Subscriber Authentication Key* - Ki);
- Número internacional ISDN (*Integrated Service Digital Network*) da estação móvel (*Mobile Station Integrated Services Digital Network* - MSISDN).

A figura 3.10 ilustra o módulo de identificação do usuário (SIM):

Figura 3.10 – Módulo de identificação do usuário (SIM).
Fonte (AUTORA DO PROJETO, 2009)

Para (SVERZUT, 2008), o processamento e a tarifação das chamadas são realizados a partir das informações contidas no cartão SIM e não no terminal móvel. O assinante para habilitar uma estação móvel precisa apenas adquirir um cartão SIM da operadora desejada para ser utilizado em qualquer terminal GSM compatível com o sistema da operadora.

O projeto faz uso de dois SIM Cards. O primeiro está no celular do usuário que solicita a compra enviando um SMS para o número fixo do modem. O segundo SIM Card é encontrado no modem GSM G24 da Motorola.

3.6.4 Computador

Um computador é composto basicamente por uma unidade central de processamento, unidades de memória, canais de comunicação e dispositivos de entrada e de saída.

Nesse projeto foi utilizado um laptop para atuar como servidor. A seguir são descritas as especificações desse servidor:

- Um laptop VAIO Intel(R) Core (TM) Duo CPU T8300 @ 2.40 GHz, com HD de 250GB, com 4.00 GB RAM, *Mobile Intel* (R) 965 *Express Chipset Family*.

3.7 Interface USB

Conforme informação do *site* WESTER (acesso março de 2009), a interface USB é um padrão que facilita a conexão de dispositivos com o computador. Essa é uma tecnologia simples e fácil que permite a conexão com diversos tipos de aparelhos eletrônicos ao computador. A seguir são descritas as vantagens dessa tecnologia:

- Padrão de conexão: usa padrões definidos de conexão;
- *Plug and Play*: o sistema operacional reconhecerá a conexão do dispositivo imediatamente;
- Alimentação elétrica: a própria conexão USB é capaz de fornecer eletricidade, por isso, a maioria dos dispositivos que usam USB não precisa ser ligada a uma fonte de energia;
- *Hot – swappabl*e: permite que dispositivos USB possam ser conectados e desconectados a qualquer momento. Em um computador, por exemplo, não é necessário reiniciá-lo ou desligá-lo para conectar ou desconectar o dispositivo;

Compõem os cabos USB quatro fios internos conforme (WESTER, 2009). São eles: VBus (VCC), D+, D- e GND. O primeiro é o responsável pela alimentação elétrica. O segundo e o terceiro são utilizados na transmissão de dados. O quarto, por sua vez, é para controle elétrico, servindo como "fio-terra".

A conexão entre o modem e o servidor local é feita através de um cabo USB que possui uma ponta para a porta Mini-USB e a outra para o tamanho padrão.

4 CRIANDO O PROTÓTIPO

4.1 Conceitos Iniciais

O protótipo do sistema de compra de ingresso usando SMS é formado por duas soluções: SMS Cinema e ADM Cinema. A primeira solução testa as funções básicas do sistema e contém o serviço que processa as mensagens recebidas; a segunda é a interface WEB para cadastramento e consulta ao banco de dados e é baseado no framework XMLNuke.

A primeira fase para desenvolver esse sistema foi a análise de requisitos para capturar as intenções e necessidades dos usuários do sistema a ser desenvolvido. O foco dessa fase é tentar mostrar o que os usuários do futuro sistema deverão esperar do aplicativo, conhecendo toda sua funcionalidade sem se importarem como esta será implementada.

A segunda fase do projeto direciona-se às primeiras abstrações (classes e objetos) e mecanismos que estarão presentes no domínio do problema. As classes são modeladas e ligadas através de relacionamentos com outras classes.

Uma terceira fase, conhecida como fase projeto, resulta no detalhamento das especificações para a fase de programação do sistema.

Por fim, a programação é a fase onde os modelos criados são convertidos em código.

Esse sistema passou por testes de unidade e integração. Os testes de unidade são para classes individuais do “SMSClasses” que permite testar os comandos do “ProjetoTeste” da solução SMSCinema. Os testes de integração são aplicados já usando as classes e componentes integrados para confirmar se as classes estão cooperando uma com as outras no projeto “ServicoModem”.

O usuário para fazer uso desse sistema de compra através do serviço da telefonia móvel padrão envia do seu aparelho celular um SMS para o número fixo do modem. A aplicação faz uma análise léxica e sintática da mensagem recebida pelo modem. Uma vez validado esses caracteres da mensagem, o sistema verifica se tem bilhetes disponíveis para aquela sessão especificada e, após confirmada a disponibilidade, envia uma resposta para o usuário.

O pagamento do bilhete será efetuado mediante débito na conta de celular. O cinema firmaria contratos com as operadoras de telefonia móvel, especificando os critérios de repasse do valor dos ingressos, que poderia ser mediante a emissão da fatura pelos cinemas, acompanhada de relatório discriminado, contendo o número de ingressos vendidos por esse sistema, o número dos respectivos telefones celulares, a data da compra e o valor do ingresso.

Vale lembrar que o aplicativo desenvolvido não é uma solução completa para a venda de ingressos usando SMS pois não tem meios para efetuar a cobrança de fato dos ingressos vendidos e também não possui uma parceria firmada com um cinema. Mesmo assim, a aplicação desenvolvida gera o registro de todas as informações processadas e está apta, para futuramente firmar as parcerias. O projeto visa demonstrar que os objetivos propostos foram atingidos por meio das tecnologias escolhidas.

Este capítulo trata dos seguintes itens:

- por que foram escolhidas determinadas tecnologias;
- requisitos funcionais do sistema de venda de ingresso e fluxo de funcionamento da aplicação;
- a Solução SMS Cinema;
- desenvolvimento da solução ADM Cinema utilizando C# e XMLNuke;
- configuração do modem de dados;
- desenvolvimento da interface para cadastrar os filmes e consultar o *log;*
- requisitos não funcionais do sistema de venda de ingresso;
- desenvolvimento do Banco de Dados *SQL Server Express Edition* 2008;
- integração entre os módulos;
- problemas e soluções encontrados, durante o desenvolvimento.

4.2 Razão do uso da tecnologia escolhida

A tecnologia .NET para programação do projeto foi escolhida em razão da existência de uma extensa coleção de ferramentas e de documentação disponível para auxiliar o desenvolvimento de aplicações tanto em livros como na Internet. Além disso possui uma excelente integração com o banco de dados SQL Server. O Framework XMLNuke foi escolhido para a ferramenta de administração pois ele possui várias classes que facilitam o cadastramento, alteração e exclusão de dados.

4.3 Requisitos Funcionais do sistema de compra de ingressos de cinema usando SMS

O usuário para comprar um bilhete de cinema envia para o número do modem de dados uma mensagem com o seguinte texto:

'COMPRAR', <espaco>, <sessao>, <espaco>, <qtd>

Esse campo <sessao> determina o nome do filme e a hora da sessão. É definido da seguinte forma: <sessao> ::= <filme>,<espaco>, <hora>,<minuto>. A seguir um exemplo da mensagem originada pelo usuário:

“COMPRAR 01 2045 2”

A aplicação primeiramente valida o comando. Após a validação do comando o próximo passo é validar a compra. A validação da compra ocorre quando confirmada a disponibilidade de assentos na sessão escolhida. Nesse caso retornará um SMS com a mensagem ”COMPRA EFETUADA”. Caso não confirmada a disponibilidade de assento, a aplicação retornará para o usuário uma mensagem “INDISPONIVEL”. Quando o comando for inválido, a aplicação enviará uma mensagem “COMANDO DESCONHECIDO”. Por último, caso ocorra algum erro não previsto no sistema será enviada uma mensagem “ERRO” para o usuário.

A aplicação vai gerar um registro das operações para que futuramente possa ser feita uma integração com o cinema e posterior débito em conta do cliente.

Em termos de *Mobile Payment* é importante frisar que esta solução não é completa e busca demonstrar a viabilidade de implementação desse sistema de compra por SMS. Assume-se que o pagamento do ingresso fica implícito porque o contrato com a telefonia celular é simulado.

O sistema retorna para o usuário uma mensagem com o status da operação efetuada por ele. A mensagem resposta pode ser de “COMPRA EFETUADA”, “INDISPONIVEL”, “COMANDO DESCONHECIDO” ou “ERRO”. Desse modo para considerar como comprovante que permite acesso ao cinema o SMS enviado pelo sistema seria interessante gerar um código único que o usuário do celular mostraria ao atendente do cinema. O atendente, por sua vez, com um terminal na mão validaria o código. No entanto, vale ressaltar que essa é uma das possíveis soluções encontradas, mas que não contempla o escopo desse projeto.

Abaixo a figura ilustra o protocolo SMS:

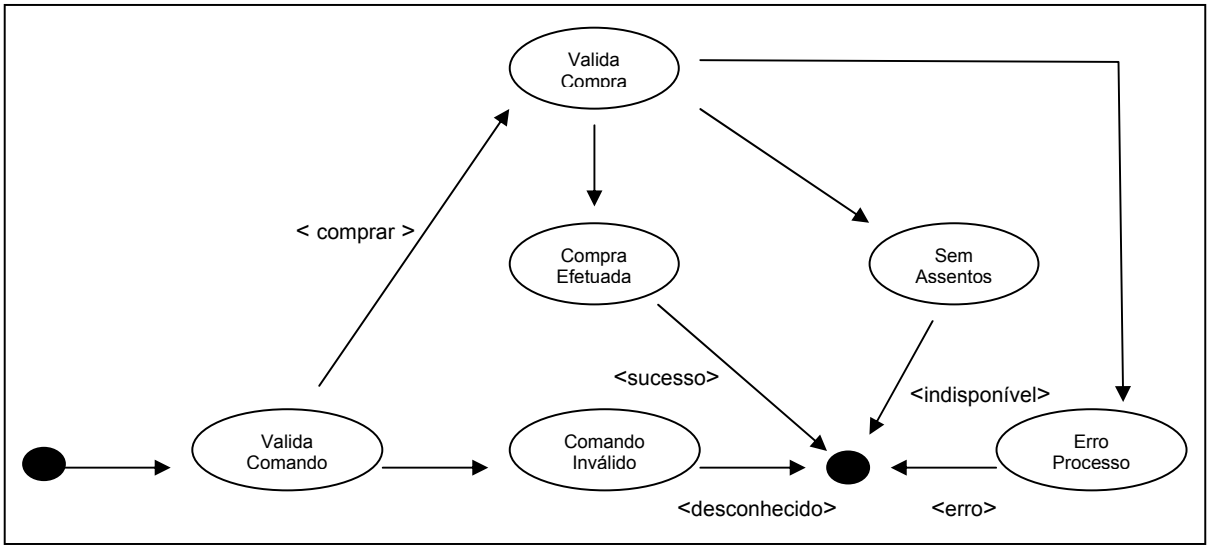


Figura 4.1 – Protocolo SMS
Fonte (AUTORA DO PROJETO, 2009)

4.3.1 A Solução SMS Cinema

A Solução SMS Cinema desenvolvida é responsável pela comunicação entre o modem G24 e o servidor local. Essa solução possui três projetos: “ProjetoTeste”, “ServicoModem” e “SMSClasses”.

O “ProjetoTeste” é uma aplicação criada para validar os comandos. Serve como um teste de unidade para os comandos AT e o modem. Compõe essa aplicação os seguintes comandos:

- (Direto): testa a conexão com a porta serial;
- Teste: comando testa a conexão com o modem;

- Listar Msg: esse comando lista todas as mensagens que estão armazenadas fisicamente no modem. Estas mensagens devem ser lidas, tratadas, armazenadas no banco de dados e então apagadas da memória do modem. Caso não tenha mensagens na lista o sistema gera um erro que deve ser tratado. O sistema espera receber uma quantidade de mensagens e caso não as receba, ele traz *null* no *matching* da expressão regular e por isso retorna um erro.
- Apaga Msg: esse comando possui dois casos. No primeiro caso depois de processar a mensagem o comando apaga a mensagem da memória do modem G24. No segundo caso, para apagar uma mensagem específica basta digitar o id da mensagem na caixa de texto logo abaixo da opção “Apaga Msg” e em seguida pressionar o comando;
- Enviar SMS: esse é um comando composto de duas partes. A primeira parte solicita o envio da mensagem e a segunda define o texto para então completar o comando. Permite enviar um SMS do modem para um celular qualquer;
- Saldo: faz uso do comando Envia SMS para enviar uma mensagem para o número 222 com o texto “SAL” e ele devolve um texto com a quantidade de créditos;
- Limpar: permite limpar a tela do formulário do “ProjetoTeste”.

A tela do formulário do “ProjetoTeste” na figura 4.2 é mostrado o resultado de um teste realizado com os comandos na seguinte ordem:

1. (Direto)
2. Teste
3. Enviar para modem a mensagem: PROJETO TESTE
4. Listar Msg
5. Enviar SMS para 81510355 mensagem Projeto Teste com sucesso
6. Enviar para o modem: Teste APAGAR essa mensagem
7. Listar Msg
8. Apagar Msg
9. Listar Msg
10. Saldo
11. Listar Msg

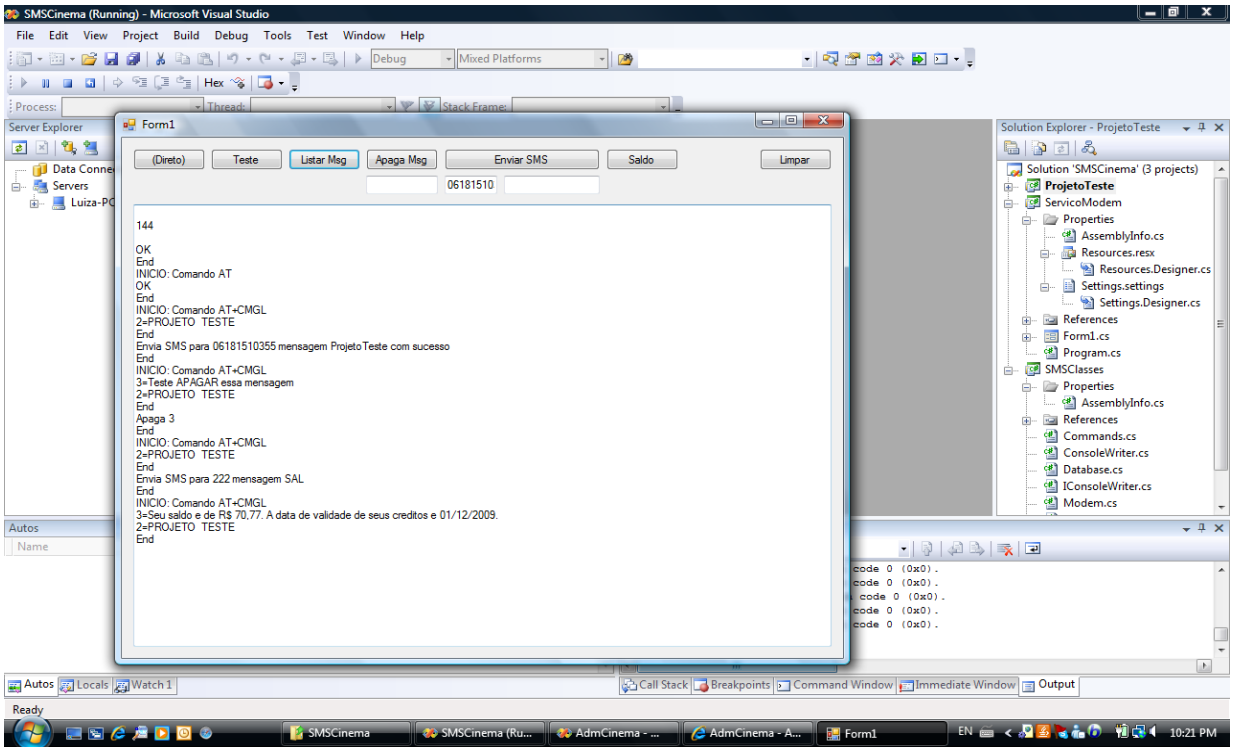


Figura 4.2– Mostra a tela do formulário “ProjetoTeste” da Solução SMS Cinema testando os comandos.
Fonte (AUTORA DO PROJETO, 2009)

O segundo projeto “ServicoModem” consulta o modem em intervalos regulares para verificar se tem mensagens novas ou não. Este é o projeto que efetivamente integra todas as classes e efetua a compra dos ingressos. A interface deste projeto é bem simples, possuindo dois comandos: iniciar e parar. O primeiro comando inicializa o modem e inicia a contagem regressiva de 15 segundos para verificação de novas mensagens. O segundo comando permite parar essa verificação. O componente que possibilita fazer a verificação em intervalos regulares é o Timer que dispara o evento Timer1_Tick. Esse evento é executado a cada 15 segundos e faz todo o processo necessário para executar as funções.

A figura 4.3 do formulário projeto “ServicoModem” dessa solução mostra as mensagens que foram encontradas na lista de memória do modem e a resposta enviada para o usuário. Note que neste exemplo foi possível observar os seguintes comandos respectivamente: COMANDO DESCONHECIDO, INDISPONIVEL e COMPRA EFETUADA. Na tela também aparece uma mensagem com o saldo disponível no chip pré-pago contido no modem.

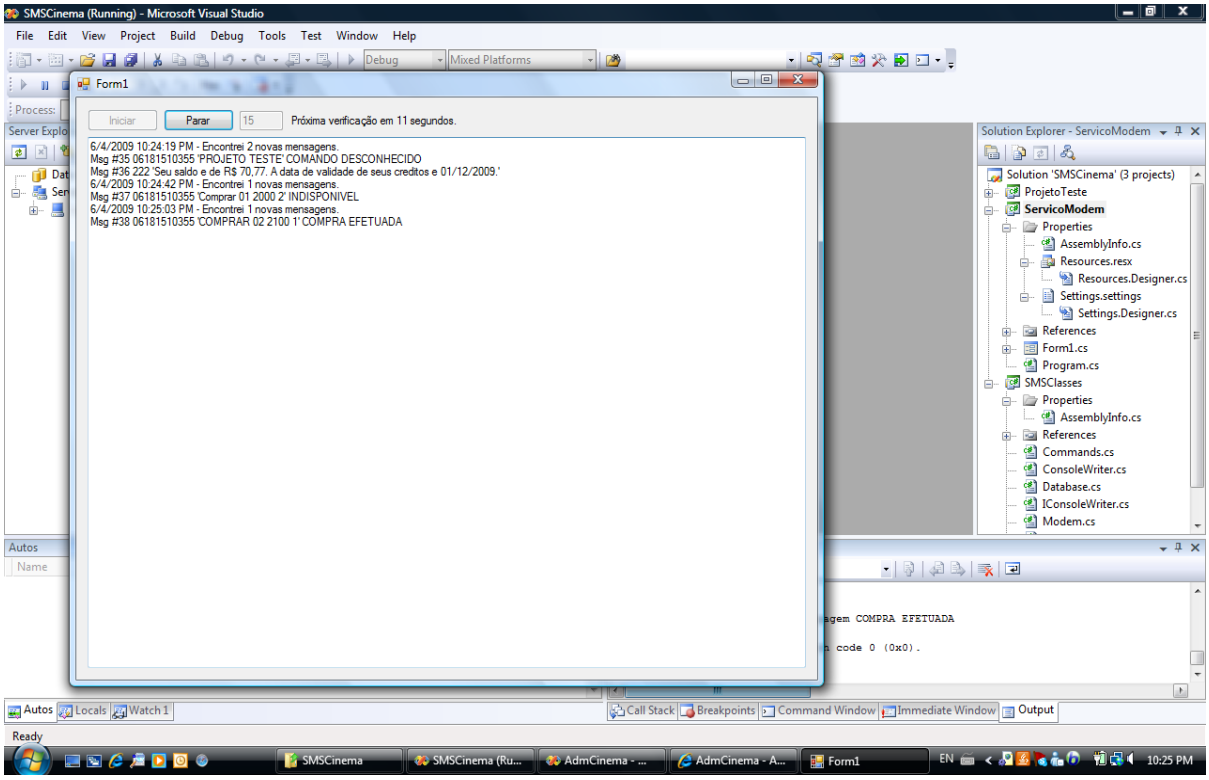


Figura 4.3 – Mostra o formulário do “ServicoModem” da solução SMS Cinema.
Fonte (AUTORA DO PROJETO, 2009)

O terceiro projeto “SMSClasses” contém as classes de acesso a dados e protocolo de funcionamento. Para iniciar esse projeto o primeiro contato com o modem foi pelo Hyperterminal para testar os comandos AT. Em seguida, procurou-se desenvolver a classe Modem com o programa para escrever os comandos em seqüências. O objetivo da classe Modem é garantir a comunicação com o modem. A segunda classe criada foi a classe *Commands* que agrupa um conjunto de comandos AT em uma estrutura mais simples. A classe protocolo, por sua vez, foi desenvolvida por último porque ela é específica da solução do sistema de venda de ingresso. Compõe essa estrutura:

- *Commands.cs*: é uma classe responsável por escrever os comandos AT na classe Modem.cs;
- *ConsoleWriter.cs*: essa classe implementa a interface IConsoleWriter e contém os comandos necessários para escrever na tela do formulário a saída do modem. Útil para debugar;
- *Database.cs*: responsável pelo armazenamento dos dados da aplicação. Estabelece conexão com banco de dados e executa o *insert* nas *querys* SQL;
- *IConsoleWriter.cs*: é uma interface de programação de classes. Define

que toda a classe que implementar essa estrutura é obrigada a implementar todos os métodos existentes nessa interface, ou seja, define uma espécie de "contrato". Sendo assim, os métodos só precisam conhecê-lá. Como o *ConsoleWriter* e a *NullWriter* implementam a Interface (esse contrato) tanto faz chamar uma classe ou outra, entretanto, cada uma executa uma função diferente da outra;

- *Modem.cs*: é uma classe responsável por conectar, enviar e receber dados do modem G24;
- *NullWriter.cs*: é uma classe que implementa a interface IConsoleWriter, entretanto descarta as mensagens recebidas pelo modem;
- *Protocol.cs*: define parâmetros esperados e recebidos através de uma máquina de estados e validação dos comandos e das respostas declarados na gramática. O principio básico:

<mensagem_recebida> → <mensagem_resposta>

É importante notar que essa estrutura foi criada utilizando C# no Visual Studio e dentro dos métodos das classes estão os comandos AT que são enviados para o modem. A figura 4.4 mostra a estrutura com as classes de acesso a dados e protocolo SMS:

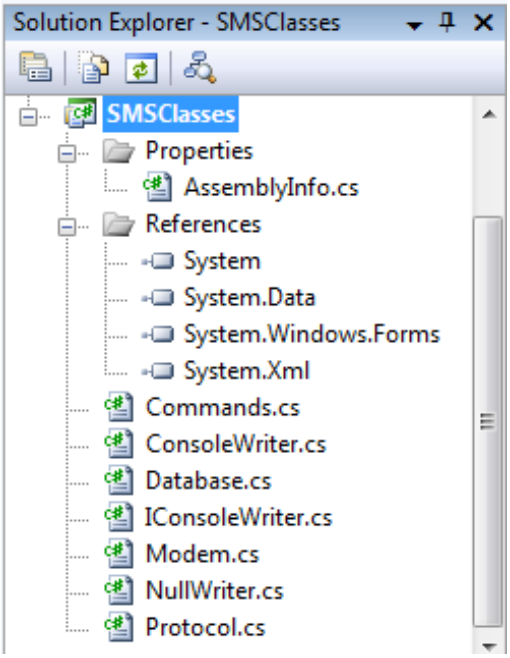


Figura 4.4 – Mostra a solução SMS Cinema desenvolvida no Visual Studio.
Fonte (AUTORA DO PROJETO, 2009)

4.3.2 Desenvolvimento da solução Adm Cinema utilizando C# e XMLNuke

A solução Adm Cinema é composta de dois projetos: Adm Cinema e C:\...\xmlnuke\. Para criar a solução contendo o projeto Adm Cinema configurado para rodar com XMLNuke no Visual Studio com C# foi usado o script "create-xmlnuke-project.vbs".

O projeto Adm Cinema ficou dividido em cinco classes:

- RegistroCompra.cs: classe de registro de compra é o módulo que apresenta em uma tabela com todas as compras efetuadas com sucesso. Essa classe instancia uma conexão com banco de dados e possui uma string responsável em concatenar os dados, fazer as junções entre as tabelas e mostrar os resultados com dados de compra na seguinte ordem: data, número celular, *messagetext*, *response*, filme, hora, minuto e quantidade;
- *NewModule.cs*: essa classe novo item permite cadastrar um novo nome de filme. Em seguida envia o dado para o banco de dados e mostra o resultado em uma tabela com todos os filmes cadastrados no sistema até o momento através de uma consulta análoga a um *select* da tabela filme;
- *LogSMS.cs*: essa classe corresponde ao módulo que permite ver o registro do SMS, uma vez que instância uma conexão com banco de dados e possui uma string que executa um *select* na tabela *smslog*. A tela dispõe o Log de Mensagens do sistema com: o idsms, o número do celular, data, *messagetext*, *response;*
- *Home.cs*: essa classe corresponde a tela inicial da interface WEB que oferece para o usuário três opções: “Cadastrar Filmes e Assentos Disponíveis”, “Consultar Registro de Compra” e “Ver Registro do SMS”;
- CadastroFilme.cs: a classe cadastro de filme permite ao usuário cadastrar novos filmes e editar filmes cadastrados. A classe em primeiro lugar estabelece uma conexão com banco de dados, em seguida mostra na tela os seguintes campos para cadastrar ou alterar: Cód Assento não editável pelo usuário, uma caixa de texto para digitar a data atual, uma lista com o nome dos filmes cadastrados, uma lista para escolher a hora entre às dez horas até às vinte e três horas, outra lista para escolher o minuto “zero zero”, quinze, trinta ou quarenta e cinco, um campo para digitar o número totais de assentos e por fim o número de assentos disponíveis para o sistema.

Abaixo pode ser visualizado na figura 4.5 as classes e as referências que o projeto Adm Cinema faz uso para rodar.

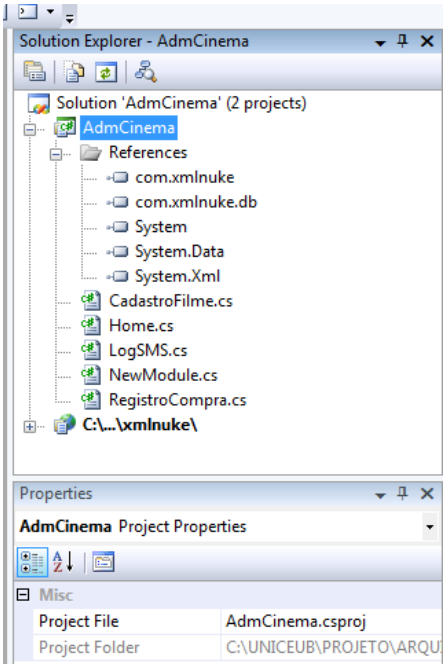


Figura 4.5 – Tela do Visual Studio com a solução Adm Cinema.
Fonte (AUTORA DO PROJETO, 2009)

Observe determinadas características desse projeto:

- toda classe que possui interação com o usuário é chamada de módulo pelo XMLNuke e herda a classe BaseModule. Essa classe possui diversas implementações básicas como segurança, cache, etc.
- apesar do XMLNuke possuir um classe para acessar dados diretamente, foi utilizado extensamente no projeto a classe ProcessPageStateDB. Essa classe possui uma pequena máquina de estados que permite Obter, Cadastrar, Alterar e Excluir dados de uma tabela simples. Essa classe é responsável por ler os dados do banco de dados, dispor em uma tabela, permitir a edição e efetuar a atualização no banco de dados.
- XMLNuke não possui uma interface de programação visual como o ASP.NET, sendo tudo programado em classes não visuais. Mas pelos recursos oferecidos de produtividade no projeto foi muito útil a sua utilização.

As classes do XMLNuke que permitem o acesso ao banco de dados são DBDataSet e o DBIterator . O DBDataSet é usado para executar uma consulta e o BDIterator é utilizado para percorrer a consulta. O interessante desse framework é que o acesso ao banco de dados é entendido como sendo um acesso a um arquivo XML.

O arquivo que define a conexão com o banco de dados é o _db.anydata.xml. A figura 4.6 abaixo mostra onde está localizado este arquivo:

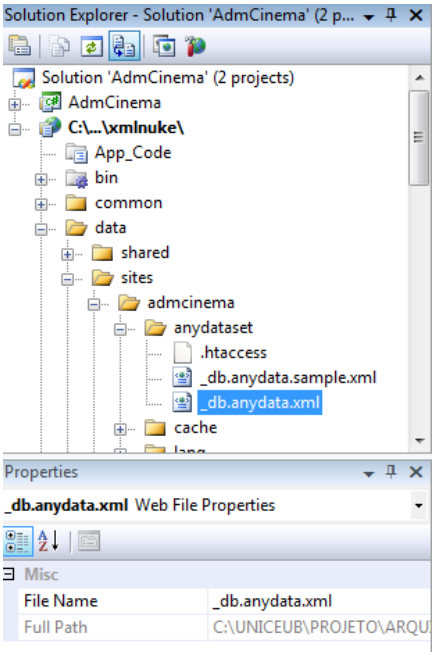


Figura 4.6 – Mostra o endereço do arquivo _db.anydata.xml no projeto C\...\xmlnuke\
Fonte (AUTORA DO PROJETO, 2009)

O XMLNuke é um Framework de desenvolvimento de aplicativos WEB. Ele possui um conjunto de classes para produzir conteúdo XML. Esse conteúdo após produzido pode ser transformado em HTML ou qualquer outro formato através de um *template* como um RSS, Atom, Excel, etc.

Além disso, o XMLNuke possui um conjunto de classes que tornam o trabalho de programar para WEB mais fácil, pois já possui classes para cadastramento de dados, classes para abstração do banco de dados, classes para abstração de diversas rotinas, dentre outras.

Uma característica importante nesse framework é que o programador não precisa se preocupar com o layout, mas sim com o que deve ser produzido. A apresentação da informação se torna secundário no desenvolvimento da aplicação para a WEB.

4.3.3 Configuração do modem de dados - Hayes AT commands

Do modem de dados para ler a mensagem no celular existem duas abordagens:

1) Ficar conectado eternamente ao celular e monitorar as mensagens de recebimento

2) Conectar de tempos em tempos e verificar se existem mensagens na lista de tal forma que obedece o status, lida ou não lida, de cada mensagem

Existem prós e contras nessas abordagens. No primeiro caso assim que a mensagem chegasse seria possível tratá-la e devolver a mensagem para o usuário. Seria quase imediato. Entretanto, haveria um esforço de programação maior pois deveria tratar se o dispositivo está ocupado recebendo outra mensagem antes de enviar, além de mandar periodicamente uma mensagem de keep-alive para checar a conexão entre o modem e o computador ainda está ativa.

Já na segunda abordagem, as mensagens são enviadas para o modem e armazenadas em sua memória interna. Em intervalos de tempo pequenos e regulares consulta o modem à procura de novas mensagens e em seguida processa todas as mensagens encontradas. Caso algum problema de dispositivo ocorra, um erro será gerado e então tratado no próximo intervalo, sem prejuízo para a mensagem. Além de ter uma implementação bem mais simples. Sendo assim, optou-se pela segunda abordagem de conexão ao modem.

No modo escolhido a mensagem chega e é armazenada no celular. O evento Timer1_Tick solicita ao celular a lista de mensagens que chegaram a cada 15 segundos. De posse dessa lista, faz o registro no banco de dados, processa a informação contida na mensagem, devolve a mensagem para o usuário e por fim o comando apagar remove a mensagem da memória do modem.

4.3.4 Interface para cadastrar os filmes e consultar o *log*

A aplicação WEB é uma aplicação em C# que permite incluir os dados no sistema para a compra de ingressos usando SMS. A vantagem dessa aplicação é que através de qualquer navegador é possível executá-la, pois ela não exige que o cliente possua o framework instalado ou tenha que fazer configurações específicas para o seu acesso.

Em termos de arquitetura da aplicação, portanto, foi escolhido o framework XMLNuke que facilita o processo de desenvolvimento porque já possui várias rotinas prontas para fazer um CRUD básico (Cadastro, Inclusão, Alteração e Exclusão).

Na aplicação WEB é feito o cadastro do nome do filme, a hora e minuto da sessão, quantidade de assentos totais e a quantidade de ingressos disponíveis para o sistema de compra de ingresso usando SMS. Além disso, na própria aplicação é possível consultar o *log* de forma análoga a uma consulta no banco de dados em duas opções: “Consultar Registro de Compra” e “Ver Registro do SMS”.

A figura 4.7 mostra a o menu oferecido pela interface WEB:

Figura 4.7 – Página principal da interface WEB.
Fonte (AUTORA DO PROJETO, 2009)

A seguir na figura 4.8 uma tabela com os filmes cadastrados no sistema acessados pela opção Cadastrar Filmes e Assentos Disponíveis:

Figura 4.8 – Mostra a tabela da opção Cadastrar Filmes e Assentos Disponíveis.
Fonte (AUTORA DO PROJETO, 2009)

A opção consultar registro de compra permite visualizar os registros das compras efetuadas com sucesso conforme mostra um exemplo na figura 4.9:

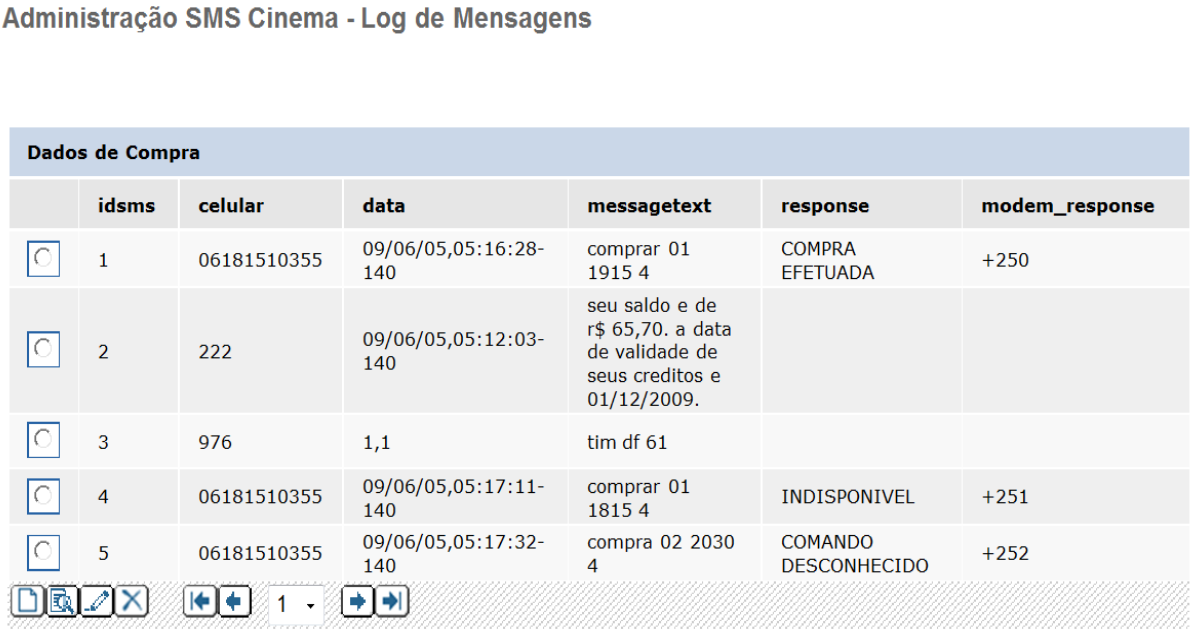
Administração SMS Cinema - Registro de Compra

Dados de Compra

	data	celular	messagetext	response	filme	hora	minuto	qtd
	09/05/19,05:54:34-140	06181510355	comprar 01 2115 2	COMPRA EFETUADA	Filme 1	21	15	2
	09/05/19,05:56:55-140	06181510355	comprar 01 2115 4	COMPRA EFETUADA	Filme 1	21	15	4

Figura 4.9 – Tabela Dados de Compra da opção Consultar Registro de Compra da interface WEB. Fonte (AUTORA DO PROJETO, 2009)

A opção Ver Registro do SMS mostra o registro de todas as transações ocorridas entre um celular GSM e a aplicação SMS Cinema:



Administração SMS Cinema - Log de Mensagens

Dados de Compra

	idsms	celular	data	messagetext	response	modem_response
	1	06181510355	09/06/05,05:16:28-140	comprar 01 1915 4	COMPRA EFETUADA	+250
	2	222	09/06/05,05:12:03-140	seu saldo e de r$ 65,70. a data de validade de seus creditos e 01/12/2009.		
	3	976	1,1	tim df 61		
	4	06181510355	09/06/05,05:17:11-140	comprar 01 1815 4	INDISPONIVEL	+251
	5	06181510355	09/06/05,05:17:32-140	compra 02 2030 4	COMANDO DESCONHECIDO	+252

1

Figura 4.10 – Tabela Log de Mensagens da opção “Ver Registro do SMS” pela interface WEB. Fonte (AUTORA DO PROJETO, 2009)

Como pode ser observada na figura 4.10 a primeira linha é um registro de uma compra efetuada, a segunda linha mostra uma verificação do saldo disponível no chip contido no modem G24, na terceira linha um número de telefone inválido para retorno por isso atribui-se nulo para mensagem, na quarta linha retornou indisponível para o usuário porque o horário da sessão não estava cadastrado para o filme especificado e por último a quinta registra um comando desconhecido porque a palavra compra para o sistema é invalida.

4.4 Requisitos Não Funcionais do sistema de compra de ingressos de cinema usando SMS

A aplicação simula a venda de ingressos de cinema usando SMS. Esse sistema, por sua vez, possui as seguintes limitações a seguir descritas:

A) a compra por esse serviço fica restrita ao dia da solicitação, isto é, esse serviço não permite comprar ingresso para uma sessão do dia seguinte;

B) a tolerância do sistema para a compra via SMS é de 15 minutos antes do início da sessão. Na hipótese de o usuário enviar uma mensagem com menos de 15 minutos de antecedência o sistema retornará uma mensagem “INDISPONIVEL”;

C) a aplicação desenvolvida no projeto considerará um número pré-definido de ingressos disponíveis para a compra por intermédio de SMS. Deixou-se de trabalhar com percentual sobre o número de assentos totais da sala de cinema porque o resultado poderia dar um número com casas decimais, e o quantitativo de ingressos é um número inteiro. Desse modo, foi desenvolvida uma interface para cadastrar os filmes e os assentos disponíveis. O universo de assentos disponíveis será definido proporcionalmente ao número total de lugares existentes na sala de cinema. O cadastro é feito da seguinte forma:. 1- Cadastrar Filmes 2 – Cadastrar Sessões e assentos por sessão. Os passos enumerados a seguir descrevem como cadastrar um novo filme utilizando a interface WEB. Selecionar a opção cadastrar filmes e assentos disponíveis, novo item. Insere o nome do filme, envia os dados. Seleciona o filme na tabela, em seguida opção editar assentos disponíveis, depois novo item. Preencher as caixas de diálogo com a data atual, seleciona o nome do filme que foi cadastrado no passo anterior, determina hora e minuto da sessão,

quantidade de assentos totais da sala de cinema e assentos disponíveis para o sistema de compra de ingresso usando SMS;

D) a compra por esse serviço é referente ao valor total do ingresso (inteira). No caso de meia - entrada, poderia ser exigido um pré-cadastro por parte do cliente e uma forma de atestar que o usuário do celular é possuidor do direito à meia-entrada;

E) essa solução fica restrita aos usuários que tem plano pós-pago junto às operadoras de telefonia móvel, pois o pagamento será feito mediante débito na fatura emitida pela operadora de telefonia móvel;

F) as mensagens do tipo “Te Ligou” não são mostradas na tela do formulário do “ProjetoTeste” mas podem ser visualizadas em uma situação de erro conforme mostra a figura 4.11:

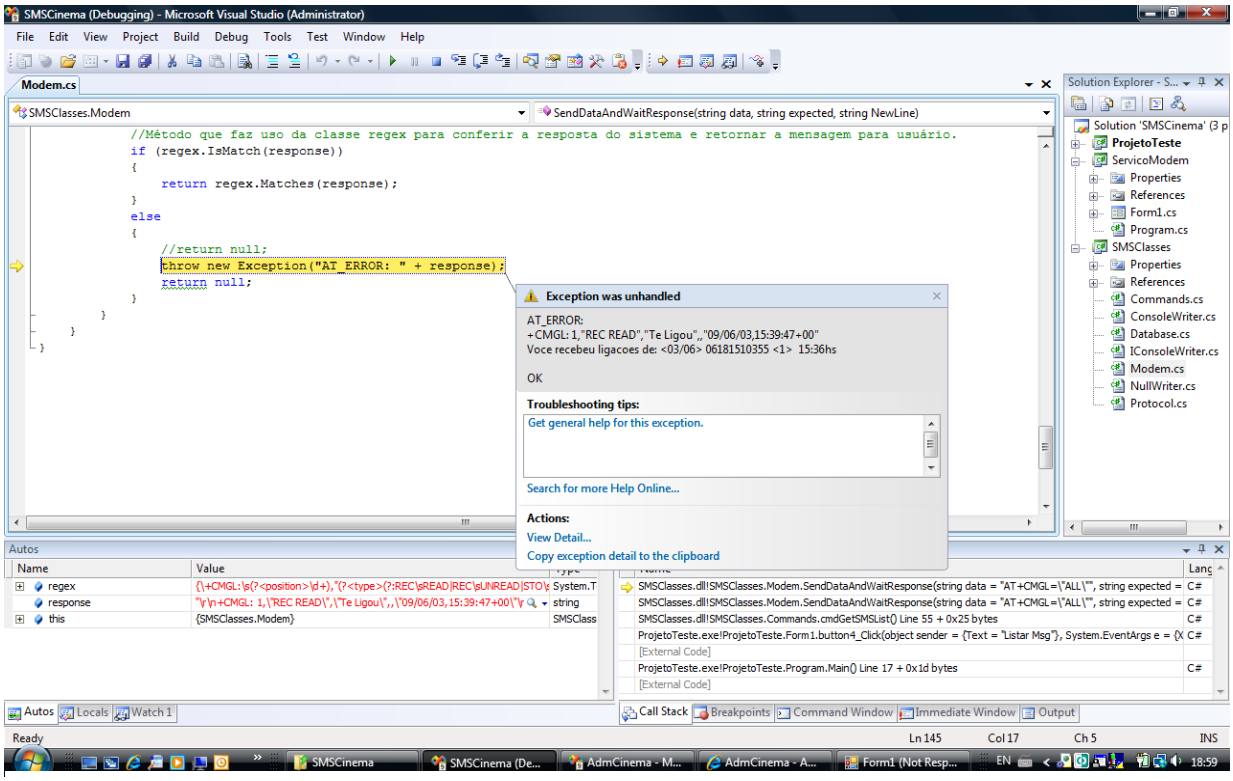


Figura 4.11 – Tela mostra um erro no Visual Studio com a mensagem “Te Ligou” enviada pela própria operadora informando data, hora, o número que ligou e quantas vezes tentou falar com este número (no caso do modem!).

Essas mensagens são descartadas pela aplicação porque não são válidas para o protocolo definido pelo projeto. No entanto, vale ressaltar que, ao ligar o modem, a própria operadora envia para o usuário da operadora TIM, que ligou para o modem, um SMS com a seguinte mensagem: “Ligue Agora! Já estou disponível para receber chamadas (data e a hora atual). Serviço gratuito TIM”. Quando o modem está ativo e ele recebe uma ligação ela será transferida para caixa postal, pois ele só foi configurado para receber e enviar SMS.

4.5 Banco de Dados

O modelo lógico se preocupa em representar a estrutura dos dados e suas relações em um banco de dados. Ele consiste em criar uma abstração da realidade do BD, enfatizando os objetos que a ele pertencem e as relações entre estes objetos e os dados que serão armazenados.

O modelo de banco de dados para gerenciar o recebimento de SMS e efetuar a venda de assentos a uma determinada sessão de cinema é composto de quatro tabelas: filme, assento, registro_compra e *smslog*.

A figura 4.12 mostra a tela do programa DB Designer com o modelo ER:

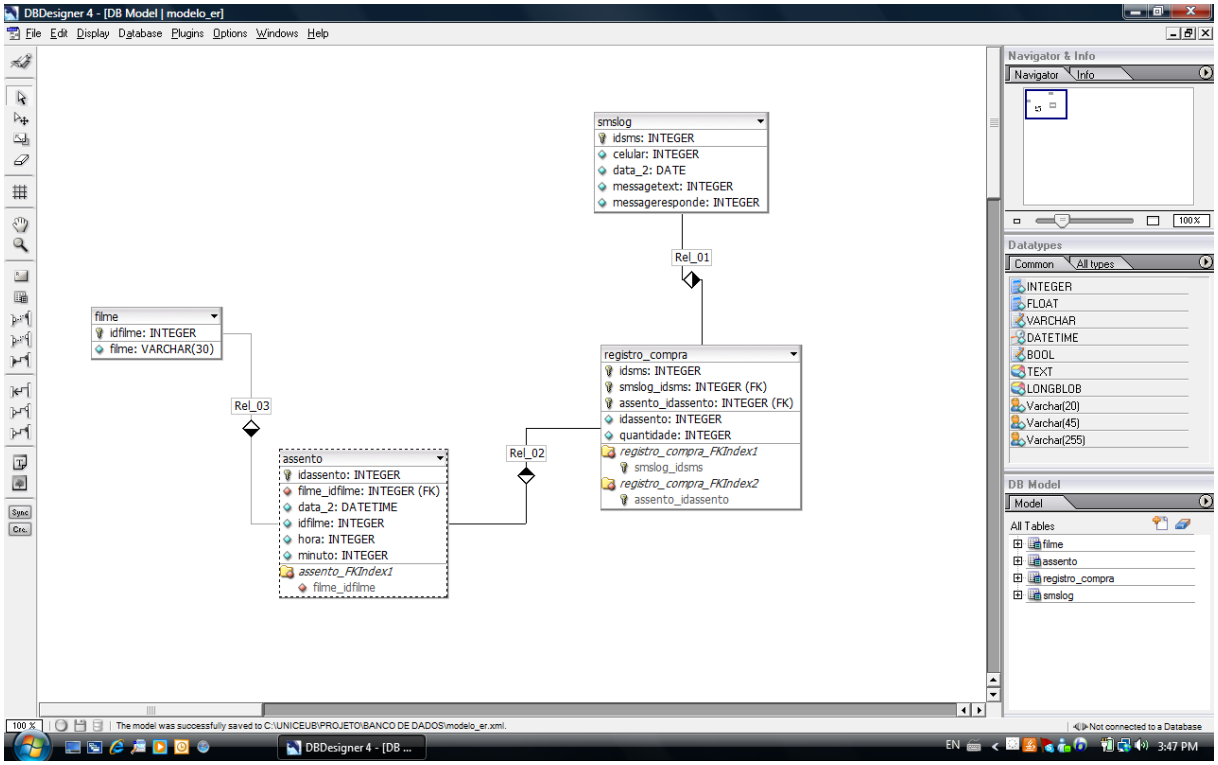


Figura 4.12 – Tela do DB Designer com o Modelo ER da aplicação SMS Cinema.
Fonte (AUTORA DO PROJETO, 2009)

Na figura 4.12 observa-se que o relação 01 entre a tabela *smslog* e a registro_compra é do tipo 1:n. A relação 02 é de 1:n entre registro_compra e assento. A relação 03 é 1:n (*Non-Identifying*) entre filme e assento.

A tabela *smslog* faz o registro de todas as operações que ocorrem no sistema. Essa tabela irá armazenar os dados em formato texto como vieram do banco de dados.

A tabela registro_compra é a tabela que registra a compra de assentos para cada sessão. Nesta tabela serão registradas apenas as compras com sucesso. As demais serão registradas na tabela *smslog*. Cada vez que um registro é inserido nesta tabela, necessariamente deve diminuir um registro de assentos_disponiveis, de acordo com a seguinte fórmula:

assentos.assentos_disponiveis = assentos.assento_disponiveis - registro_compra.qtd

A tabela assento conta a quantidade de assentos disponíveis para o sistema de compra de ingresso usando SMS.

A tabela filme é responsável pelo cadastro de todos os filmes.

Para implementar o modelo de dados no *SQL Server Management Studio 2008* foi criada uma nova *database* no banco de dados chamado smscinema para executar o arquivo SMSCinema.sql conforme ilustrado na figura 4.13:

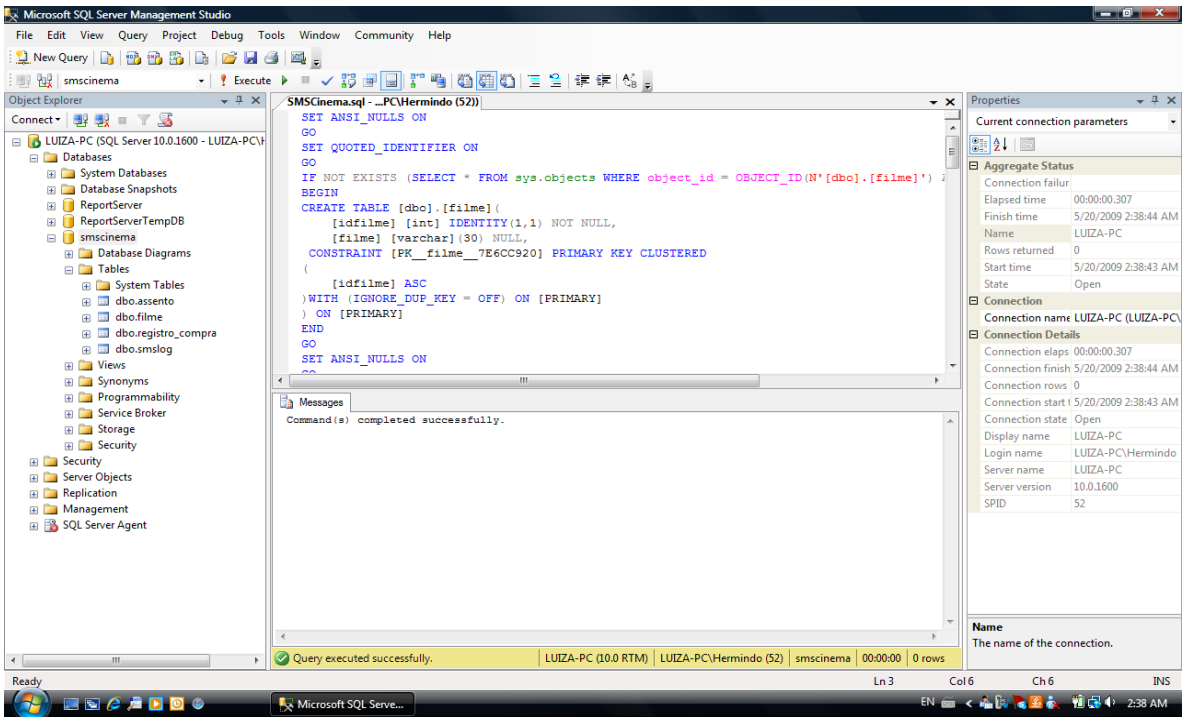


Figura 4.13 – Tela do SQL Server Management Studio.

4.6 Integração entre os módulos

Os passos abaixo enumerados mostra como foi feita a integração entre o módulo SQL e a aplicação SMS Cinema:

Passo 1 – Desmarcar a opção “Ocultar as extensões dos tipos de arquivos conhecidos” na seção ferramentas, opções de pasta, modo de exibição para criar um link com terminação “.udl” e estabelecer a conexão com a base de dados smscinema criada como mostrado na figura 4.14 abaixo:

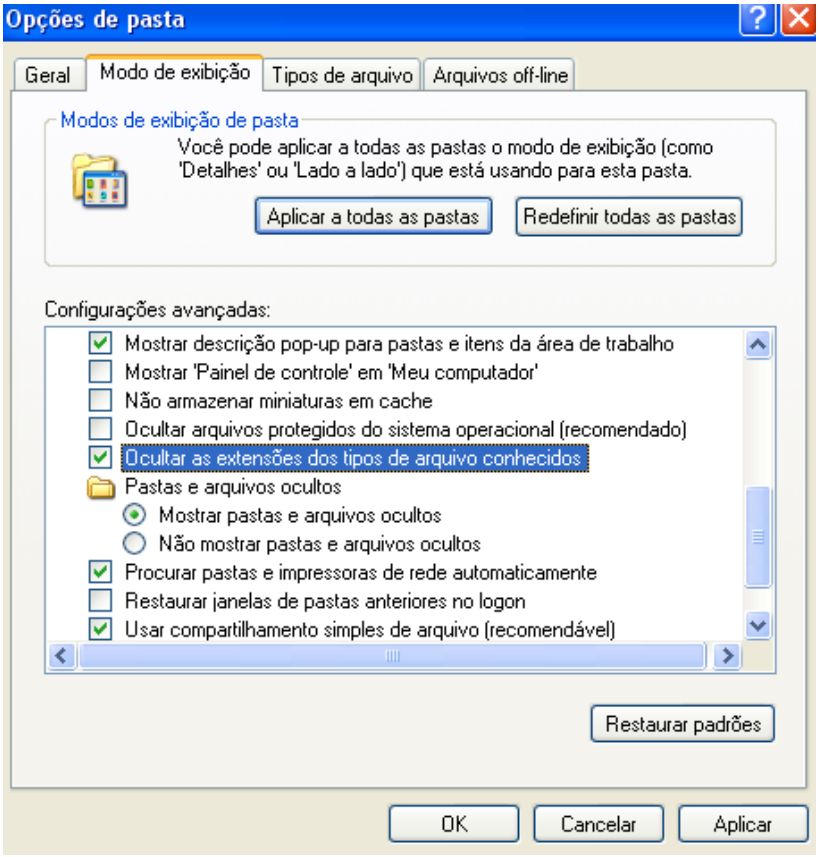


Figura 4.14 Mostra a opção “ocultar as extensões dos tipos de arquivos conhecidos”.

Passo 2 – Do link criado com terminação “.udl” acessar o *Data Link Properties* para selecionar a conexão com smscinema conforme mostrado na figura 4.15 :

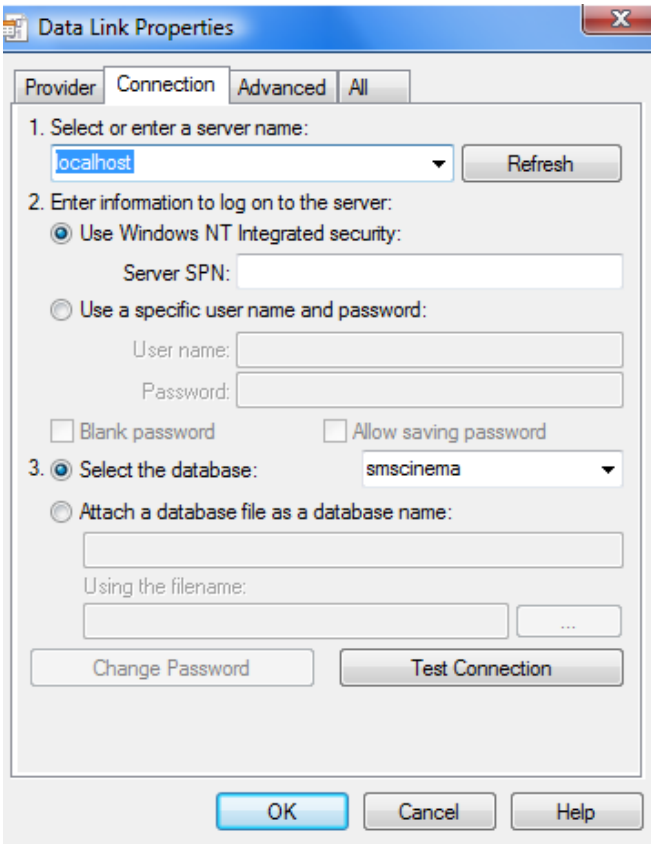


Figura 4.15 – Tela mostra a conexão com o smscinema.

4.7 Problemas e Soluções encontrados

Os problemas encontrados no decorrer do desenvolvimento desse projeto são listados abaixo. As soluções encontradas são indicadas logo em seguida.

- Problema: Pelo HyperTerminal para enviar um SMS é preciso solicitar o envio e em seguida definir o texto para então completar o comando. Para indicar que terminou de escrever a mensagem é preciso pressionar CTRL+Z no teclado. O comando ENTER é formado por um CR (*Carriage Return*) e outro LF (*Line Feed*) enquanto o comando desejado só requer um CR. O problema é que esses caracteres enviados a mais não eram visíveis.
    - Solução: Na programação para evitar precisar pressionar o comando no teclado foi preferível usar o código ASCII 26 correspondente ao pressionar das teclas CTRL+Z.
- Problema: O HyperTerminal fazia o tratamento de caracteres não visíveis na digitação, como o ENTER. No Windows o ENTER é feito por um CR + LF. Entretanto no modem, esperava-se receber apenas um CR ou um LF. Isso tomou muito tempo do desenvolvimento.
    - Solução: Ao detectar o problema, foi passado apenas o código ASCII solicitado para o CR (13) ou LF (10) e não os dois juntos.
- Problema: Erro para excluir filme da interface WEB. Foi encontrada a seguinte mensagem de erro: “The DELETE statement conflicted with the REFERENCE constraint “FK_assento_idfilme_023D5A04.” The conflict occurred in database “smscinema”, table “dbo.assento”, column “idfilme”. The statement has been terminated. “
    - Para resolver esse erro, foi preciso acessar a solução SQL, tabela dbo.assento, opção *Foreign Key Relationships*, editar propriedades de relações existentes na parte do *INSERT And UPDATE* alterar o *Delete Rule* para *Cascade*.

5 EXPERIMENTO E RESULTADOS

5.1 Introdução

Este capítulo visa mostrar os resultados obtidos com a implementação do sistema de compra de ingresso usando SMS. Para verificar a funcionalidade do sistema os primeiros testes foram feitos com a solução SMS Cinema. Essa solução testa a conexão entre o modem e a aplicação e verifica a ocorrência de novas mensagens. Uma vez confirmada a conexão com o modem foi usado o celular GSM para enviar e receber mensagens por essa solução.

Em seguida foi testada a solução ADM Cinema responsável pela interface WEB e pelo cadastro no banco de dados.

Após os primeiros testes começou a fase de experimentação. Nessa fase foi testado o recebimento do SMS e o protocolo de compra para detectar se o mesmo estava funcionando adequadamente. Sendo assim, uma mensagem SMS era enviada do celular para o número do modem e então a aplicação era ativada passo a passo para verificar se tudo estava funcionando adequadamente.

Constitui-se o cenário de experimento o celular GSM, o modem G24 e o servidor local.

5.2 Configuração do ambiente de testes

O ambiente de testes foi estabelecido com um modem G24, um iPhone 2G e a máquina de desenvolvimento com sistema operacional Windows Vista, Driver do Modem, .NET, *SQL Management Studio* 2008*, Visual Studio,* 2008*.*

O cenário de testes foi montado da seguinte forma:

- Passo 1: Abrir a solução SQL no *SQL Server Management Studio 2008*. Criar nova base de dados “smscinema”. Executar a *query* SQL para criar as tabelas dbo.assento, dbo.filme, dbo.registro_compra e dbo.smslog no *database* “smscinema”;
- Passo 2: Conectar o modem G24 no servidor para permitir enviar informações ao computador utilizando uma conexão USB. Verificar em qual porta COM o modem está conectado ao servidor;

- Passo 3: Abrir a solução SMSCinema.sln usando o *Visual Studio*. Na classe *Modem.cs* verificar o número da porta COM para permitir conexão com o modem. Marcar com o botão direito no "ProjetoTeste" desta solução a opção *Set as StartUp Project* e executar;
- Passo 4: Abre uma janela chamada Form1 denominada formulário. Esse janela permite o usuário testar a conexão com o modem. Esse projeto utilizando o G24 permite ao usuário escrever um texto e enviar SMS, listar as mensagens recebidas, apagar mensagens da memória do G24 e verificar o saldo do chip pré-pago utilizado no modem;
- Passo 5: Parar de executar o "ProjetoTeste", mas ainda na solução SMSCinema.sln selecionar o projeto "ServicoModem" e marcar com o botão direito a opção *Set as StartUp Project* para executar. Esse projeto possui um evento que executado a cada 15 segundos verifica por novas mensagens na memória do modem G24. Deixar esse projeto aberto e seguir para o sexto passo;
- Passo 6: Usando o *Visual Studio* executar a segunda solução Adm Cinema.sln que administra as tabelas do banco de dados e é um projeto WEB baseado no framework XMLNuke;
- Passo 7: Fechar essa primeira janela que abre da interface WEB gerada porque quando tenta entrar em uma opção da interface aparece no Visual Studio a seguinte mensagem: "*There is no source code available for the current location*" mas não aparece erro. Se tentar selecionar uma opção novamente a interface WEB procura o endereço, mas não recebe resposta;
- Passo 8: Desse modo é possível abrir a interface WEB pelo ícone no canto inferior direito com o botão direito selecionar a opção *Open in WEB Browser* que todas as opções e as funções podem ser executadas sem intercorrências;
- Passo 9: Na interface WEB fazer cadastro com o nome do filme, horário da sessão, número de assentos totais da sala e número de assentos disponíveis para o sistema de compra de ingresso;
- Passo 10: Enviar do celular GSM um SMS para o número do modem com o texto solicitando a compra;
- Passo 11: Aguardar uma resposta via SMS;

- Passo 12: Verificar o registro de compra ou *log* de mensagens pela interface WEB;
- Passo 13: Fazer uma consulta pelo SQL das tabelas dbo.assento, dbo.filme, dbo.registro_compra e dbo.*smslog*.

5.3 Testes realizados

1) Rotina de testes da solução SMS Cinema:

✓ Caso 1: COMPRA EFETUADA.

- comprar ingressos usando SMS para quantidades variadas. O maior valor para cadastrar assentos disponíveis para o sistema de compra de ingressos no sistema é de até 3 dígitos. Para compra o valor máximo é de apenas 2 dígitos. Esse número de assentos disponíveis é cadastrado pela interface WEB. A figura 5.1 mostra os registros de compras para os seguintes exemplos: 10, 1, 50, 2, 200, 100, 4 e 99 respectivamente.

	data	celular	messagetext	response	filme	hora	minuto	qtd
	09/06/04,21:08:05-140	06181510355	comprar 02 2100 10	COMPRA EFETUADA	Filme Dois	21	0	10
	09/06/04,21:08:27-140	06181510355	comprar 01 2015 1	COMPRA EFETUADA	Filme Um	20	15	1
	09/06/04,21:08:54-140	06181510355	comprar 01 2245 50	COMPRA EFETUADA	Filme Um	22	45	50
	09/06/04,21:12:11-140	06181510355	comprar 02 2100 2	COMPRA EFETUADA	Filme Dois	21	0	2
	09/06/04,21:12:58-140	06181510355	comprar 02 1845 200	COMPRA EFETUADA	Filme Dois	18	45	20
	09/06/04,21:13:55-140	06181510355	comprar 01 2245 100	COMPRA EFETUADA	Filme Um	22	45	10
	09/06/04,21:15:41-140	06181510355	comprar 01 2015 4	COMPRA EFETUADA	Filme Um	20	15	4
	09/06/04,21:21:29-140	06181510355	comprar 01 2245 99	COMPRA EFETUADA	Filme Um	22	45	99

Figura 5.1 – Visualização da tabela Registro de Compra pela interface WEB.
Fonte (AUTORA DO PROJETO, 2009)

Observe que quando o SMS solicitou a compra de 200 ou 100 ingressos o sistema desconsiderou o último algoritmo e efetuou a compra como sendo 20 e 10 respectivamente. O maior valor aceito pelo sistema corresponde a compra de 99 bilhetes conforme mostrado no registro da tabela.

A figura a seguir 5.2 mostra uma consulta análoga a tabela dbo.registro_compra pelo *SQL Server Management Studio*:

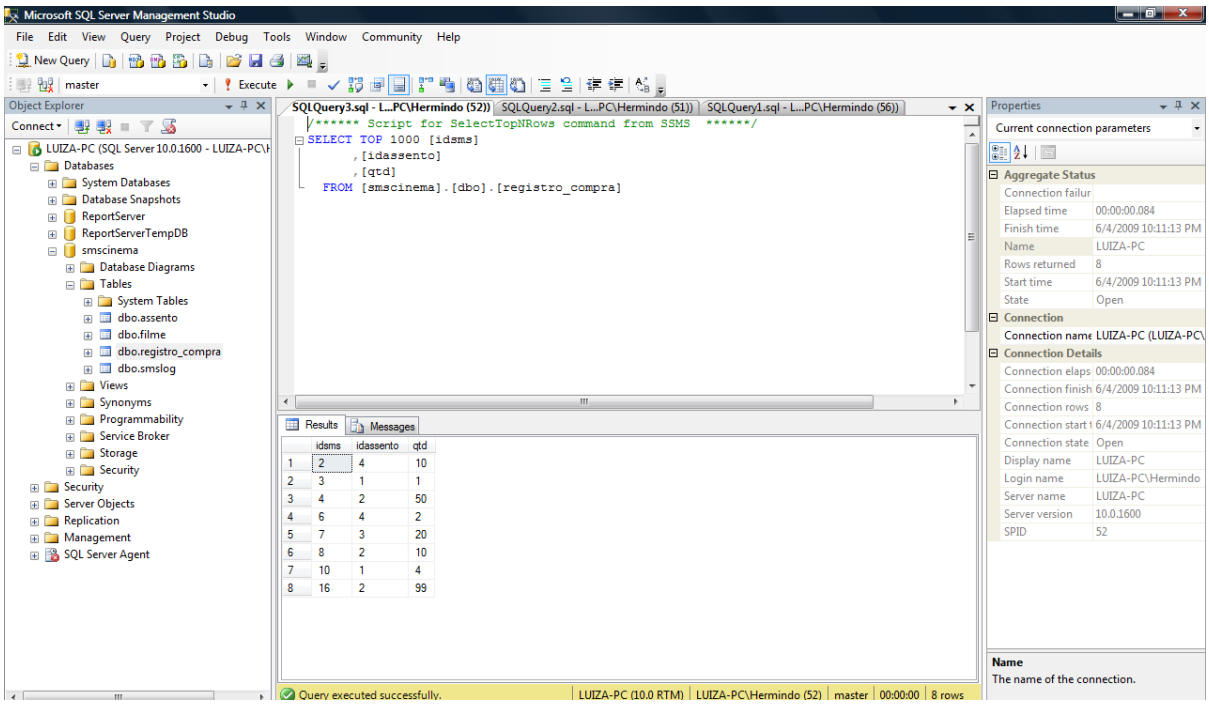


Figura 5.2 – Consulta a tabela dbo.registro_compra pelo *SQL Server Management Studio.*

- ✓ Caso 2: INDISPONIVEL: para todos os casos a seguir expostos foram usados comandos corretos para solicitar a compra de ingressos usando SMS, no entanto, houve equivoco na indicação de um determinado dado.

    - o quando o código do filme está correto mas o horário da sessão especificada não está cadastrada para o filme escolhido;
    - o quando o horário da sessão está cadastrado no sistema mas corresponde a outro código de filme.
    - o quando um SMS solicita: comprar n ingressos e tem apenas n-1 disponível para a sessão especificada na mensagem;
    - o quando não há disponíveis para sessão especificada;
    - o no caso de solicitar a compra de “zero” ingresso;

Pela opção Ver Registro do SMS pela interface WEB acessa-se a tabela da figura 5.3 que mostra os registros dos casos que retornam INDISPONIVEL para o usuário conforme os casos descritos acima respectivamente:

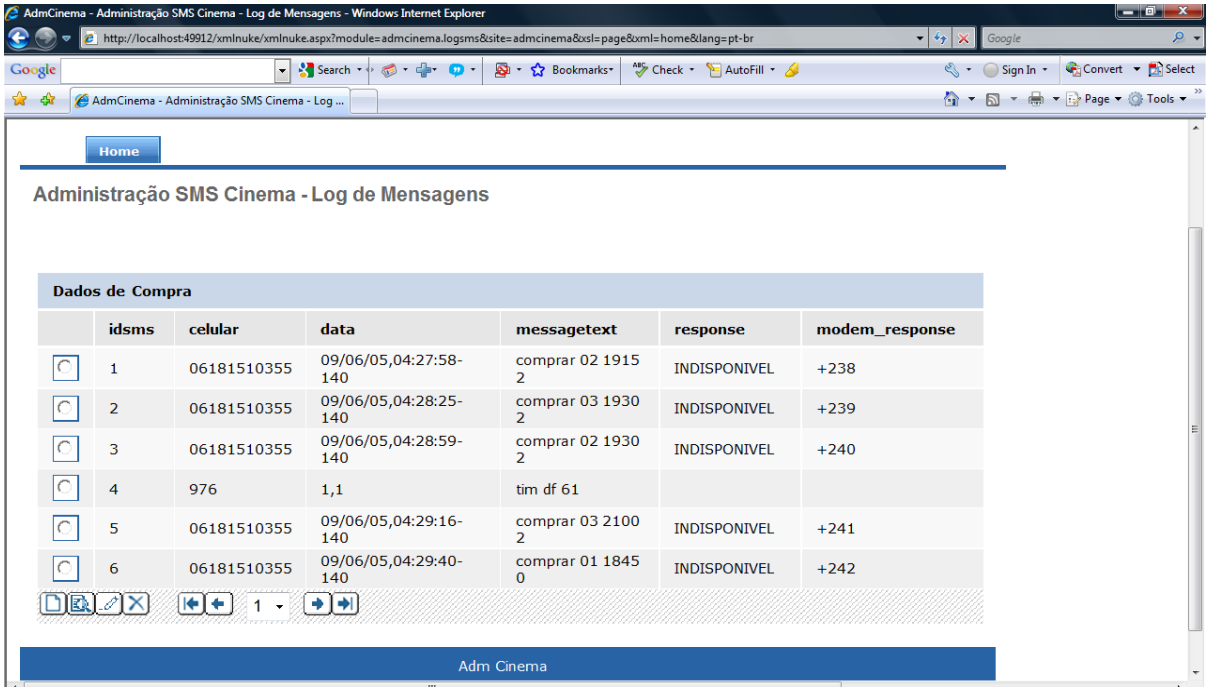


Figura 5.3 – Mostra os registros da tabela dbo.*smslog* que retornou para o usuário a mensagem INDISPONÍVEL.
Fonte (AUTORA DO PROJETO, 2009)

A figura 5.4 traz uma consulta pelo *SQL Server Management Studio* e exibe os mesmos resultados observados pela interface WEB na figura 5.3.

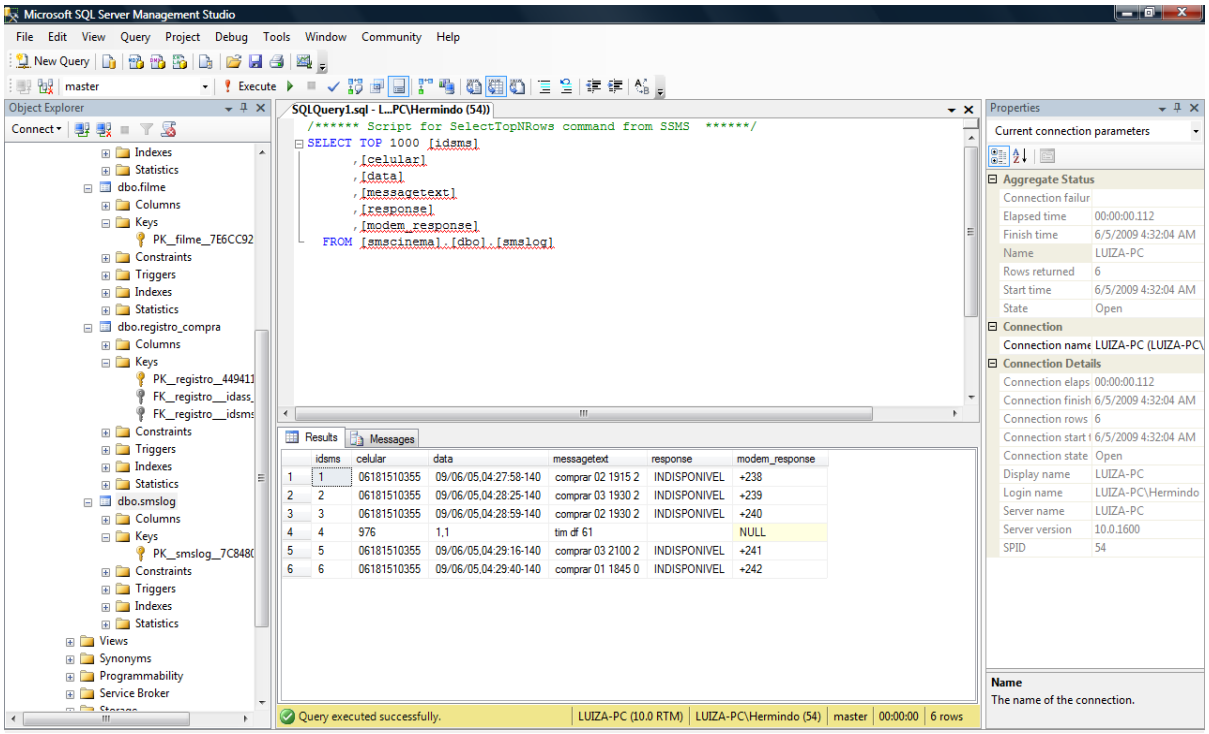
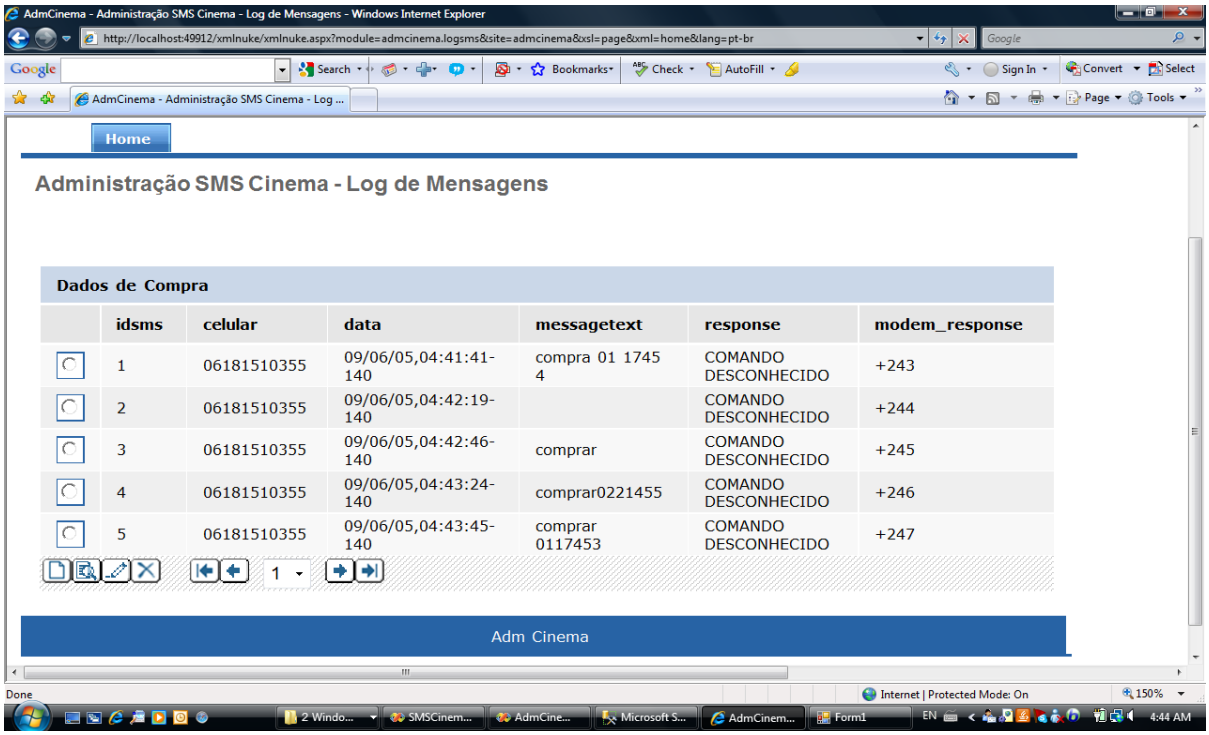


Figura 5.4 – Consulta a tabela dbo.*smslog* pelo *SQL Server Management Studio.*

- ✓ Caso 3: COMANDO DESCONHECIDO:
  - o quando não reconhece os caracteres recebidos do SMS;
  - o quando o comando para solicitar a compra começa na segunda linha da caixa de texto que recebe o texto ele é descartado porque o modem não consegue fazer o reconhecimento quando tem muito espaço no início da mensagem;
  - o escrever cada trecho do comando em uma linha diferente com caracteres válidos ou inválidos;
  - o escrever os caracteres corretamente, mas sem espaço entre os campos;
  - o escrever o comando com apenas um espaço entre a palavra e os números.

Observe na figura abaixo 5.5 os registros para as situações que o sistema retorna para o usuário uma mensagem de COMANDO DESCONHECIDO.



	idsms	celular	data	messagetext	response	modem_response
	1	06181510355	09/06/05,04:41:41-140	compra 01 1745 4	COMANDO DESCONHECIDO	+243
	2	06181510355	09/06/05,04:42:19-140		COMANDO DESCONHECIDO	+244
	3	06181510355	09/06/05,04:42:46-140	comprar	COMANDO DESCONHECIDO	+245
	4	06181510355	09/06/05,04:43:24-140	comprar0221455	COMANDO DESCONHECIDO	+246
	5	06181510355	09/06/05,04:43:45-140	comprar 0117453	COMANDO DESCONHECIDO	+247

Figura 5.5 – Mostra os registros para comando desconhecido da tabela Log de Mensagens pela interface WEB.
Fonte (AUTORA DO PROJETO, 2009)

Quando feita uma comparação utilizando o *SQL Server Management Studio* é possível encontrar o mesmo resultado para os registros que retornaram COMANDO DESCONHECIDO como mostra a figura 5.6:

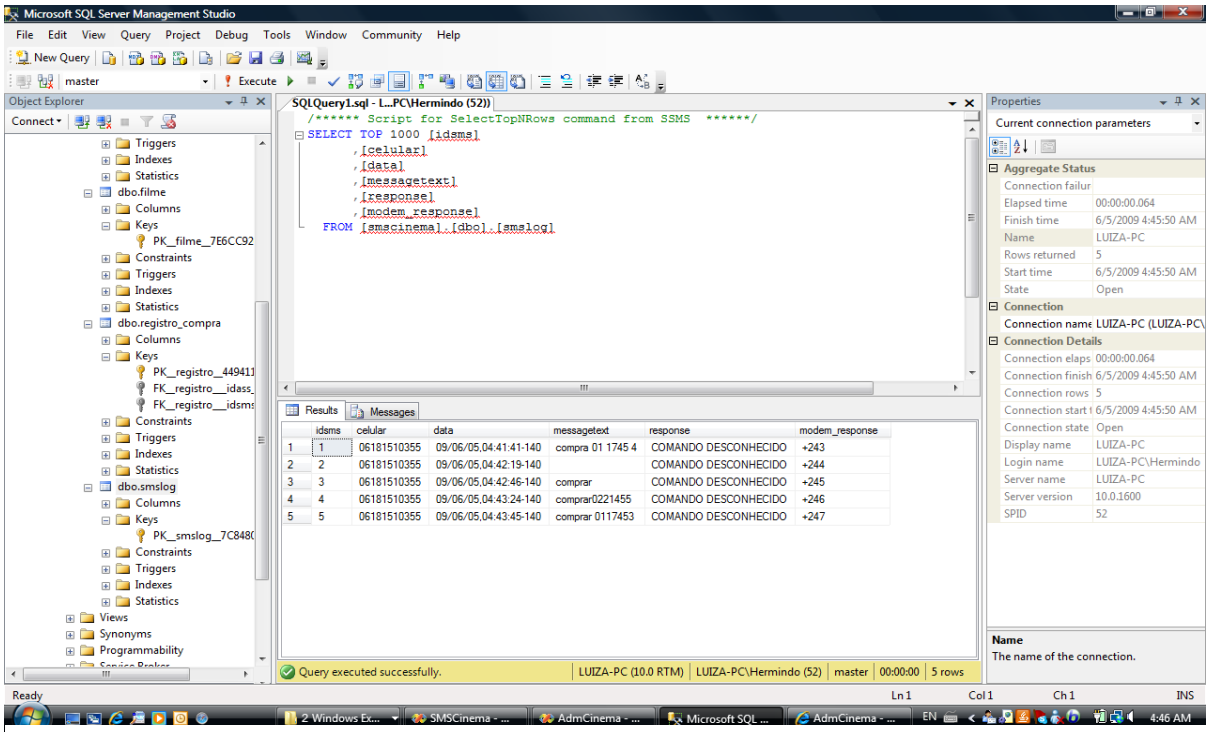


Figura 5.6 – Consulta a dbo.*smslog* pelo *SQL Server Management Studio.*

2) Rotina para testes da solução ADM Cinema utilizando a interface WEB:

- ✓ Caso 1: cadastrar um novo filme. Selecionar a opção “Cadastrar Filmes e Assentos disponíveis”, novo item. Insere o nome do filme e enviar os dados. Seleciona o filme na tabela, em seguida opção editar assentos disponíveis, depois novo item. Preencher as caixas de diálogo com a data atual, seleciona da lista o nome do filme que foi cadastrado no passo anterior, determina hora e minuto da sessão, quantidade de assentos totais da sala de cinema e assentos disponíveis para o sistema de compra de ingresso usando SMS observando se julgar necessário uma proporção de 20% entre o número total de assentos e o número de assentos disponíveis para o sistema;
- ✓ Caso 2: Incluir uma nova sessão para um filme cadastrado no cinema pela opção “Cadastrar Filmes e Assentos disponíveis”, seleciona o filme na tabela, em seguida opção editar assentos disponíveis, depois novo item. Preencher as caixas de diálogo com a data atual seleciona o nome do filme que deseja incluir nova sessão, determina hora e minuto da sessão, quantidade de

assentos totais da sala de cinema e assentos disponíveis para o sistema usando SMS;

- ✓ Caso 3: Deletar apenas uma sessão do filme cadastrado no sistema. Opção "Cadastrar Filmes e Assentos disponíveis", seleciona o filme na tabela, em seguida opção editar assentos disponíveis, depois marcar o campo da sessão que deseja apagar e selecionar o comando apagar;
- ✓ Caso 4: Deletar um filme cadastrado no sistema. Opção cadastrar filmes e assentos disponíveis, marcar o filme na tabela, em seguida comando apagar;
- ✓ Caso 5: Editar a data, o horário da sessão, assentos totais e assentos disponíveis de um filme cadastrado. Selecionar a opção "Cadastrar Filmes e Assentos disponíveis", seleciona o filme na tabela, em seguida opção editar assentos disponíveis, marcar a sessão que deseja alterar o horário e selecionar o comando editar. Preencher as caixas de diálogo com a data atual, seleciona o nome do filme, determina hora e minuto da sessão, quantidade de assentos totais da sala de cinema e assentos disponíveis para o sistema de compra de ingressos usando SMS.
- ✓ Caso 6: Editar o nome de um filme cadastrado. Opção cadastrar filmes e assentos disponíveis, seleciona o filme na tabela e pressionar o comando editar em seguida enviar dados.
- ✓ Caso 7: Verificar o registro de compra pela opção "Consultar Registro de Compra" da tela principal da interface WEB.
- ✓ Caso 8: Verificar o *log* de mensagens pela opção "Ver Registro do SMS" da tela principal da interface WEB.
- ✓ Caso 9: Fazer uma consulta SQL nas tabelas dbo.assento, dbo.filme, dbo.registro_compra e dbo.*smslog* pelo *SQL Server Management Studio* 2008.

As fotos desses experimentos estão no apêndice IV deste documento.

6 CONCLUSÕES

6.1 Conclusões

O projeto SMS Cinema foi criado com o objetivo de simular um sistema de compra de ingressos de cinema usando SMS. A essa tarefa seriam acrescidas duas outras: receber uma resposta via SMS no celular do usuário, cadastrar filmes e assentos disponíveis através de uma interface WEB.

Para desenvolver o projeto, a dificuldade inicial encontrada foi o primeiro contato com o HyperTerminal que foi bastante desanimador. Havia apenas uma tela branca sem nenhuma outra informação. Tudo o que se digitava não aparecia na tela. Entretanto, ao se pressionar “ENTER”, voltava uma mensagem “ERROR”. Depois de muito ler e tentar ficou entendido que se deveria digitar os comandos AT conforme eram definidos. Em seguida descobriu-se o comando para ecoar na tela o que era digitado. A partir desse momento começou a fluir melhor a utilização com o HyperTerminal.

O sistema foi desenvolvido em duas etapas: Solução SMS Cinema e solução Adm Cinema utilizando C# e XMLNuke. Fez uso de soluções Microsoft. O Visual Studio foi a IDE escolhida e o Microsoft SQL Server Management Studio 2008 foi usado para criar as tabelas que armazenam os dados dos registros.

Em seguida foi iniciada a fase de testes e experimentos que permitiram avaliar o desempenho do sistema desenvolvido.

Os resultados obtidos demonstram um funcionamento satisfatório do sistema de compra de ingressos e do serviço de SMS da telefonia móvel. O resultado esperado confere com o objetivo proposto no projeto.

6.2 Sugestões de trabalhos futuros

O projeto ora desenvolvido mostra que é possível o envio de SMS, o seu tratamento e a resposta correspondente à mensagem enviada.

Fica como sugestão de trabalhos futuros a elaboração de estudo mais aprofundado com vistas à evolução do sistema de pagamento e do modo de integração do sistema com as operadoras. Outra sugestão é a análise de uma forma de se integrar o sistema desenvolvido com a bilheteria do cinema, de forma a permitir o acesso mais rápido e seguro do cliente à respectiva sala de sessão.

REFERENCIAL BIBLIOGRÁFICO

1. JARGAS, A. M.. Expressões regulares: uma abordagem divertida 2ª Ed. Novatec, São Paulo, 2008.

2. SVERZUT, J. U. . Redes GSM, GPRS, EDGE e UMTS: Evolução a Caminho da Quarta Geração (4G) 2ª Ed. ÉRICA, São Paulo, 2008

3. VIEIRA, H. P. . Proposta de Política de Adaptação de Enlace EDGE como Caminho de Evolução para Provimento de Serviços IMS de Terceira Geração (Distrito Federal). Dissertação de Mestrado Profissionalizante em Engenharia Elétrica, Publicação PPGENE.DM-055/08,Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF,99p., 2008.

4. ANATEL (notícia: Anatel revisa distribuição de celulares por tecnologia 02 de July de 2009). Disponível em:
< http://www.anatel.gov.br/Portal/exibirPortalInternet.do >
Acesso: 06 de julho de 2007.

5. APPLE. Disponível em:
< http://www.apple.com/br/iphone/ >
Acesso: 25 de março de 2009.

6. C# *and AT Command.* Disponível em:
< http://social.msdn.microsoft.com/Forums/en-US/netfxnetcom/thread/800f12fa-4da8-42bf-b61b-65f6bb58c871 >
Acesso: 14 de março de 2009.

7. C# Regular Expressions - The Basics. 2009. Disponível em:
< http://www.mikesdotnetting.com/Article.aspx?ArticleID=50 >
Acesso: 14 de março de 2009.

8. Criando uma aplicação XMLNuke em CSharp. Disponível em:
<http://www.xmlnuke.com/site/xmlnuke.php?site=docs&xml=creatingapp_csharp&xsl=page&lang=pt-br >
Acesso: 21 de março de 2009.

9. Explicando os parâmetros do ProcessPageField. Disponível em:
<http://www.xmlnuke.com/site/xmlnuke.php?site=docs&xml=createmodule_5&xsl=page&lang=pt-br#H3 >
Acesso: 21 de março de 2009.

10. Expressões Regulares (Armamento Pesado). Publicado 22/08/2008. Disponível em:
< http://www.linhadecodigo.com.br/ArtigoImpressao.aspx?id=2015 >
Acesso: 14 de março de 2009.

11. INFO WESTER. Disponível em:
< http://www.infowester.com/usb.php >
Acesso: 24 de março de 2009.

12. Informat Technology. Disponível em:
<http://www.informattechnology.com.br/catalogo/wireless_catalogo.php?cat_id=17&pro_id=13 >
Acesso: 01 de março de 2009.

13. Mobile Messaging Technologies and Services: SMS, EMS and MMS. Published Online: 21 Jul 2003. Author(s): Gwenaël Le Bodic. Print ISBN: 9780470848760 Online ISBN: 9780470858080. Copyright © 2003 John Wiley & Sons, Ltd < http://www3.interscience.wiley.com/cgi-bin/summary/104547659/SUMMARY?CRETRY=1&SRETRY=0 >
Acesso: 27 de abril de 2009.

14. MSDN, 2009 “Getting Started with ASP.NET”. Disponível em:
< http://msdn.microsoft.com/en-us/asp.net/bb418488.aspx >
Acesso: 23 de março de 2009.

15. MSDN. Microsoft Developer Network. Visual C# 2009. Disponível em:
< http://msdn.microsoft.com/pt-br/library/kx37x362.aspx >
Acesso: 23 de março de 2009.

16. Netpedia, 2009 “Backus-Naur form forma de Backus-Naur”. Disponível em:
< http://www.netpedia.com.br/MostraTermo.php?TermID=470 >
Acesso: 26 de março de 2009

17. *Sending SMS using .NET*. Última atualização: 11/06/2007. Disponível em:
<
http://www.codeproject.com/KB/IP/Sending_SMS_using_Net.aspx?fid=424247&df=90&mpp=25&noise=3&sort=Position&view=Quick&fr=51 >
Acesso: 20 de março de 2009.

18. WIKIPEDIA “Backus-Naur Form ” Disponível em:
< http://pt.wikipedia.org/wiki/Formalismo_de_Backus-Naur >
Acesso: 26 de março de 2009.

19. Wille, C. . Apresentando C# Ed. Berkeley, São Paulo, 2001.

20. WNEWS. Disponível em:
<http://wnews.uol.com.br/site/noticias/materia.php?id_secao=4&id_conteudo=11503
>
Acesso: 24 de março de 2009.

21. XMLNuke. Disponível em:
< http://www.xmlnuke.com/site/ >
Acesso: 21 de março de 2009.

APÊNDICE I – CÓDIGOS DE PROGRAMAÇÃO - SOLUÇÃO SMS CINEMA

Este é um arquivo gerado automaticamente pelo Visual Studio e contém as informações para definir as propriedades do arquivo “ProjetoTeste” compilado.

```
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;

// General Information about an assembly is controlled through the
following
// set of attributes. Change these attribute values to modify the
information
// associated with an assembly.
[assembly: AssemblyTitle("ProjetoTeste")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyProduct("ProjetoTeste")]
[assembly: AssemblyCopyright("Copyright ©  2009")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]

// Setting ComVisible to false makes the types in this assembly not visible
// to COM components.  If you need to access a type in this assembly from
// COM, set the ComVisible attribute to true on that type.
[assembly: ComVisible(false)]

// The following GUID is for the ID of the typelib if this project is
exposed to COM
[assembly: Guid("7cf1a295-7c30-43af-b9f1-8561a6a8225d")]

// Version information for an assembly consists of the following four
values:
//
//      Major Version
//      Minor Version
//      Build Number
//      Revision
//
// You can specify all the values or you can default the Build and Revision
Numbers
// by using the '*' as shown below:
// [assembly: AssemblyVersion("1.0.*")]
[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyFileVersion("1.0.0.0")]
```

```
//------------------------------------------------------------------------------
// <auto-generated>
//     This code was generated by a tool.
//     Runtime Version:2.0.50727.1433
//
//     Changes to this file may cause incorrect behavior and will be lost if
//     the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------

namespace ProjetoTeste.Properties
{


      /// <summary>
      ///   A strongly-typed resource class, for looking up localized strings, etc.
      /// </summary>
      // This class was auto-generated by the StronglyTypedResourceBuilder
      // class via a tool like ResGen or Visual Studio.
      // To add or remove a member, edit your .ResX file then rerun ResGen
      // with the /str option, or rebuild your VS project.
      [global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Resources.Tools.StronglyTypedResourceBuilder", "2.0.0.0")]
      [global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
      [global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]
      internal class Resources
      {

            private static global::System.Resources.ResourceManager resourceMan;

            private static global::System.Globalization.CultureInfo resourceCulture;


      [global::System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance", "CA1811:AvoidUncalledPrivateCode")]
            internal Resources()
            {
            }

            /// <summary>
            ///   Returns the cached ResourceManager instance used by this class.
            /// </summary>

      [global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.EditorBrowsableState.Advanced)]
            internal static global::System.Resources.ResourceManager ResourceManager
            {
                  get
                  {
                        if ((resourceMan == null))
                        {
```

```
                              global::System.Resources.ResourceManager temp
= new
global::System.Resources.ResourceManager("ProjetoTeste.Properties.Resources
", typeof(Resources).Assembly);
                              resourceMan = temp;
                        }
                        return resourceMan;
                  }
            }

            /// <summary>
            ///   Overrides the current thread's CurrentUICulture property
for all
            ///   resource lookups using this strongly typed resource
class.
            /// </summary>

      [global::System.ComponentModel.EditorBrowsableAttribute(global::Syste
m.ComponentModel.EditorBrowsableState.Advanced)]
            internal static global::System.Globalization.CultureInfo
Culture
            {
                  get
                  {
                        return resourceCulture;
                  }
                  set
                  {
                        resourceCulture = value;
                  }
            }
      }
}
```

```
//------------------------------------------------------------------------------
// <auto-generated>
//     This code was generated by a tool.
//     Runtime Version:2.0.50727.1433
//
//     Changes to this file may cause incorrect behavior and will be lost if
//     the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------

namespace ProjetoTeste.Properties
{


      [global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]
      [global::System.CodeDom.Compiler.GeneratedCodeAttribute("Microsoft.VisualStudio.Editors.SettingsDesigner.SettingsSingleFileGenerator", "9.0.0.0")]
      internal sealed partial class Settings : global::System.Configuration.ApplicationSettingsBase
      {

            private static Settings defaultInstance = ((Settings)(global::System.Configuration.ApplicationSettingsBase.Synchronized(new Settings())));

            public static Settings Default
            {
                  get
                  {
                        return defaultInstance;
                  }
            }
      }
}
```

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;

namespace ProjetoTeste
{
      static class Program
      {
            /// <summary>
            /// The main entry point for the application.
            /// </summary>
            [STAThread]
            static void Main()
            {
                  Application.EnableVisualStyles();
                  Application.SetCompatibleTextRenderingDefault(false);
                  Application.Run(new Form1());
            }
      }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using SMSClasses;
using System.Text.RegularExpressions;

namespace ProjetoTeste
{
      public partial class Form1 : Form
      {
        public Form1()
            {
                  InitializeComponent();
            }
        //Comando Teste para testar a conexão com o modem
            private void button1_Click(object sender, EventArgs e)
            {
                  ConsoleWriter console = new ConsoleWriter(this.textBox1);
                  Modem m = new Modem(console);
                  Commands commands = new Commands(m);

                  console.Write(commands.cmdConnectionTest());
               console.Write("End");

                  commands.Dispose();
            }
        //Comando (Direto) para testar a conexão com a porta serial
            private void button2_Click(object sender, EventArgs e)
            {
            ConsoleWriter console = new ConsoleWriter(this.textBox1);
            Modem m = new Modem(console);

                  m.Open();
                  m.SendData("ATI");
                  console.Write(m.ReceiveData());
                  m.Close();

                  console.Write("End");
            }
        //Comando Limpar para limpar a tela do formulário do ProjetoTeste
            private void button3_Click(object sender, EventArgs e)
            {
                  this.textBox1.Text = "";
            }
        //Comando Listar Msg para listar todas as mensagens enviadas para o
modem.
            private void button4_Click(object sender, EventArgs e)
            {
            ConsoleWriter console = new ConsoleWriter(this.textBox1);
            Modem m = new Modem(console);
            Commands commands = new Commands(m);

                  MatchCollection mc = commands.cmdGetSMSList();

                  console.Write("End");

                  commands.Dispose();
```

```
            }
        //Comando Apaga Msg para apagar uma mensagem da lista de memória do
modem através do envio de um comando AT.
            private void button5_Click(object sender, EventArgs e)
            {
            ConsoleWriter console = new ConsoleWriter(this.textBox1);
            Modem m = new Modem(console);
            Commands commands = new Commands(m);

                  commands.cmdApagaMensagem(textBox2.Text);

                  console.Write("End");

                  commands.Dispose();
            }
        //Comando Enviar SMS usado para enviar um SMS utilizando a tela do
formulário do ProjetoTeste para inserir o número de celular e o texto da
mensagem para ser enviada através do envio de um comando AT.
            private void button6_Click(object sender, EventArgs e)
            {
            ConsoleWriter console = new ConsoleWriter(this.textBox1);
            Modem m = new Modem(console);
            Commands commands = new Commands(m);

                  commands.cmdEnviaSMS(textBox3.Text, textBox4.Text);

                  console.Write("End");

                  commands.Dispose();
            }
        //Comando Saldo para consultar o saldo disponível do chip pré-pago
contido no modem G24 através do envio de um comando AT.
            private void button7_Click(object sender, EventArgs e)
            {
            ConsoleWriter console = new ConsoleWriter(this.textBox1);
            Modem m = new Modem(console);
            Commands commands = new Commands(m);

                  commands.cmdVerificarSaldo();

                  console.Write("End");

                  commands.Dispose();
            }

        private void textBox3_TextChanged(object sender, EventArgs e)
        {

        }
      }
}
```

Este é um arquivo gerado automaticamente pelo Visual Studio e contém as informações para definir as propriedades do arquivo “ServicoModem” compilado.

```
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;

// General Information about an assembly is controlled through the
following
// set of attributes. Change these attribute values to modify the
information
// associated with an assembly.
[assembly: AssemblyTitle("ServicoModem")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyProduct("ServicoModem")]
[assembly: AssemblyCopyright("Copyright ©  2009")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]

// Setting ComVisible to false makes the types in this assembly not visible
// to COM components.  If you need to access a type in this assembly from
// COM, set the ComVisible attribute to true on that type.
[assembly: ComVisible(false)]

// The following GUID is for the ID of the typelib if this project is
exposed to COM
[assembly: Guid("ba8fe28e-4aff-4bb9-9f18-a233706f6f6b")]

// Version information for an assembly consists of the following four
values:
//
//      Major Version
//      Minor Version
//      Build Number
//      Revision
//
// You can specify all the values or you can default the Build and Revision
Numbers
// by using the '*' as shown below:
// [assembly: AssemblyVersion("1.0.*")]
[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyFileVersion("1.0.0.0")]
```

```
//------------------------------------------------------------------------------
// <auto-generated>
//     This code was generated by a tool.
//     Runtime Version:2.0.50727.1433
//
//     Changes to this file may cause incorrect behavior and will be lost if
//     the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------

namespace ServicoModem.Properties
{


      /// <summary>
      ///   A strongly-typed resource class, for looking up localized strings, etc.
      /// </summary>
      // This class was auto-generated by the StronglyTypedResourceBuilder
      // class via a tool like ResGen or Visual Studio.
      // To add or remove a member, edit your .ResX file then rerun ResGen
      // with the /str option, or rebuild your VS project.
      [global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Resources.Tools.StronglyTypedResourceBuilder", "2.0.0.0")]
      [global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
      [global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]
      internal class Resources
      {

            private static global::System.Resources.ResourceManager resourceMan;

            private static global::System.Globalization.CultureInfo resourceCulture;


      [global::System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance", "CA1811:AvoidUncalledPrivateCode")]
            internal Resources()
            {
            }

            /// <summary>
            ///   Returns the cached ResourceManager instance used by this class.
            /// </summary>

      [global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.EditorBrowsableState.Advanced)]
            internal static global::System.Resources.ResourceManager ResourceManager
            {
                  get
                  {
                        if ((resourceMan == null))
                        {
```

```
                                global::System.Resources.ResourceManager temp
= new
global::System.Resources.ResourceManager("ServicoModem.Properties.Resources
", typeof(Resources).Assembly);
                                resourceMan = temp;
                        }
                        return resourceMan;
                }
        }

        /// <summary>
        ///   Overrides the current thread's CurrentUICulture property
for all
        ///   resource lookups using this strongly typed resource
class.
        /// </summary>

    [global::System.ComponentModel.EditorBrowsableAttribute(global::Syste
m.ComponentModel.EditorBrowsableState.Advanced)]
        internal static global::System.Globalization.CultureInfo
Culture
        {
                get
                {
                        return resourceCulture;
                }
                set
                {
                        resourceCulture = value;
                }
        }
    }
}
```

```
//------------------------------------------------------------------------------
// <auto-generated>
//     This code was generated by a tool.
//     Runtime Version:2.0.50727.1433
//
//     Changes to this file may cause incorrect behavior and will be lost if
//     the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------

namespace ServicoModem.Properties
{


      [global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]
      [global::System.CodeDom.Compiler.GeneratedCodeAttribute("Microsoft.VisualStudio.Editors.SettingsDesigner.SettingsSingleFileGenerator", "9.0.0.0")]
      internal sealed partial class Settings : global::System.Configuration.ApplicationSettingsBase
      {

            private static Settings defaultInstance = ((Settings)(global::System.Configuration.ApplicationSettingsBase.Synchronized(new Settings())));

            public static Settings Default
            {
                  get
                  {
                        return defaultInstance;
                  }
            }
      }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace ServicoModem
{
      static class Program
      {
            /// <summary>
            /// The main entry point for the application.
            /// </summary>
            [STAThread]
            static void Main()
            {
                  Application.EnableVisualStyles();
                  Application.SetCompatibleTextRenderingDefault(false);
                  Application.Run(new Form1());
            }
      }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using SMSClasses;
using System.Text.RegularExpressions;

namespace ServicoModem
{
      public partial class Form1 : Form
      {
            protected int _contador = 0;
            protected int _disparar = 0;

            public Form1()
            {
                  InitializeComponent();
            }
        //Comando Iniciar para iniciar consultar a lista de memória do
modem em intervalos regulares.
            private void btnStart_Click(object sender, EventArgs e)
            {
                  btnStop.Enabled = true;
                  btnStart.Enabled = false;
                  txtTempo.Enabled = false;

                  this._disparar = Convert.ToInt32(txtTempo.Text);
                  this._contador = this._disparar-1;

            timer1.Enabled = true;

            }
        //Comando Parar pára a consulta ao modem em intervalos regulares.
            private void btnStop_Click(object sender, EventArgs e)
            {
                  btnStop.Enabled = false;
                  btnStart.Enabled = true;
                  txtTempo.Enabled = true;
                  timer1.Enabled = false;
                  lblStatus.Text = "<< Parado >>";
            }
        //Faz a contagem regressiva para a próxima verificação da lista de
memória do modem G24.
            private void timer1_Tick(object sender, EventArgs e)
            {
                  if (this._contador != this._disparar-1)
                  {
                        this._contador = (this._contador + 1) %
this._disparar;
                        lblStatus.Text = "Próxima verificação em " +
(this._disparar - this._contador).ToString() + " segundos.";
                return;
                  }

                  timer1.Enabled = false;

                  lblStatus.Text = "Iniciando...";
```

```
                // Inicializa Modem
                Modem modem = new Modem(new NullWriter());
                //Modem modem = new Modem(new
ConsoleWriter(this.txtLog));
                Commands command = new Commands(modem);
                Protocol protocol = new Protocol();
            //Método que verifica a lista de memória do modem G24 em busca
de novas mensagens através do envio de um comando AT.
            MatchCollection mc = null;
                try
                {
                      mc = command.cmdGetSMSList();
                }
                catch
                {
                ///<summary>
                ///Se não tiver mensagens na lista gera um erro que deve
                ser abafado
                /// porque ele espera receber uma quantidade de
                mensagens.
                /// Caso não as receba ele traz null no matching da
                expressão regular e por isso retorna um erro.
               /// </summary>
                }
            //Mostra na tela quantas novas mensagens foram encontradas.
            if ((mc != null) && (mc.Count > 0))
                {
                      txtLog.AppendText(DateTime.Now.ToString() + " -
Encontrei " + mc.Count.ToString() + " novas mensagens.\r\n");

                foreach (Match m in mc)
                       {
                              string position = m.Groups["position"].Value;
                   string type = m.Groups["type"].Value;
                              string phone = m.Groups["phone"].Value;
                              string text = m.Groups["text"].Value;
                              string date = m.Groups["date"].Value;

                              // Só valida mensagens que foram recebidas!
                              if (type.ToUpper().IndexOf("SENT") < 0)
                              {
                        //Processa as mensagens válidas.
                        RetornoProtocolo result =
protocol.Processa(position, type, phone, text, date);
                        //Envia o resultado encontrado pelo sistema via SMS
para o usuário através do envio de um comando AT e atualiza a tabela
smslog.
                                    if
(!String.IsNullOrEmpty(result.mensagem))
                                    {
                               string resposta = command.cmdEnviaSMS(phone,
result.mensagem);
                                          Database db = new Database();
                                          db.AtualizaLog(result.idlog,
resposta);
                                          txtLog.AppendText("Msg #" +
result.idlog.ToString() + " " + phone + " '" + text.Trim() + "' " +
result.mensagem + "\r\n");

                                    }
```

```
                        //Faz o registro na tabela smslog das mensagens
encontradas que possuem número de telefone inválido para o sistema.
                        else
                                    {
                                        txtLog.AppendText("Msg #" +
result.idlog.ToString() + " " + phone + " '" + text.Trim() + "'\r\n");



                                    }
                                }
                      //Comando que apaga as mensagens lidas e registradas no
banco de dados da lista do modem através do envio de um comando AT.
                        command.cmdApagaMensagem(position); //
                            }

                    }
                    modem.Close();

                    // Reinicia verificação
                    this._contador = 0;
                    timer1.Enabled = true;
            }

      }
}
```

Este é um arquivo gerado automaticamente pelo Visual Studio e contém as informações para definir as propriedades do arquivo “SMSClasses” compilado.

```
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;

// General Information about an assembly is controlled through the
following
// set of attributes. Change these attribute values to modify the
information
// associated with an assembly.
[assembly: AssemblyTitle("SMSClasses")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyProduct("SMSClasses")]
[assembly: AssemblyCopyright("Copyright ©  2009")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]

// Setting ComVisible to false makes the types in this assembly not visible
// to COM components.  If you need to access a type in this assembly from
// COM, set the ComVisible attribute to true on that type.
[assembly: ComVisible(false)]

// The following GUID is for the ID of the typelib if this project is
exposed to COM
[assembly: Guid("6619a2f1-f3c4-471e-b69f-88b95ee51418")]

// Version information for an assembly consists of the following four
values:
//
//      Major Version
//      Minor Version
//      Build Number
//      Revision
//
// You can specify all the values or you can default the Build and Revision
Numbers
// by using the '*' as shown below:
// [assembly: AssemblyVersion("1.0.*")]
[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyFileVersion("1.0.0.0")]
```

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Text.RegularExpressions;

namespace SMSClasses
{
    /// <summary>
    /// Classe responsável por escrever os comandos AT na classe Modem.cs
    /// </summary>
      public class Commands
      {

            // Lista de Comandos
            const char CTR_Z = (char)26;
            const string AT_RESPONSE_OK = @"OK\r?\n";
            const string AT_RESPONSE_LIST_ALL =
@"\+CMGL:\s(?<position>\d+),""(?<type>(?:REC\sREAD|REC\sUNREAD|STO\sSENT|ST
O\sUNSENT))"",""?(?<phone>[\d\+]+)""?,(?<unk>[\d]*),""?(?<date>[\d/:,+-
]*)""?\r?\n(?<text>.*)\n";


        const string AT_RESPONSE_CMGS = @"\+CMGS:\s(?<position>\d+)\r?\n";


            protected Modem _modem;

            public Commands(Modem modem)
            {
                  this._modem = modem;
                  this._modem.Open();
            }

            public void Dispose()
            {
                  this._modem.Close();
            this._modem = null;
            }

        //Checa se a conexão com o modem está estabelecida através do envio
de um comando AT.
            public string cmdConnectionTest()
            {
            this._modem.Console.Write("INICIO: Comando AT");

                  MatchCollection mc;

                  mc = this._modem.SendDataAndWaitResponse("AT",
AT_RESPONSE_OK);

            return mc[0].Value;
            }

        //Método que consulta o modem para verificar se tem mensagens novas
ou não através do envio de comandos AT.
            public MatchCollection cmdGetSMSList()
            {
            this._modem.Console.Write("INICIO: Comando AT+CMGL");

                  MatchCollection mc;
```

```
                  mc = this._modem.SendDataAndWaitResponse("AT",
AT_RESPONSE_OK);
            mc = this._modem.SendDataAndWaitResponse("AT+CMGF=1",
AT_RESPONSE_OK);
                  mc =
this._modem.SendDataAndWaitResponse("AT+CMGL=\"ALL\"",
AT_RESPONSE_LIST_ALL);
            //Lista as mensagens novas encontradas e o respectivo texto
recebido pelo modem.
            foreach (Match m in mc)
                  {

      this._modem.Console.Write(m.Groups["position"].Value + "=" +
m.Groups["text"]);
                  }

                  return mc;
            }
        //Utiliza o id para apagar uma mensagem da lista de memória do
modem através do envio de um comando AT.
            public bool cmdApagaMensagem(string id)
            {
            this._modem.Console.Write("Apaga " + id);

                  MatchCollection mc;

            mc = this._modem.SendDataAndWaitResponse("AT", AT_RESPONSE_OK);
                  mc = this._modem.SendDataAndWaitResponse("AT+CMGD=" + id,
AT_RESPONSE_OK);

                  return true;
            }
            //Envia um SMS ou a resposta do sistema para o usuário através
do envio de comando AT em duas partes.
            public string cmdEnviaSMS(string telefone, string sms)
            {
                  this._modem.Console.Write("Envia SMS para " + telefone +
" mensagem " + sms);
            //Define 160 caracteres como o valor máximo permitido, caso
contrário não será feito registro.
                  if (sms.Length > 160)
                  {
                        sms = sms.Substring(0, 160);
                  }
            //A primeira parte solicita o envio da mensagem, aguarda sinal
de cofirmação, define o texto para então completar o comando com CTRL+Z.
                  MatchCollection mc;
                  string retorno = "";

                  mc = this._modem.SendDataAndWaitResponse("AT",
AT_RESPONSE_OK);
                  mc = this._modem.SendDataAndWaitResponse("AT+CMGF=1",
AT_RESPONSE_OK);
                  this._modem.SendData("AT+CMGS=\"" + telefone + "\"",
"\r");
            retorno = this._modem.ReceiveData();  // Ler o prompt de
resposta. Pode ignorar o valor
            this._modem.SendData(sms, "" + CTR_Z);
            retorno = this._modem.ReceiveData();

                  int condSaida = 0;
```

```
                  while ( (retorno == "") && (condSaida++ < 25) ) // Se não
confirmar o envio, espera por 5 seg... Depois dá o erro.
                  {
                 System.Threading.Thread.Sleep(200);
                        retorno = this._modem.ReceiveData();
                  }

                  Regex regex = new Regex(AT_RESPONSE_CMGS);
            if (regex.IsMatch(retorno))
                  {
                        Match m = regex.Match(retorno);
                        retorno = "+" + m.Groups["position"].Value;
                  }

                  return retorno;
            }
        //Consulta o saldo disponível através do envio de um comando AT.
            public string cmdVerificarSaldo()
            {
                  return this.cmdEnviaSMS("222", "SAL");
            }


      }
}
```

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Windows.Forms;

namespace SMSClasses
{
    ///<summary>
    ///Classe que contém os comandos necessários para escrever na tela do
formulário a saída do modem.
    ///</summary>
    public class ConsoleWriter : IConsoleWriter
      {
            protected TextBox _textBox;
            public ConsoleWriter(TextBox txt)
            {
            this._textBox = txt;
            }

            public void Write(string message)
            {
                  this.Write(message, false);
            }

            public void Write(string message, bool error)
            {
                  if (error)
                  {
                        message = "*** ERRO *** ===> " + message;
                  }
                  this._textBox.AppendText(message + (message.IndexOf("\n")
< 0 ? "\r\n" : ""));
            }
      }
}
```

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Data.SqlClient;
using System.Data;

namespace SMSClasses
{
      /// <summary>
      /// Classe responsável pelo armazenamento dos dados da aplicação.
      /// </summary>
    public class Database
      {
            public const string CONEXAO = "Integrated Security=SSPI;Persist
Security Info=False;Initial Catalog=smscinema;Data Source=localhost";

            protected SqlConnection GetConnection()
            {
            SqlConnection conn = new SqlConnection(CONEXAO);
                  return conn;
            }
        //Checa conexão com banco de dados.
            protected void ExecuteSQL(string sql)
            {
            SqlConnection conn = this.GetConnection();
            SqlCommand command = new SqlCommand(sql, conn);
                  conn.Open();
                  try
                  {
                command.ExecuteNonQuery();

                  }
                  finally
                  {
                        conn.Close();
                  }
            }

            protected DataSet QuerySQL(string sql)
            {
            SqlDataAdapter adapter = new SqlDataAdapter(sql,
this.GetConnection());
            DataSet ds = new DataSet();
            adapter.Fill(ds);

                  return ds;
            }

            /// <summary>
            ///
            /// </summary>
            /// <param name="celular"></param>
            /// <param name="data"></param>
            /// <param name="messagetext"></param>
            /// <param name="response"></param>
```

```
        //Executa o SQL e espera como resultado uma coleção de dados para a
tabela smslog.
        public int RegistrarLog(string celular, string data, string
messagetext, string response)
            {

                  string sql = "insert into smslog (celular, data,
messagetext, response) values ( " +
                        "'" + celular + "', '" + data + "', '" +
messagetext + "', '" + response + "'); select @@identity id";

            DataSet ds = this.QuerySQL(sql);
                  return
Convert.ToInt32(ds.Tables[0].Rows[0]["id"].ToString());
            }
        //Atualiza o banco com dados novos/modificados da tabela.
            public void AtualizaLog(int idlog, string mensagem)
            {
                  string sql = "update smslog set modem_response = '" +
mensagem + "' where idsms = " + idlog.ToString();
                  this.ExecuteSQL(sql);
            }
        //Executa o SQL para fazer o insert na tabela registro_compra.
            public void RegistrarCompra(int idsms, int idassento, int qtd)
            {

            string sql = "insert into registro_compra (idsms, idassento,
qtd) values ( " +
                        idsms.ToString() + ", " + idassento.ToString() + ",
" + qtd.ToString() + ");";
            this.ExecuteSQL(sql);
            }
        //Conta a quantidade de assentos disponíveis para o sistema de
compra de ingresso usando SMS.
            public int ReservarAssentos(string idfilme, string hora, string
minuto, string qtd)
            {
                  try
                  {
                 if (Convert.ToInt32(qtd) <= 0)
                        {
                              return -2;
                        }
                  }
                  catch
                  {
                        return -3;
                  }

                  string sql = "select idassento, idfilme, assentos_totais,
assentos_disponiveis from assento ";

                  string filtro = " where idfilme = " + idfilme;
                  filtro += "   and hora = " + hora;
                  filtro += "   and minuto = " + minuto;
                  filtro += "   and assentos_disponiveis >= " + qtd;
                  // Descomente para validar também o dia de hoje!!
                  //filtro += " and year(data) = year(getdate()) and
month(data) = month(getdate()) and day(data) = day(getdate()) ";
```

```
            DataSet ds = this.QuerySQL(sql + filtro);

                  if (ds.Tables[0].Rows.Count == 0)
                  {
                        // Não encontrou assentos disponiveis
                        return -1;
                  }
                  else
                  {
                        // Encontrou assentos, reserva e retorna para a
funcao chamadora.
                sql = " update assento set assentos_disponiveis =
assentos_disponiveis - " + qtd;
                        this.ExecuteSQL(sql + filtro);
                        return
Convert.ToInt32(ds.Tables[0].Rows[0]["idassento"].ToString());
            }
            }
      }
}
```

```
using System;
using System.Collections.Generic;
using System.Text;

namespace SMSClasses
{
      /// <summary>
    /// Essa classe define que toda classe que implementar essa estrutura é
obrigada a implementar todos os métodos existentes nessa interface.
      /// </summary>
    public interface IConsoleWriter
      {
            void Write(string message);
            void Write(string message, bool error);
      }
}
```

```
using System.IO.Ports;
using System.Text.RegularExpressions;
using System.Diagnostics;
using System;

namespace SMSClasses
{
      /// <summary>
      /// Classe que conecta, envia e recebe dados do modem G24.
      /// </summary>
    public class Modem
      {
        //Mostra em qual porta o modem está conectado
        const string PORT = "COM9";

            protected SerialPort modemPort;

            public IConsoleWriter Console { get; set; }

            public Modem()
            {
                  this.Console = new NullWriter();
            }

            public Modem(IConsoleWriter console)
            {
                  this.Console = console;
            }
        //Checa a conexão via COMMPORT com modem G24 através do envio de um
comando AT.
            public void Open()
        {
                  if (this.modemPort != null && this.modemPort.IsOpen)
                  {
                        return;
                  }

            try
            {
                        this.modemPort = new SerialPort(PORT, 115200,
Parity.None, 8, StopBits.One);

                        this.modemPort.WriteTimeout = 5000;
                  this.modemPort.ReadTimeout = 5000;
                  this.modemPort.DtrEnable = true;
                  this.modemPort.RtsEnable = true;
                        this.modemPort.NewLine =
System.Environment.NewLine;
                        this.modemPort.Handshake = Handshake.RequestToSend;
                        this.modemPort.Open();
                SendData("ATE0");
                ReceiveData();
            }
            catch (Exception ex)
            {
                        this.Console.Write("Não foi possível conectar ao
MODEM. " + ex.Message, true);
            }
        }
```

```
        public void Close()
        {
            if (this.modemPort.IsOpen)
                this.modemPort.Close();
        }

        public void SendData(string message)
        {
            this.SendData(message, System.Environment.NewLine);
        }
        //Envia comandos AT e espera receber uma resposta do modem.
            public void SendData(string message, string NewLine)
            {
                  if (!this.modemPort.IsOpen)
                        this.Console.Write("Porta não está aberta.", true);

                  try
                  {
                        byte[] buffer =
System.Text.Encoding.ASCII.GetBytes(message + NewLine);
                        modemPort.Write(buffer, 0, buffer.Length);
                        //modemPort.WriteLine(message);
                  }
                  catch (Exception ex)
                  {
                        throw;
                  }
            }

            public string ReceiveData()
        {
            if (!this.modemPort.IsOpen)
                        this.Console.Write("Porta não está aberta.", true);
            //Consulta o modem a cada 15 segundos para verificar se há ou
não novas mensagens
            try
            {
                System.Text.StringBuilder result = new
System.Text.StringBuilder();

                if (modemPort.BytesToRead == 0)
                {
                    int retries = 0;
                    while (retries < 5)
                    {
                        System.Threading.Thread.Sleep(150);
                        if (modemPort.BytesToRead > 0) break;
                        retries++;
                    }
                }

                while (modemPort.BytesToRead > 0)
                {
                    byte[] buffer = new byte[modemPort.BytesToRead];
                    int numRead = modemPort.Read(buffer, 0, buffer.Length);
                    if (numRead != 0)

result.Append(System.Text.Encoding.ASCII.GetString(buffer, 0, numRead));
```

```
                    else
                        break;
                    System.Threading.Thread.Sleep(150);
                }
                return result.ToString();
            }
            catch (TimeoutException)
            {
                return "";
            }
            catch (Exception ex)
            {
                    this.Console.Write(ex.Message, true);
            }

                return "";
        }

            public MatchCollection SendDataAndWaitResponse(string data,
string expected)
            {
                return this.SendDataAndWaitResponse(data, expected,
System.Environment.NewLine);
            }
        //Método que mostra na tela do formulário a resposta do modem.
            public MatchCollection SendDataAndWaitResponse(string data,
string expected, string NewLine)
            {
                this.SendData(data, NewLine);
            string response = this.ReceiveData();

                Regex regex = new Regex(expected);
            //Faz uso da classe regex para conferir a resposta do sistema e
retornar a mensagem para usuário.
            if (regex.IsMatch(response))
                {
                return regex.Matches(response);
                }
                else
                {
                    //return null;
                    throw new Exception("AT_ERROR: " + response);
                    return null;
                }
            }
      }
}
```

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Diagnostics;

namespace SMSClasses
{
    ///<summary>
    ///Classe implementa a interface IConsoleWriter e descarta as mensagens
recebidas pelo modem.
    ///</summary>
    public class NullWriter : IConsoleWriter
      {
            public void Write(string message)
            {
                  this.Write(message, false);
            }

            public void Write(string message, bool error)
            {
                  if (error)
                  {
                        throw new Exception(message);
                  }
                  else
                  {
                Debug.Print(message);
                  }
            }
      }
}
```

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Text.RegularExpressions;

namespace SMSClasses
{
      public struct RetornoProtocolo
      {
            public int idlog;
            public string mensagem;
      }
    ///<summary>
    ///Classe que define os tipos de resposta do sistema com base no
protocolo SMS para ser enviado o retorno para o usuário.
    ///</summary>
      public class Protocol
      {
            protected const string RESP_SUCESSO = "COMPRA EFETUADA";
            protected const string RESP_INDISPONIVEL = "INDISPONIVEL";
            protected const string RESP_DESCONHECIDO = "COMANDO
DESCONHECIDO";
            protected const string RESP_ERRO = "ERRO NA OPERACAO";
        const string CMD_COMPRAR =
@"comprar\s(?<sessao>(?<filme>\d?\d)\s(?<hora>[012]\d)(?<minuto>[0-
5]\d))\s(?<qtd>\d?\d)";
        const string VALIDA_TELEFONE =
@"(?<telefone>(?<ddd>0\d{2})(?<numero>\d{8}))";


            public RetornoProtocolo Processa(string position, string type,
string phone, string text, string date)
            {

                  Regex regexTelefone = new Regex(VALIDA_TELEFONE);
                  Regex regexCmdComprar = new Regex(CMD_COMPRAR);

                  Database db = new Database();

            RetornoProtocolo retStruct = new RetornoProtocolo();

            //Deixa todos os textos todos iguais.
                  text = text.ToLower();

                  try
                  {
            // Descarta o processamento se o telefone for invalido, pois
nao tem como devolver a resposta mas faz o registro na tabela smslog.
                        if (!regexTelefone.IsMatch(phone))
                        {
                              retStruct.idlog = db.RegistrarLog(phone, date,
text, "");
                    retStruct.mensagem = null;
                              return retStruct;
                        }
```

```
            // Se a mensagem nao for reconhecida, devolve um erro.
                        if (!regexCmdComprar.IsMatch(text))
                        {
                              retStruct.idlog = db.RegistrarLog(phone, date,
text, RESP_DESCONHECIDO);
                              retStruct.mensagem = RESP_DESCONHECIDO;
                              return retStruct;
                        }

                        Match m = regexCmdComprar.Match(text);

                        // A mensagem aparentemente é valida, tentar
reservar o assento
                        int retorno =
db.ReservarAssentos(m.Groups["filme"].Value, m.Groups["hora"].Value,
m.Groups["minuto"].Value, m.Groups["qtd"].Value);
                        if (retorno <= 0)
                        {
                              retStruct.idlog = db.RegistrarLog(phone, date,
text, RESP_INDISPONIVEL);
                              retStruct.mensagem = RESP_INDISPONIVEL;
                              return retStruct;
                        }
                        else
                        {
                              int idsms = db.RegistrarLog(phone, date, text,
RESP_SUCESSO);
                    db.RegistrarCompra(idsms, retorno,
Convert.ToInt32(m.Groups["qtd"].Value));
                              retStruct.idlog = idsms;
                    retStruct.mensagem = RESP_SUCESSO;
                              return retStruct;
                        }
                  }
                  catch
                  {
                        retStruct.idlog = db.RegistrarLog(phone, date,
text, RESP_ERRO);
                        retStruct.mensagem = RESP_ERRO;
                        return retStruct;
                  }

                  return retStruct;
            }

      }
}
```

APÊNDICE II – Códigos de Programação - Solução ADM Cinema

Adm Cinema é a solução que administra as tabelas do banco de dados e é um projeto WEB baseado no framework XMLNuke.

```
using System;
using System.Collections.Generic;
using System.Text;

// Xmlnuke
using com.xmlnuke.admin;
using com.xmlnuke.anydataset;
using com.xmlnuke.classes;
using com.xmlnuke.database;
using com.xmlnuke.db;
using com.xmlnuke.engine;
using com.xmlnuke.exceptions;
using com.xmlnuke.international;
using com.xmlnuke.module;
using com.xmlnuke.processor;
using com.xmlnuke.util;
using System.Collections.Specialized;
using System.Collections;

namespace AdmCinema
{
    ///<summary>
    ///Classe que define as funcionalidades para exibir o cadastro de filme
na tela.
    ///</summary>
      public class CadastroFilme : BaseModule
      {
        ///<summary>
        ///Método que cria a página para opção "Cadastrar Filmes e Assentos
disponíveis" e mostra em uma tabela os códigos e os nomes dos filmes
cadastrados no sistema.
        ///</summary>>
        public override IXmlnukeDocument CreatePage()
            {
                  // Load default Language Collection
                  this._myWords = this.WordCollection();

                  // Create a XMLNuke document
                  this.defaultXmlnukeDocument = new
XmlnukeDocument("Administração SMS Cinema - Cadastro Filme", "Administração
SMS Cinema - Cadastro Filme");

                  XmlBlockCollection block = new
XmlBlockCollection("Administração SMS Cinema", BlockPosition.Center);
                  XmlParagraphCollection paragraph = new
XmlParagraphCollection();
                  block.addXmlnukeObject(paragraph);
                  this.defaultXmlnukeDocument.addXmlnukeObject(block);

                  DBDataSet db = new DBDataSet("conexao", this._context);
            ProcessPageFields fields = new ProcessPageFields();
                  ProcessPageField field;
```

```
                  if ( (this._action != "editassento") &&
(this._context.ContextValue("editing") == "") )
                  {
                        // Create a Block, a Paragraph and a Text
                  block.setTitle("Administração SMS Cinema - Editar Filme");

                        field = ProcessPageFields.Factory("idfilme", "Cód
Filme", 0, true, true);
                        field.key = true;
                        field.editable = false;
                        fields.addProcessPageField(field);

                        field = ProcessPageFields.Factory("filme", "Nome do
Filme", 30, true, true);
                  field.maxLength = 30;
                        fields.addProcessPageField(field);

                        CustomButtons cb = new CustomButtons();
                        cb.icon = "common/editlist/ic_subcategorias.gif";
                  cb.alternateText = "Editar Assentos Disponíveis";
                        cb.action = "editassento";

                        ProcessPageStateDB process = new
ProcessPageStateDB(this._context, fields, "Cadastro de Filme",
"module:admcinema.cadastrofilme", new CustomButtons[] { cb }, "filme",
"conexao");

                        paragraph.addXmlnukeObject(process);
                  }
                  else
                  {
                  //Obtém o "idfilme", se disponível.
                  string idfilme = this._context.ContextValue("idfilmeedit");
                        if (String.IsNullOrEmpty(idfilme))
                        {
                              idfilme =
this._context.ContextValue("valueid");
                        }

                  block.setTitle("Administração SMS Cinema - Editar
Assentos");

                  field = ProcessPageFields.Factory("idassento", "Cód
Assento", 0, true, true);
                  field.key = true;
                        field.editable = false;
                        fields.addProcessPageField(field);

                        field = ProcessPageFields.Factory("data", "Data",
10, true, true);
                  field.dataType = INPUTTYPE.DATE;
                        field.maxLength = 10;
                        fields.addProcessPageField(field);

                        field = ProcessPageFields.Factory("idfilme", "Nome
do Filme", 30, true, true);
                        IIterator it = db.getIterator("select idfilme,
filme from filme");
                        NameValueCollection filmeArray =
BaseDBAccess.getArrayFromIterator(it, "idfilme", "filme");
```

```
                        field.fieldXmlInput =
XmlInputObjectType.SELECTLIST;
                        field.arraySelectList = filmeArray;
                        fields.addProcessPageField(field);

                        field = ProcessPageFields.Factory("hora", "Hora",
2, true, true);
                        field.dataType = INPUTTYPE.NUMBER;
                        field.fieldXmlInput =
XmlInputObjectType.SELECTLIST;
                        NameValueCollection horaArray = new
NameValueCollection();
                        horaArray.Add("10", "10");
                        horaArray.Add("11", "11");
                        horaArray.Add("12", "12");
                        horaArray.Add("13", "13");
                        horaArray.Add("14", "14");
                        horaArray.Add("15", "15");
                        horaArray.Add("16", "16");
                        horaArray.Add("17", "17");
                        horaArray.Add("18", "18");
                        horaArray.Add("19", "19");
                        horaArray.Add("20", "20");
                        horaArray.Add("21", "21");
                        horaArray.Add("22", "22");
                        horaArray.Add("23", "23");
                        field.arraySelectList = horaArray;
                        field.maxLength = 2;
                        fields.addProcessPageField(field);

                        field = ProcessPageFields.Factory("minuto",
"Minuto", 2, true, true);
                        field.dataType = INPUTTYPE.NUMBER;
                field.fieldXmlInput = XmlInputObjectType.SELECTLIST;
                        NameValueCollection minutoArray = new
NameValueCollection();
                        minutoArray.Add("0", "00");
                        minutoArray.Add("15", "15");
                        minutoArray.Add("30", "30");
                        minutoArray.Add("45", "45");
                        field.arraySelectList = minutoArray;
                        field.maxLength = 2;
                        fields.addProcessPageField(field);

                        field =
ProcessPageFields.Factory("assentos_totais", "Assentos Totais", 4, true,
true);
                field.dataType = INPUTTYPE.NUMBER;
                        field.maxLength = 3;
                        fields.addProcessPageField(field);

                        field =
ProcessPageFields.Factory("assentos_disponiveis", "Assentos Disp.", 4,
true, true);
                field.dataType = INPUTTYPE.NUMBER;
                field.maxLength = 3;
                fields.addProcessPageField(field);
```

```
                        ProcessPageStateDB process = new
ProcessPageStateDB(this._context, fields, "Cadastro de Assentos",
"module:admcinema.cadastrofilme", null, "assento", "conexao");
                        process.setFilter( " idfilme = " + idfilme);
                        process.addParameter("idfilmeedit", idfilme);
                        process.addParameter("editing", "assento");

                        paragraph.addXmlnukeObject(process);
                }


                return this.defaultXmlnukeDocument;
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Text;

// Xmlnuke
using com.xmlnuke.admin;
using com.xmlnuke.anydataset;
using com.xmlnuke.classes;
using com.xmlnuke.database;
using com.xmlnuke.db;
using com.xmlnuke.engine;
using com.xmlnuke.exceptions;
using com.xmlnuke.international;
using com.xmlnuke.module;
using com.xmlnuke.processor;
using com.xmlnuke.util;
using System.Collections.Specialized;
using System.Collections;

namespace AdmCinema
{
    ///<summary>
    ///Classe que define a página principal da interface WEB.
    ///</summary>
    public class Home : BaseModule
      {
        ///<summary>
        ///Método que cria a página inicial da interface WEB AdmCinema e o
menu com três opções disponíveis do sistema.
        ///</summary>
        public override IXmlnukeDocument CreatePage()
            {
                  // Load default Language Collection
                  this._myWords = this.WordCollection();

                  // Create a XMLNuke document
                  this.defaultXmlnukeDocument = new
XmlnukeDocument("Administração SMS Cinema", "Administração SMS Cinema");

                  // Create a Block, a Paragraph and a Text
            XmlBlockCollection block = new
XmlBlockCollection("Administração SMS Cinema", BlockPosition.Center);
                  XmlParagraphCollection paragraph = new
XmlParagraphCollection();

            XmlListCollection list = new
XmlListCollection(XmlListType.UnorderedList, "Selecione a opção desejada");
            NameValueCollection items = new NameValueCollection();
            items.Add("module:admcinema.cadastrofilme", "Cadastrar Filmes e
Assentos disponíveis");
            items.Add("module:admcinema.registrocompra", "Consultar
Registro de Compra");
            items.Add("module:admcinema.logsms", "Ver Registro do SMS");

            foreach (string key in items)
                  {
                        XmlAnchorCollection href = new
XmlAnchorCollection(key);
                        href.addXmlnukeObject(new XmlnukeText(items[key]));
                        list.addXmlnukeObject(href);
                  }
```

```
                    paragraph.addXmlnukeObject(list);
                    block.addXmlnukeObject(paragraph);
                    this.defaultXmlnukeDocument.addXmlnukeObject(block);

                    return this.defaultXmlnukeDocument;
                }
            }
        }
```

```
using System;
using System.Collections.Generic;
using System.Text;

// Xmlnuke
using com.xmlnuke.admin;
using com.xmlnuke.anydataset;
using com.xmlnuke.classes;
using com.xmlnuke.database;
using com.xmlnuke.db;
using com.xmlnuke.engine;
using com.xmlnuke.exceptions;
using com.xmlnuke.international;
using com.xmlnuke.module;
using com.xmlnuke.processor;
using com.xmlnuke.util;

namespace AdmCinema
{
    /// <summary>
    /// Classe que define a página para ver o Log de Mensagens.
    /// </summary>
    public class LogSMS : BaseModule
      {
        /// <summary>
        /// Método que cria a página para opção "Ver Registro do SMS" da
página inicial e mostra os registros da tabela dbo.smslog.
        /// </summary>
        public override IXmlnukeDocument CreatePage()
            {
            // Load default Language Collection
                  this._myWords = this.WordCollection();
            // Create a XMLNuke document
                  this.defaultXmlnukeDocument = new
XmlnukeDocument("Administração SMS Cinema - Log de Mensagens",
this._myWords.Value("ABSTRACT"));

                  // Create a Block, a Paragraph and a Text
            XmlBlockCollection block = new
XmlBlockCollection("Administração SMS Cinema - Log de Mensagens",
BlockPosition.Center);
                  XmlParagraphCollection paragraph = new
XmlParagraphCollection();
                  block.addXmlnukeObject(paragraph);
                  this.defaultXmlnukeDocument.addXmlnukeObject(block);

                  DBDataSet db = new DBDataSet("conexao", this._context);
                  string sql = "select * from smslog";
            IIterator it = db.getIterator(sql);
            XmlEditList editlist = new XmlEditList(this._context, "Dados de
Compra", "module:admcinema.logsms");
                  editlist.setDataSource(it);
                  editlist.setEnablePage(true);
                  editlist.setPageSize(150, 0);
                  block.addXmlnukeObject(editlist);
                  //Debug.Print(it);
                  //
                  return this.defaultXmlnukeDocument;
            }
      }
}
```

```
using System;
using System.Collections.Generic;
using System.Text;

// Xmlnuke
using com.xmlnuke.admin;
using com.xmlnuke.anydataset;
using com.xmlnuke.classes;
using com.xmlnuke.database;
using com.xmlnuke.db;
using com.xmlnuke.engine;
using com.xmlnuke.exceptions;
using com.xmlnuke.international;
using com.xmlnuke.module;
using com.xmlnuke.processor;
using com.xmlnuke.util;

namespace AdmCinema
{
    /// <summary>
    /// Classe que recebe novo registro e exibe o resultado em uma tabela.
    /// </summary>
    public class NewModule : BaseModule
    {
        /// <summary>
        ///Método que cria a página para cadastrar novo filme e insere o
registro na tabela dbo.filme.
        /// </summary>

        public override IXmlnukeDocument CreatePage()
        {
                    // Load default Language Collection
                    this._myWords = this.WordCollection();

                    // Create a XMLNuke document
                this.defaultXmlnukeDocument = new
XmlnukeDocument(this._myWords.Value("TITLE"),
this._myWords.Value("ABSTRACT"));
                    // Create a Block, a Paragraph and a Text
                    XmlBlockCollection block = new
XmlBlockCollection(this._myWords.Value("TITLE"), BlockPosition.Center);
                    XmlParagraphCollection paragraph = new
XmlParagraphCollection();
                    paragraph.addXmlnukeObject(new
XmlnukeText(this._myWords.Value("FIRST_TEXT")));
                    block.addXmlnukeObject(paragraph);
                    this.defaultXmlnukeDocument.addXmlnukeObject(block);
                    DBDataSet db = new DBDataSet("conexao", this._context);
                IIterator it = db.getIterator("select * from filme");
                XmlEditList editlist = new XmlEditList(this._context, "Teste",
"teste");
                    editlist.setDataSource(it);
                    block.addXmlnukeObject(editlist);
                    //Debug.Print(it);
                    //
                return this.defaultXmlnukeDocument;
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Text;

// Xmlnuke
using com.xmlnuke.admin;
using com.xmlnuke.anydataset;
using com.xmlnuke.classes;
using com.xmlnuke.database;
using com.xmlnuke.db;
using com.xmlnuke.engine;
using com.xmlnuke.exceptions;
using com.xmlnuke.international;
using com.xmlnuke.module;
using com.xmlnuke.processor;
using com.xmlnuke.util;

namespace AdmCinema
{
    /// <summary>
    /// Classe que exibe em uma tabela todas as compras efetuadas com
sucesso.
    /// </summary>

    public class RegistroCompra : BaseModule
      {
        ///<summary>
        ///Método que cria a página para opção "Registro de Compra" da
página inicial e mostra os registros da tabela dbo.registro_compra.
        ///</summary>
        public override IXmlnukeDocument CreatePage()
            {
                  // Load default Language Collection
                  this._myWords = this.WordCollection();

            // Create a XMLNuke document
                  this.defaultXmlnukeDocument = new
XmlnukeDocument("Administração SMS Cinema - Registro de Compra",
this._myWords.Value("ABSTRACT"));

            // Create a Block, a Paragraph and a Text
            XmlBlockCollection block = new
XmlBlockCollection("Administração SMS Cinema - Registro de Compra",
BlockPosition.Center);
                  XmlParagraphCollection paragraph = new
XmlParagraphCollection();
                  block.addXmlnukeObject(paragraph);
                  this.defaultXmlnukeDocument.addXmlnukeObject(block);

                  DBDataSet db = new DBDataSet("conexao", this._context);
                  string sql = "select s.data, s.celular, s.messagetext,
s.response, f.filme, a.hora, a.minuto, rc.qtd " +
                               " from registro_compra rc inner join assento a
on rc.idassento = a.idassento " +
                               " inner join filme f on a.idfilme = f.idfilme"
+
                    " inner join smslog s on rc.idsms = s.idsms";
            IIterator it = db.getIterator(sql);
                  XmlEditList editlist = new XmlEditList(this._context,
"Dados de Compra", "Dados de Compra");
                  editlist.setDataSource(it);
```

```
                block.addXmlnukeObject(editlist);
                //Debug.Print(it);

                //
                return this.defaultXmlnukeDocument;
            }
        }
}
```

APÊNDICE III – SOLUÇÃO SQL

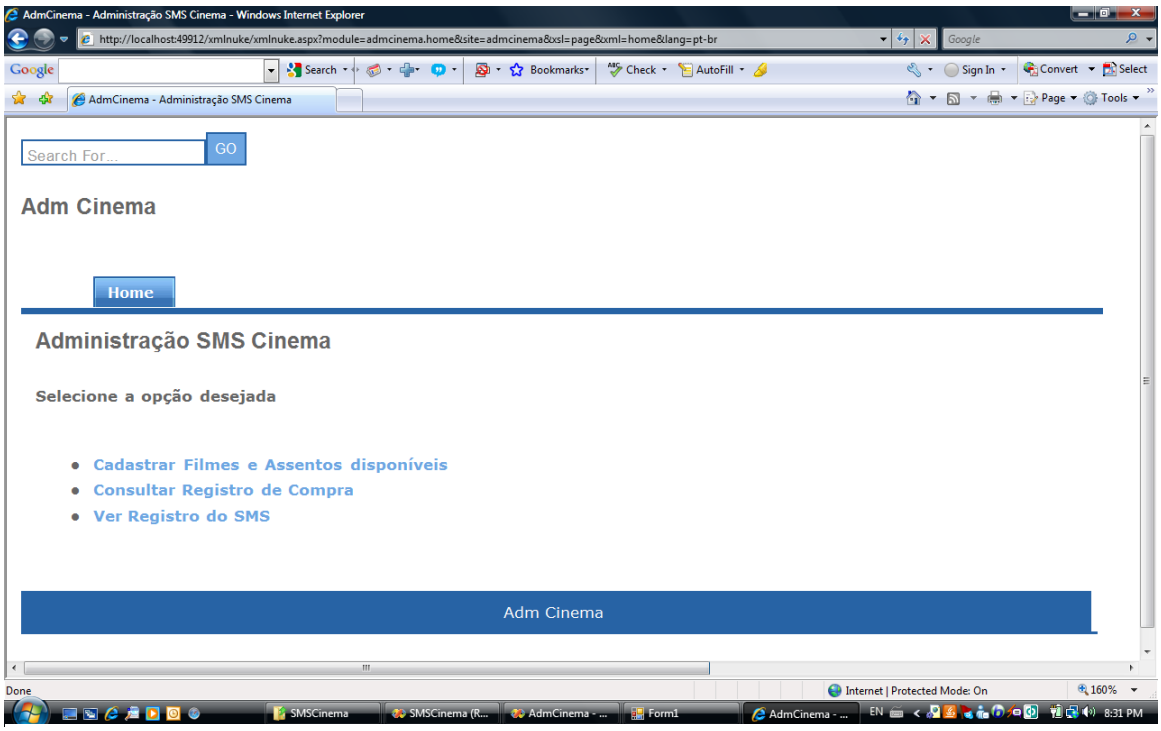
Este script contém os comandos SQL para se criar o banco de dados do SMSCinema.

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[filme]') AND type in (N'U'))
BEGIN
/*
A tabela filme é responsável pelo cadastro de todos os filmes no sistema
*/
CREATE TABLE [dbo].[filme](
      [idfilme] [int] IDENTITY(1,1) NOT NULL,
      [filme] [varchar](30) NULL,
 CONSTRAINT [PK__filme__7E6CC920] PRIMARY KEY CLUSTERED
(
      [idfilme] ASC
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[smslog]') AND type in (N'U'))
BEGIN
/* Faz o registro de todas as operações que ocorrem no sistema */
CREATE TABLE [dbo].[smslog](
      [idsms] [int] IDENTITY(1,1) NOT NULL,
      [celular] [varchar](24) NULL,
      [data] [varchar](24) NULL,
      [messagetext] [varchar](170) NULL,
      [response] [varchar](170) NULL,
      [modem_response] [varchar](170) NULL,
 CONSTRAINT [PK__smslog__7C8480AE] PRIMARY KEY CLUSTERED
(
      [idsms] ASC
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[assento]') AND type in (N'U'))
BEGIN
/* A tabela assento é usada para atualizar a quantidade de ingressos
disponíveis segundo a fórmula definida no projeto */
CREATE TABLE [dbo].[assento](
      [idassento] [int] IDENTITY(1,1) NOT NULL,
```

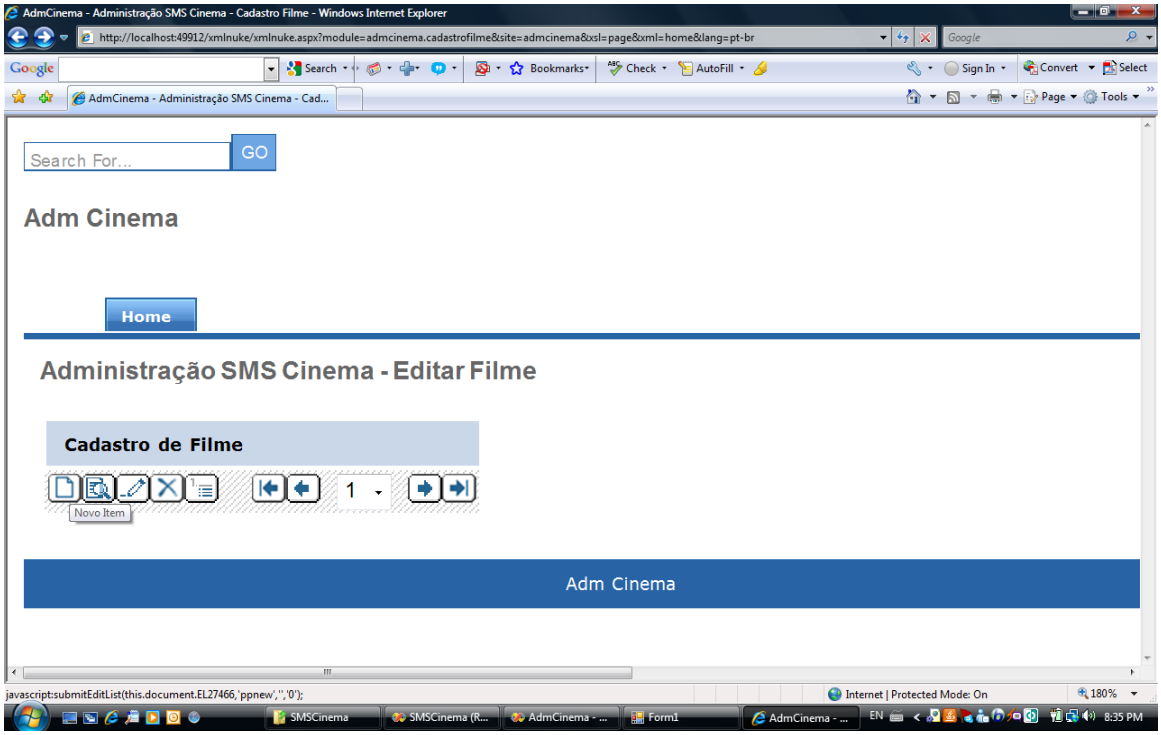
```
	[data] [datetime] NULL,
	[idfilme] [int] NULL,
	[hora] [int] NULL,
	[minuto] [int] NULL,
	[assentos_totais] [int] NOT NULL,
	[assentos_disponiveis] [int] NOT NULL,
 CONSTRAINT [PK__assento__00551192] PRIMARY KEY CLUSTERED
(
	[idassento] ASC
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY],
 CONSTRAINT [UQ__assento__014935CB] UNIQUE NONCLUSTERED
(
	[data] ASC,
	[idfilme] ASC,
	[hora] ASC,
	[minuto] ASC
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[registro_compra]') AND type in (N'U'))
BEGIN
/* Faz o registro de todas as compras efetuadas com sucesso do sistema */
CREATE TABLE [dbo].[registro_compra](
	[idsms] [int] NOT NULL,
	[idassento] [int] NOT NULL,
	[qtd] [int] NULL,
PRIMARY KEY CLUSTERED
(
	[idsms] ASC,
	[idassento] ASC
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
IF NOT EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK__assento__idfilme__023D5A04]') AND parent_object_id =
OBJECT_ID(N'[dbo].[assento]'))
ALTER TABLE [dbo].[assento]  WITH CHECK ADD  CONSTRAINT
[FK__assento__idfilme__023D5A04] FOREIGN KEY([idfilme])
REFERENCES [dbo].[filme] ([idfilme]) ON DELETE CASCADE
GO
IF NOT EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK__registro___idass__060DEAE8]') AND parent_object_id =
OBJECT_ID(N'[dbo].[registro_compra]'))
ALTER TABLE [dbo].[registro_compra]  WITH CHECK ADD  CONSTRAINT
[FK__registro___idass__060DEAE8] FOREIGN KEY([idassento])
REFERENCES [dbo].[assento] ([idassento]) ON DELETE CASCADE
GO
IF NOT EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK__registro___idsms__0519C6AF]') AND parent_object_id =
OBJECT_ID(N'[dbo].[registro_compra]'))
ALTER TABLE [dbo].[registro_compra]  WITH CHECK ADD  CONSTRAINT
[FK__registro___idsms__0519C6AF] FOREIGN KEY([idsms])
REFERENCES [dbo].[smslog] ([idsms]) ON DELETE CASCADE
```

APÊNDICE IV – Experimentos com a Interface WEB

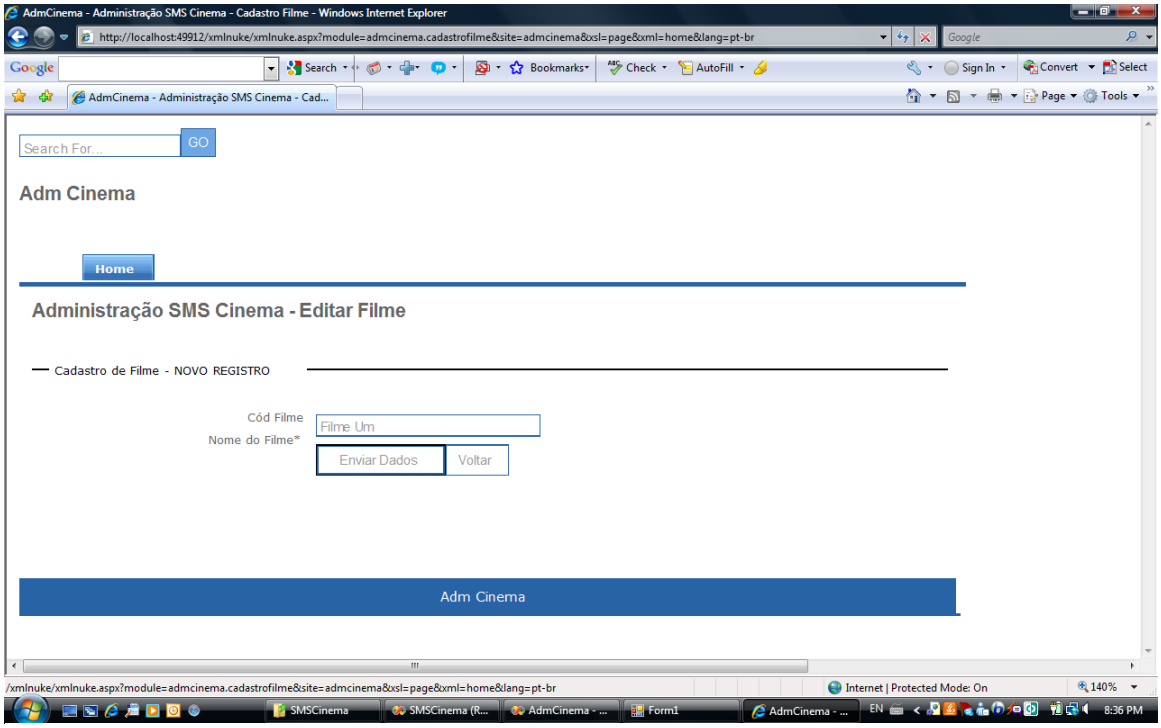
A seguir são expostas as fotos dos testes da solução ADM Cinema utilizando a interface WEB:



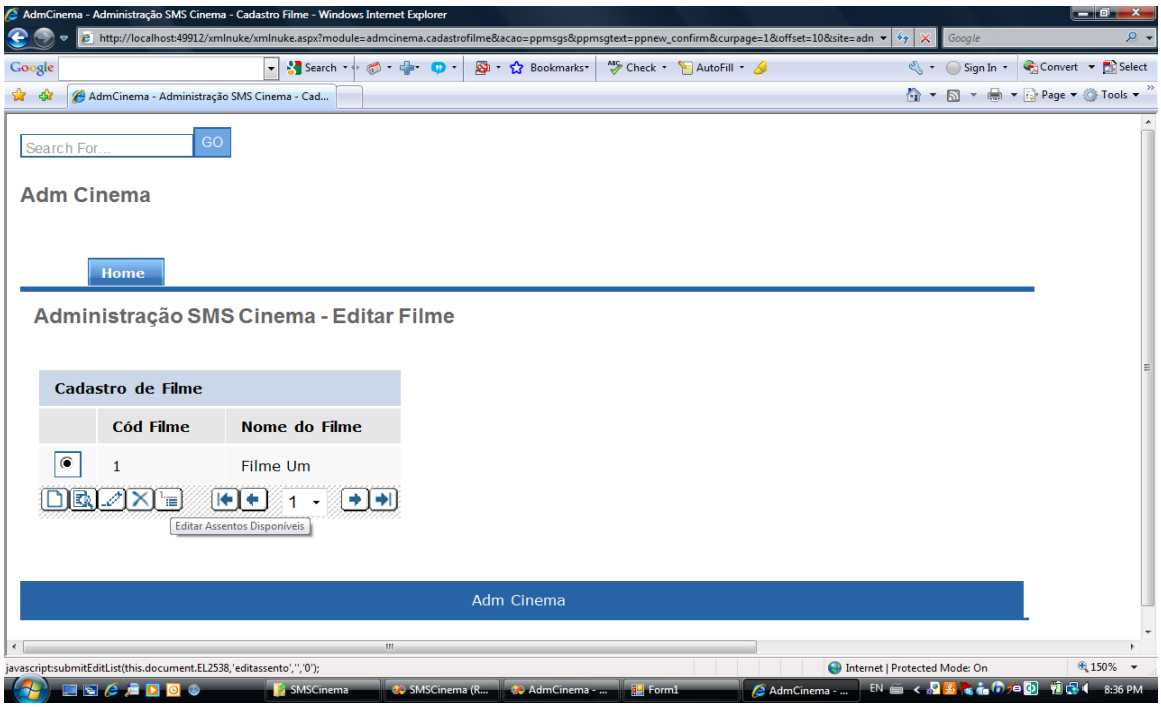
Página principal da interface.



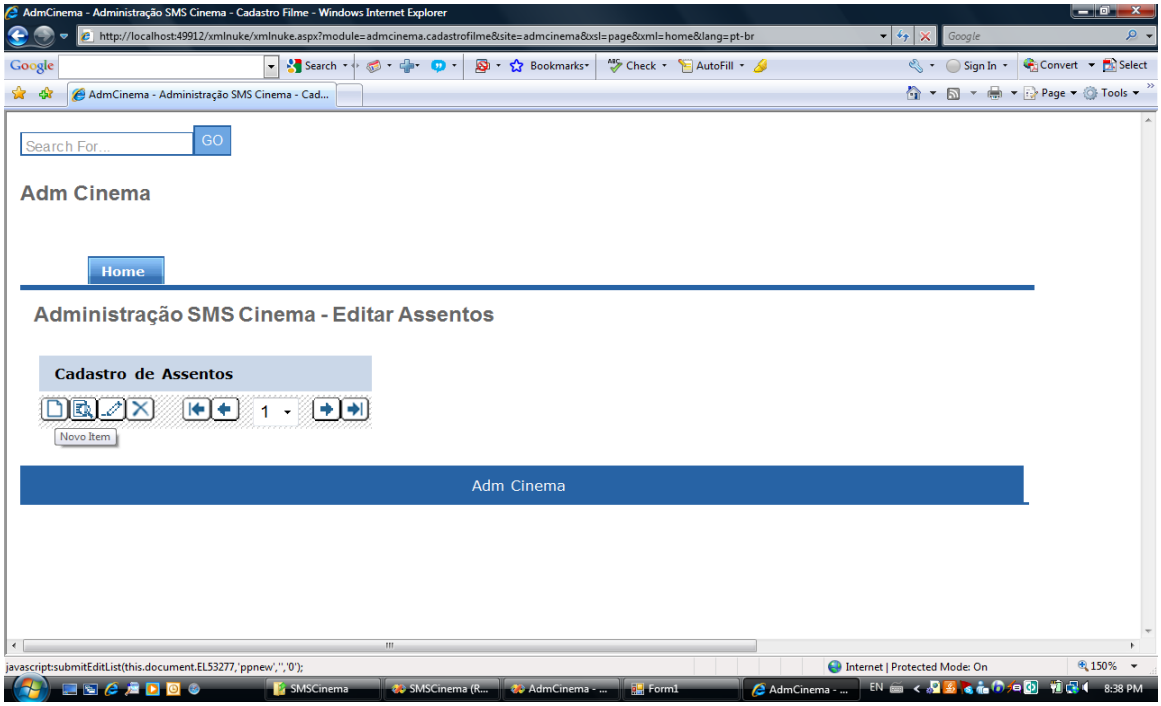
Para cadastrar novo filme: opção “Cadastrar Filmes e Assentos Disponíveis” – novo item.



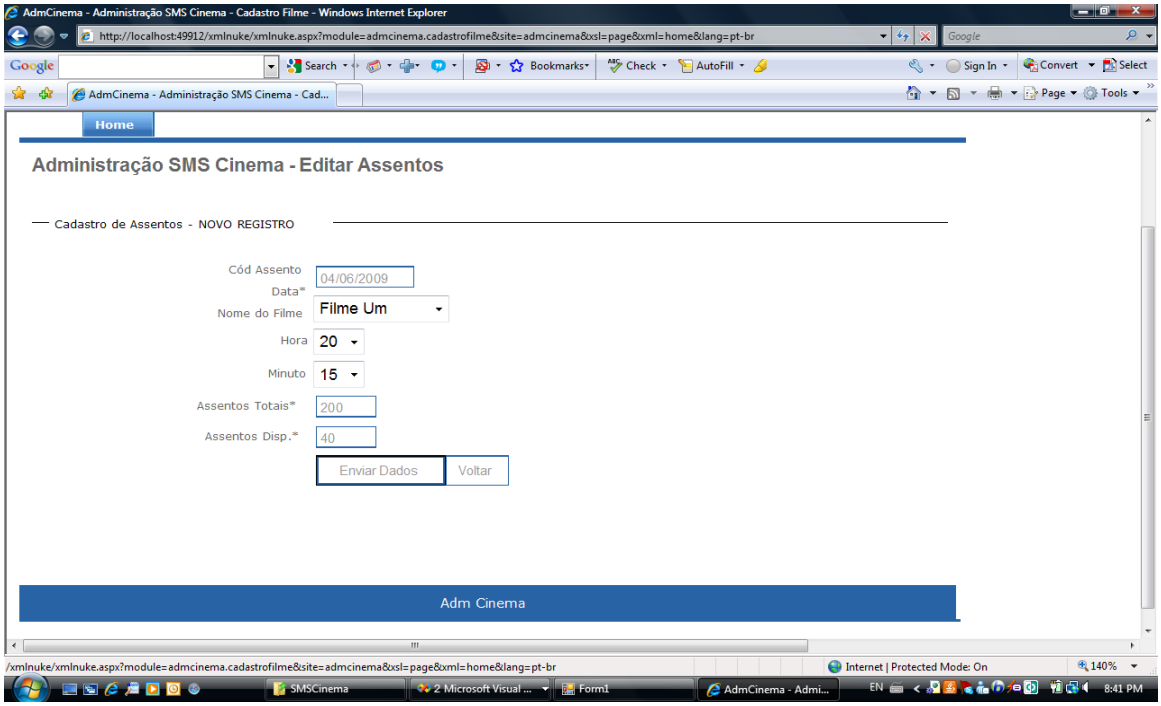
Inserido o nome do filme e enviado os dados para o banco de dados.



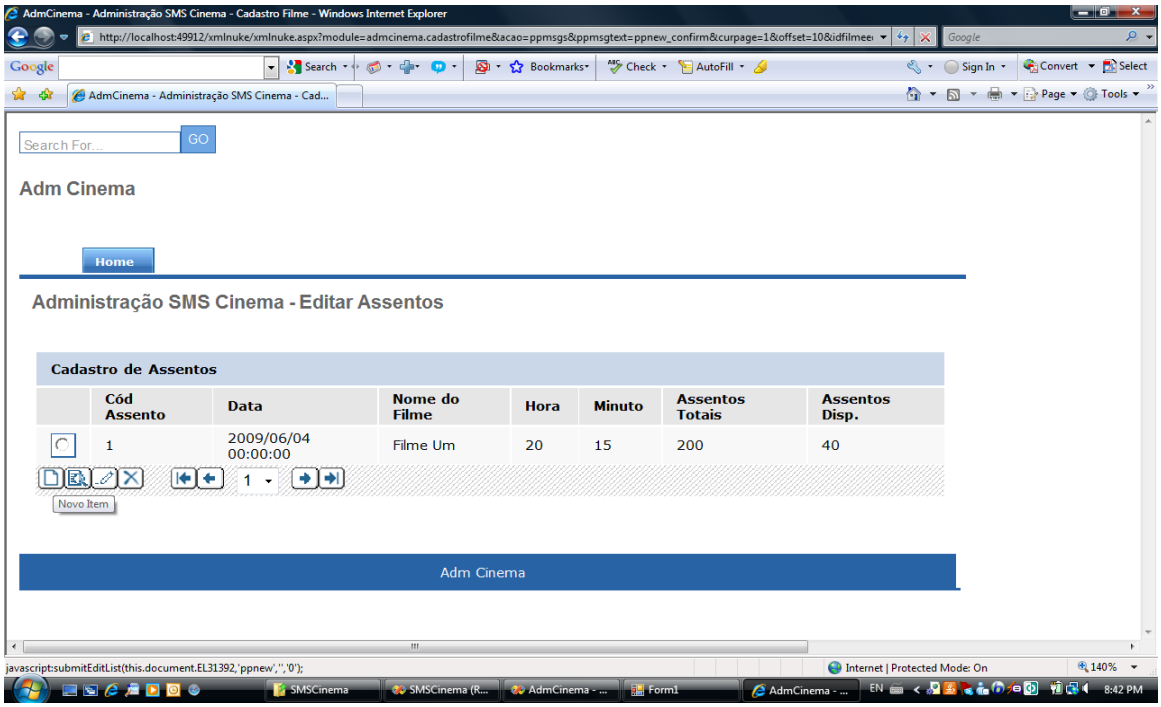
Seleciona na tabela o filme cadastrado e em seguida opção: “Editar Assentos Disponíveis”.



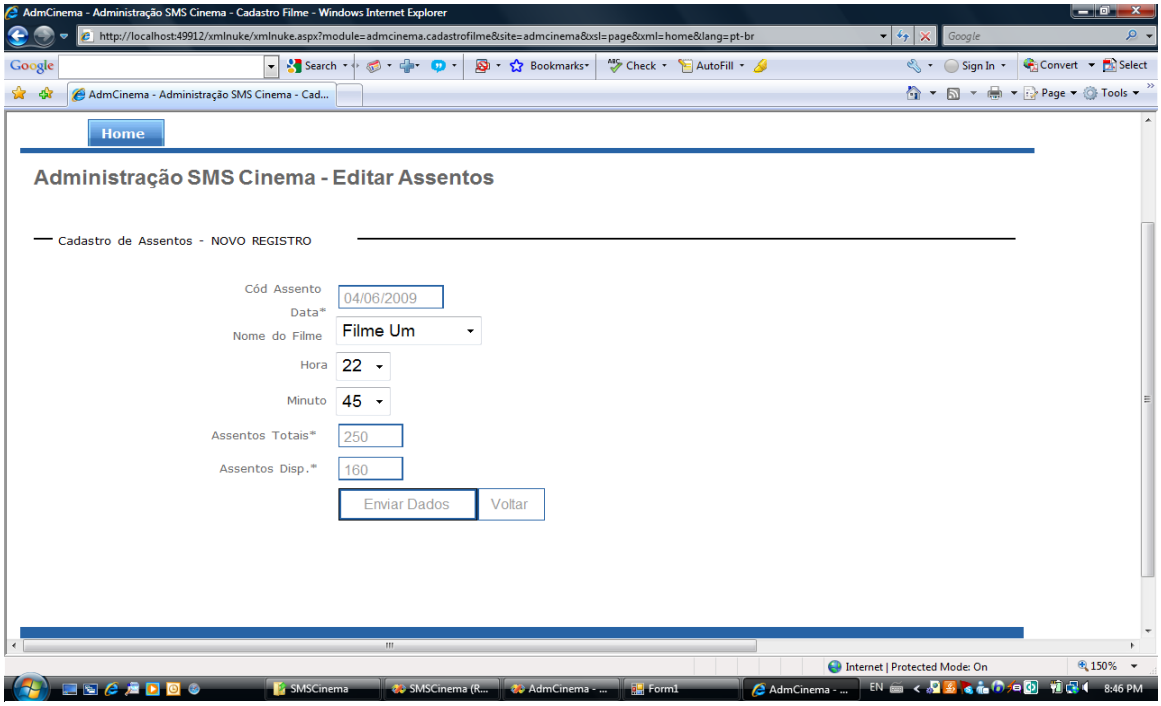
Para cadastrar os assentos totais e disponíveis para este filme selecione do menu a opção novo item.



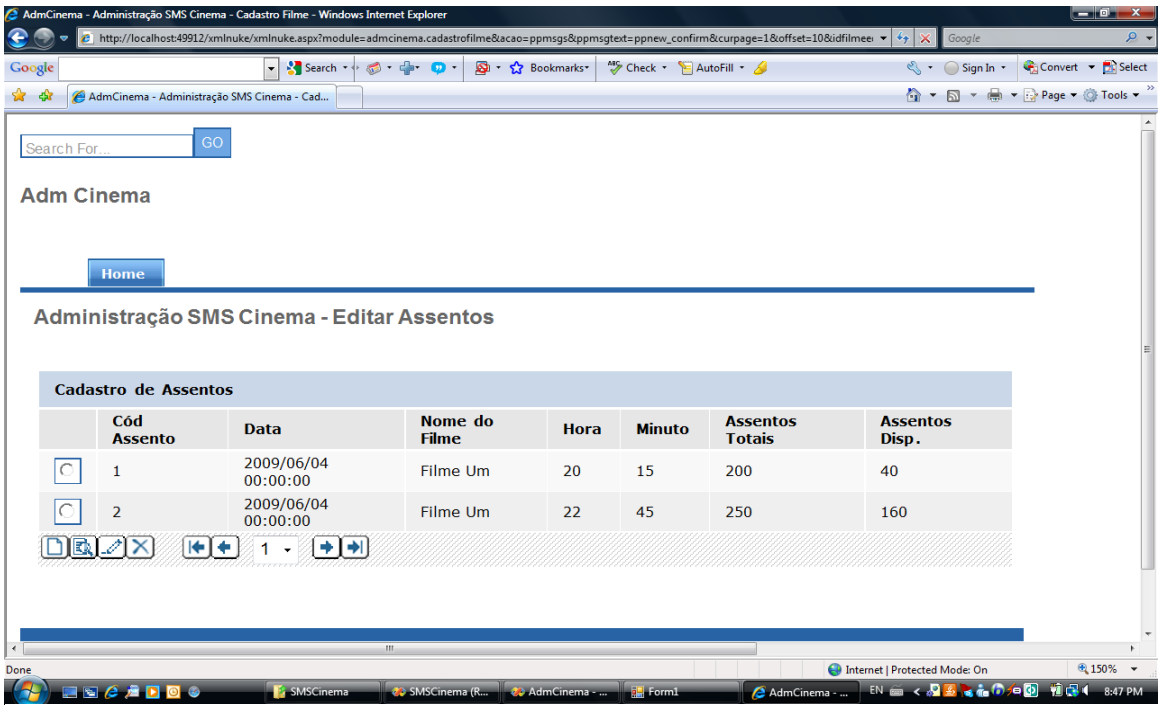
Inserido os dados manualmente em cada campo. Note que este exemplo obedece a relação percentual de 20% do número de assentos totais destinados para compra usando SMS.



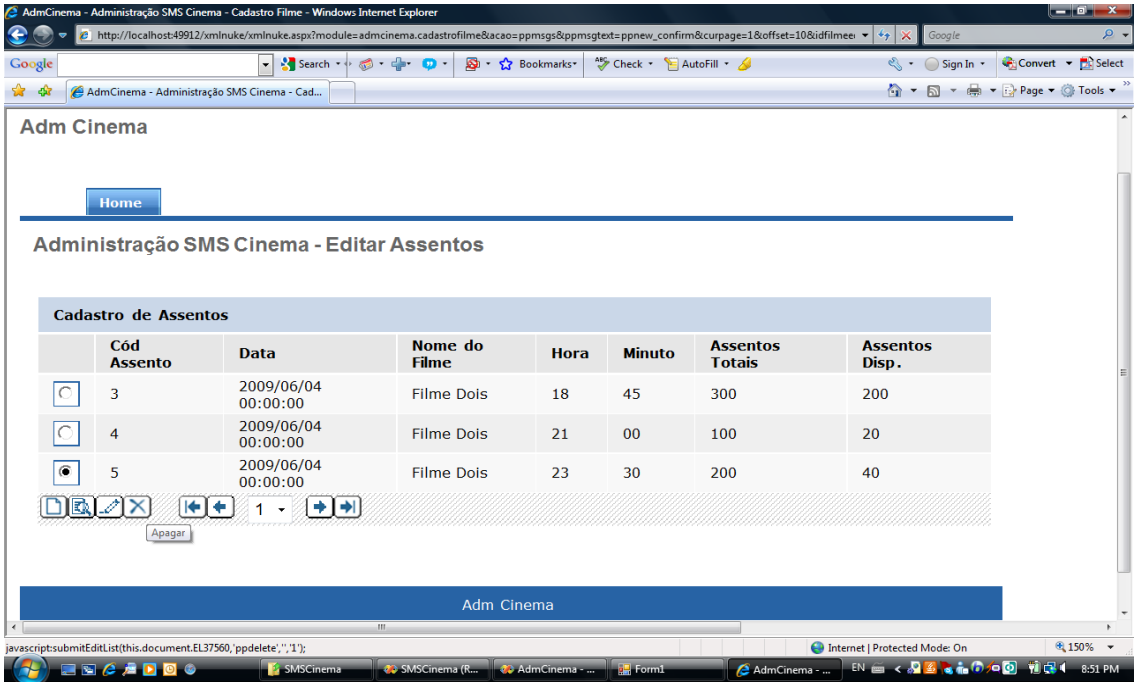
Mostra o resultado do cadastro.
Observe Novo Item para incluir uma outra sessão para este mesmo filme.



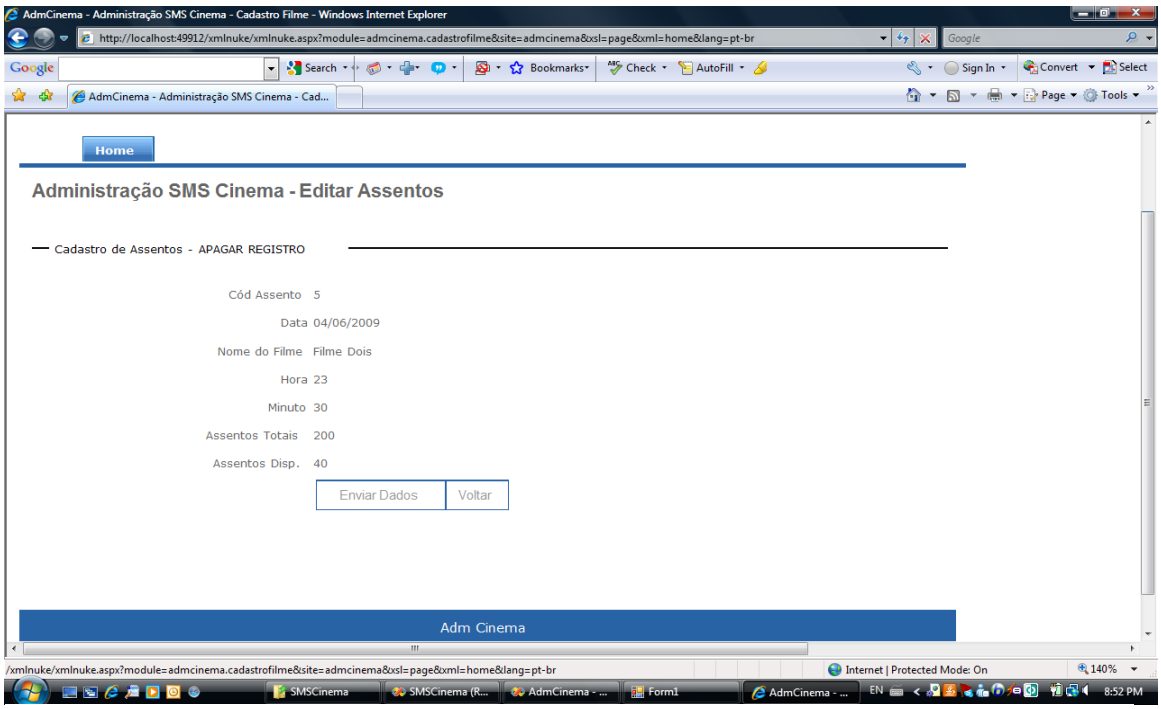
Cadastrado de uma nova sessão para um filme já cadastrado no sistema.



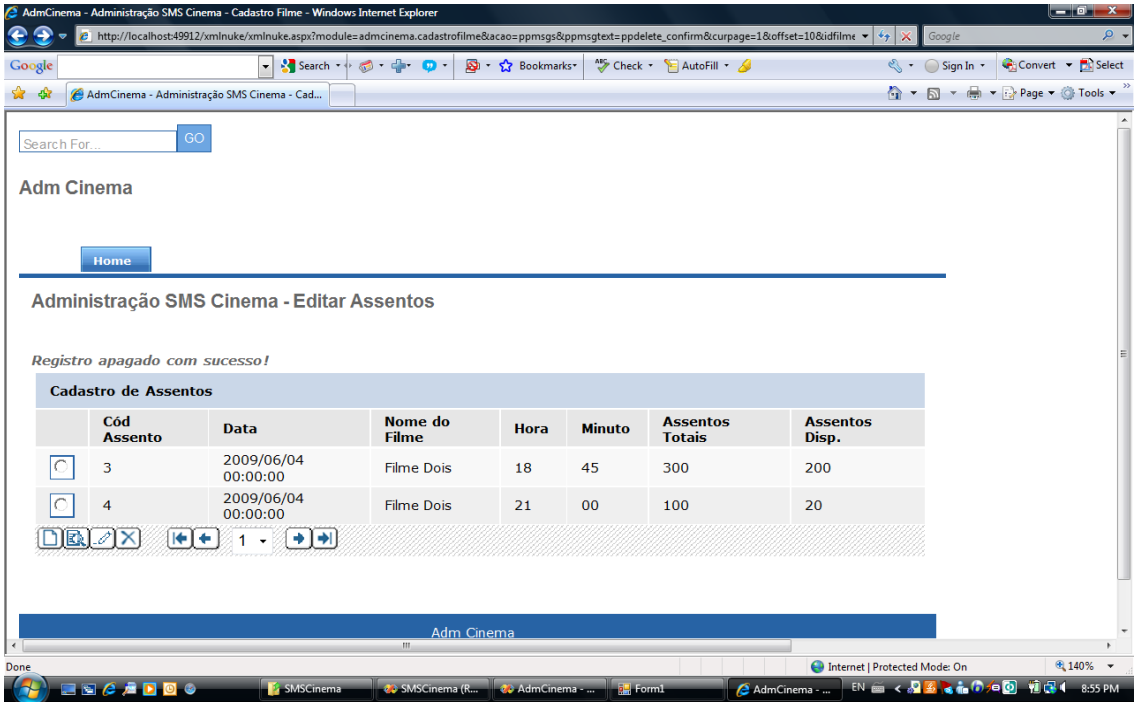
Resultado. Agora existem duas sessões cadastradas para o Filme Um.



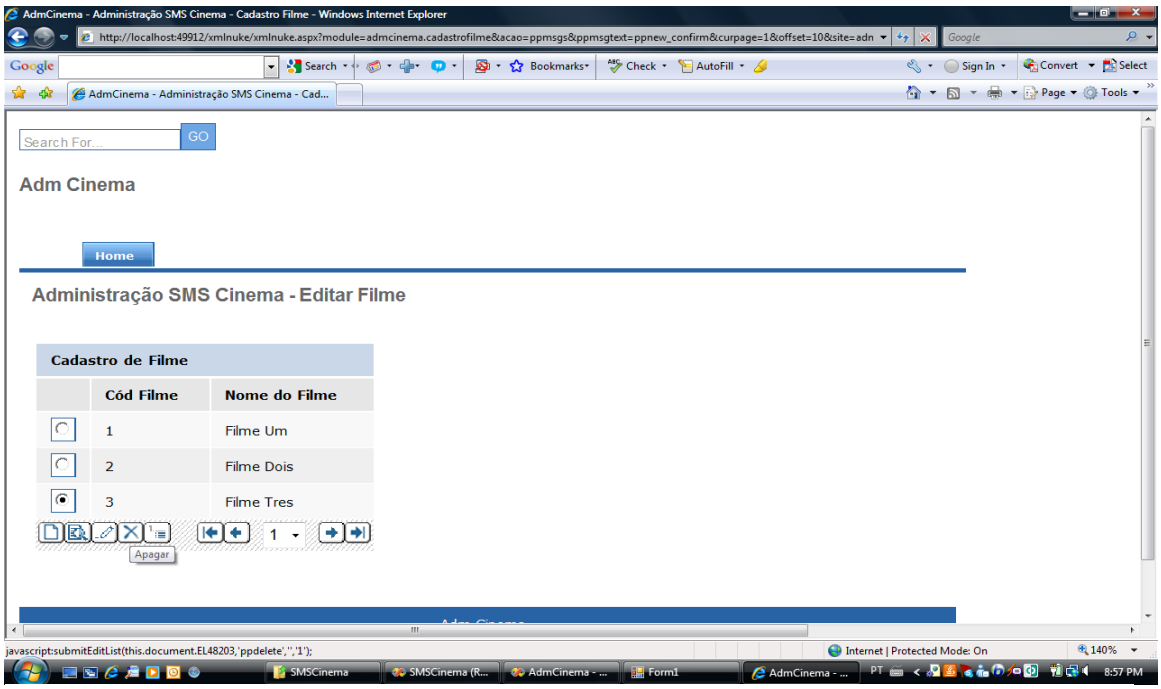
Observe existem três sessões cadastradas para o Filme Dois.
Para apagar uma sessão seleciona a linha e do menu comando apagar.
No exemplo acima a terceira linha está selecionada para ser apagada do sistema.



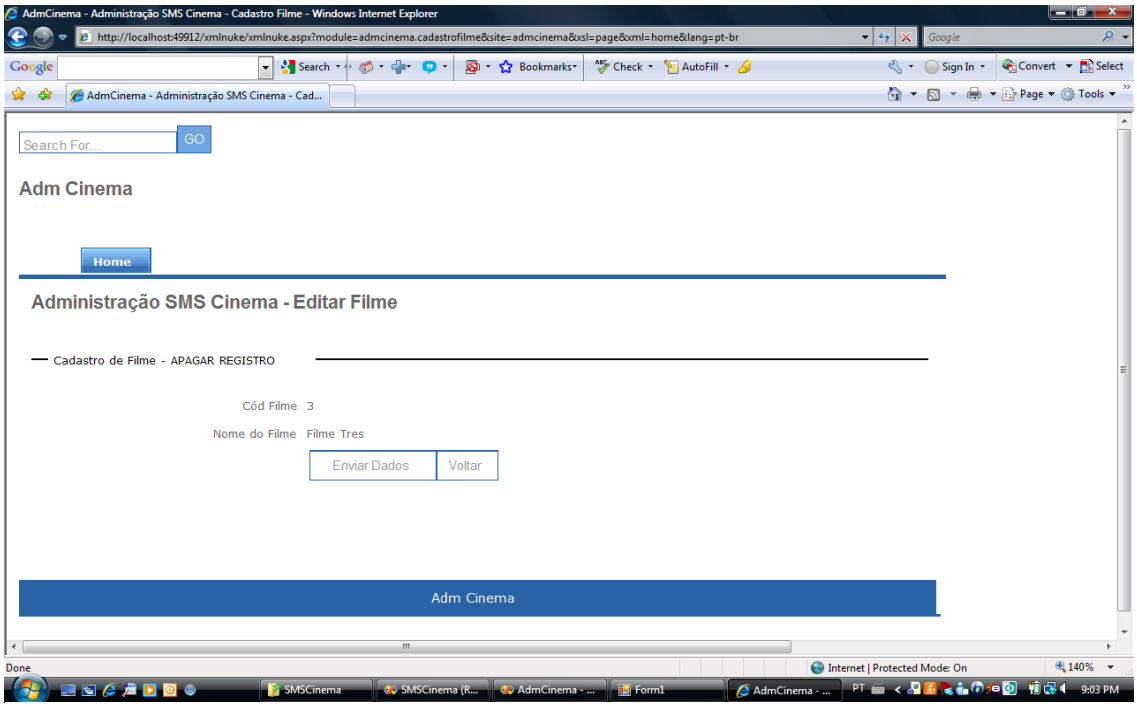
Confirma apagar mesmo a sessão: “Enviar Dados”, caso contrário “Voltar”.



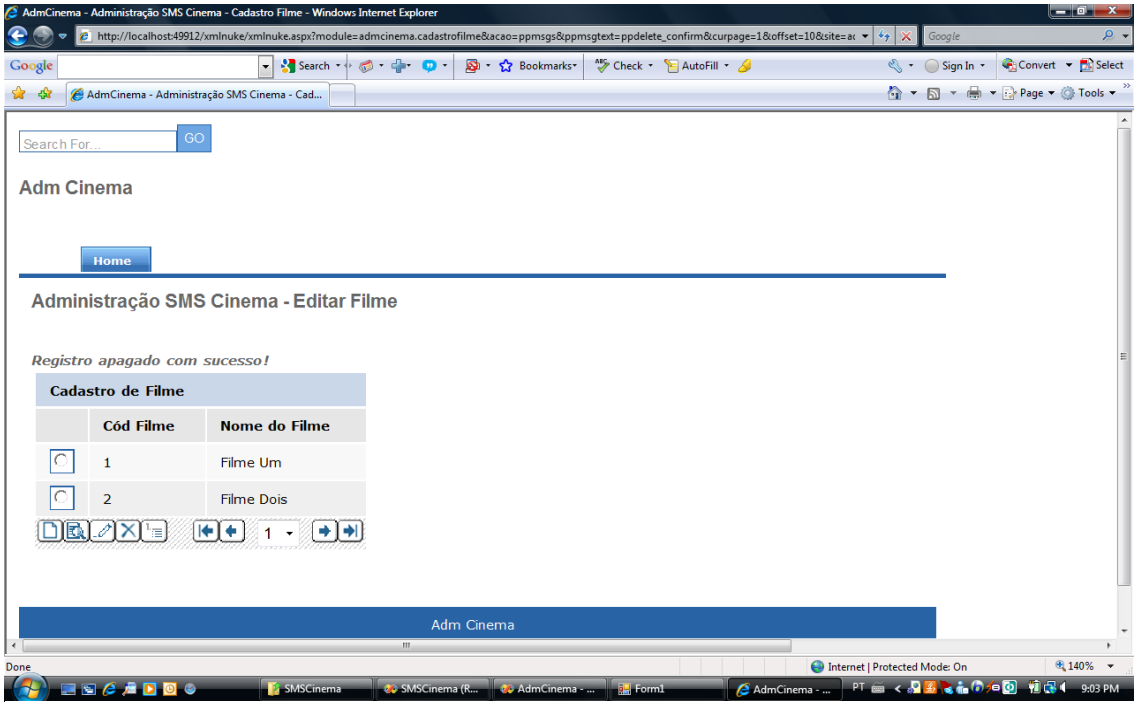
Mostra o resultado com a mensagem “Registro apagado com sucesso!” e as sessões que ainda estão disponíveis no sistema.



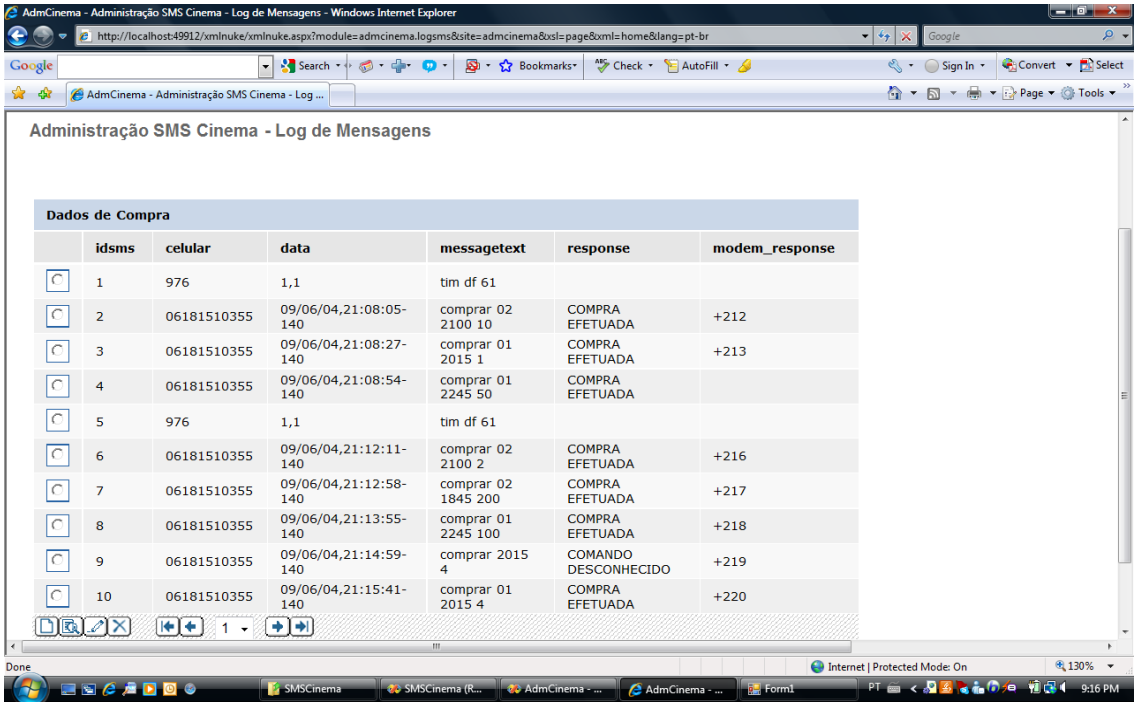
Esse exemplo mostra que também é possível apagar dados de um filme inteiro.



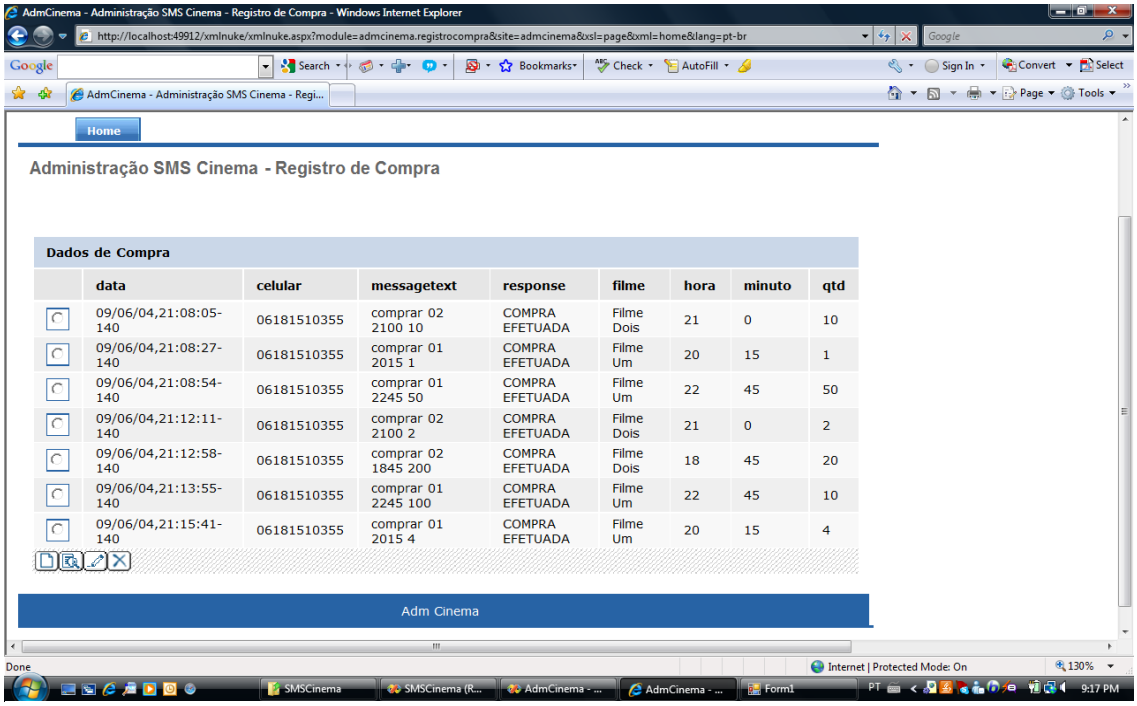
Confirmar comando.



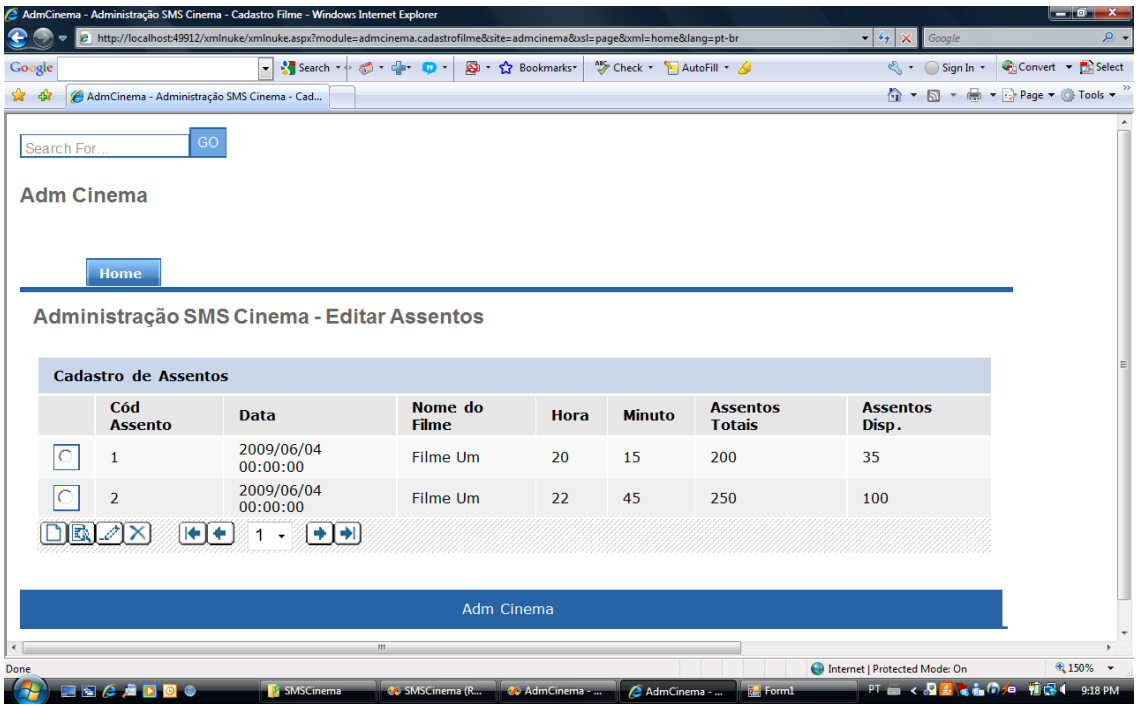
Mostra o resultado com a mensagem “Registro apagado com sucesso!” e os filmes que ainda estão disponíveis no sistema. Note que o Filme Três foi apagado do sistema.



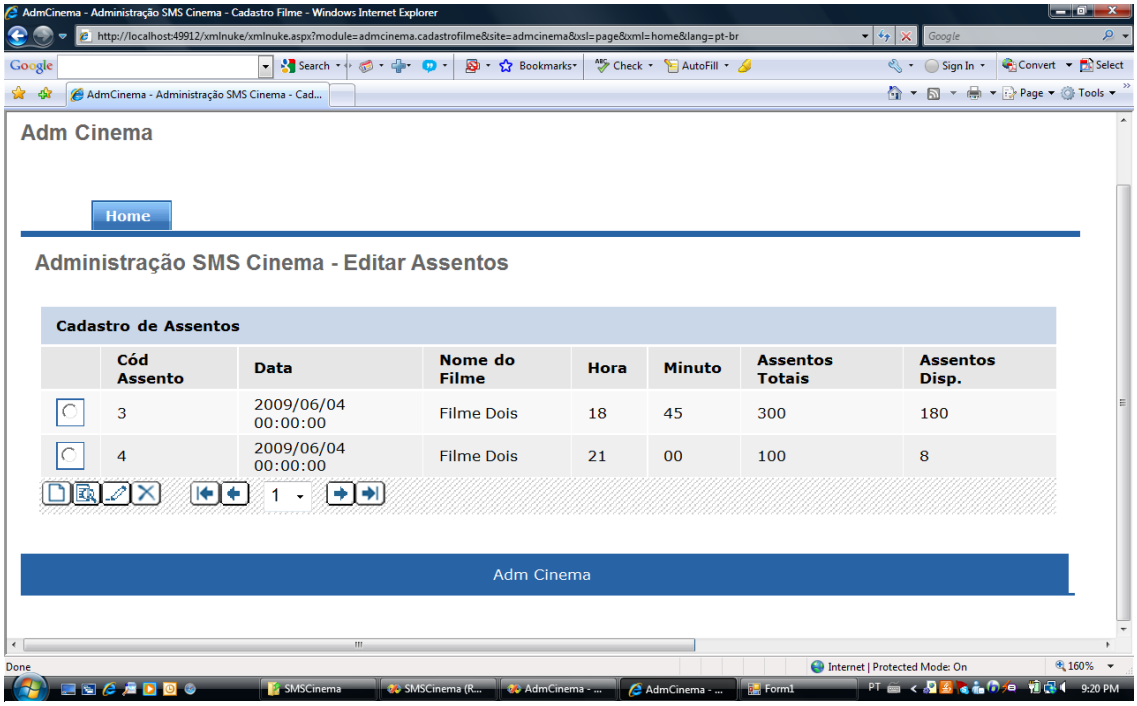
Opção “Ver Registro do SMS” da página principal que mostra a tabela com o “Log de Mensagens”.



Opção “Registro de compra” da página principal que mostra a tabela com todas as compras realizadas com sucesso.

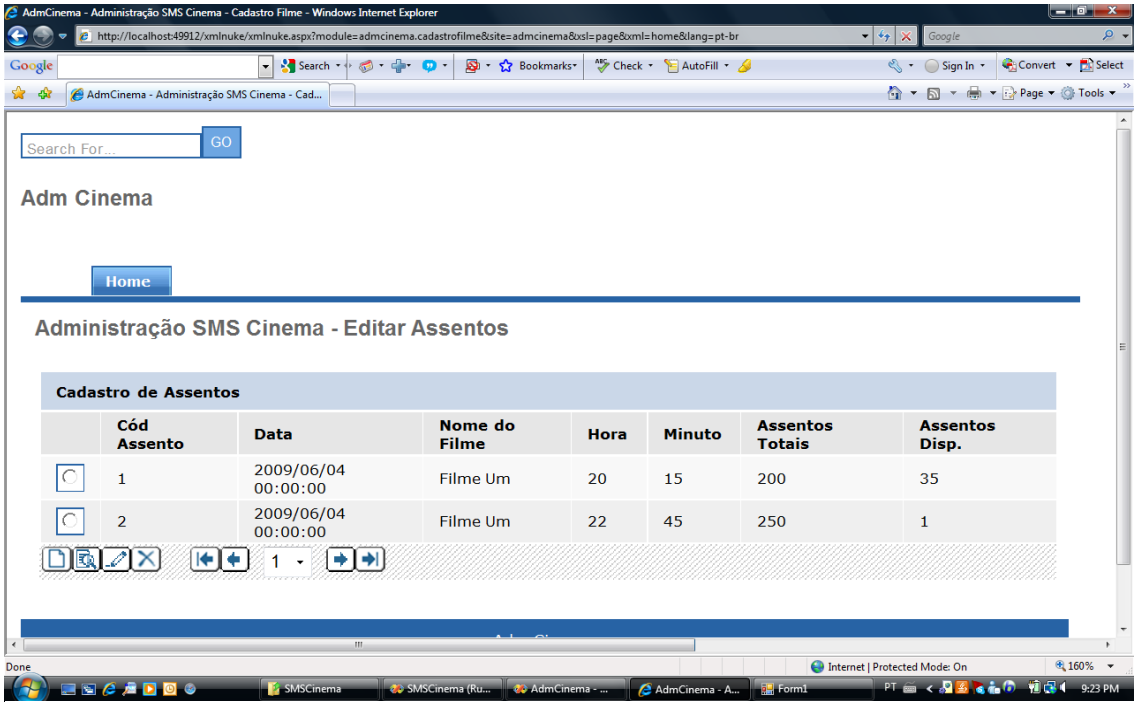


Depois de efetuada as compras observe que o sistema subtrai o número de ingressos disponíveis.

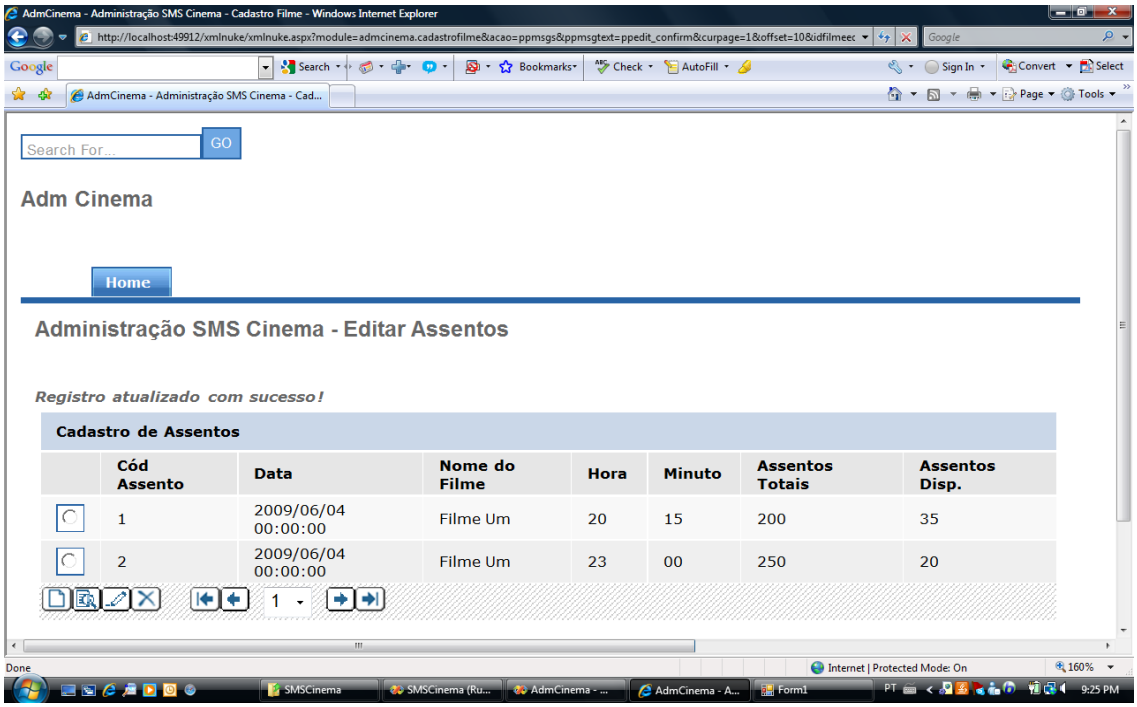


Note que quando tenta comprar ingressos acima de três dígitos ele descarta o ultimo algoritmo. Ex: 200 ele considerou como 20 bilhetes apenas.

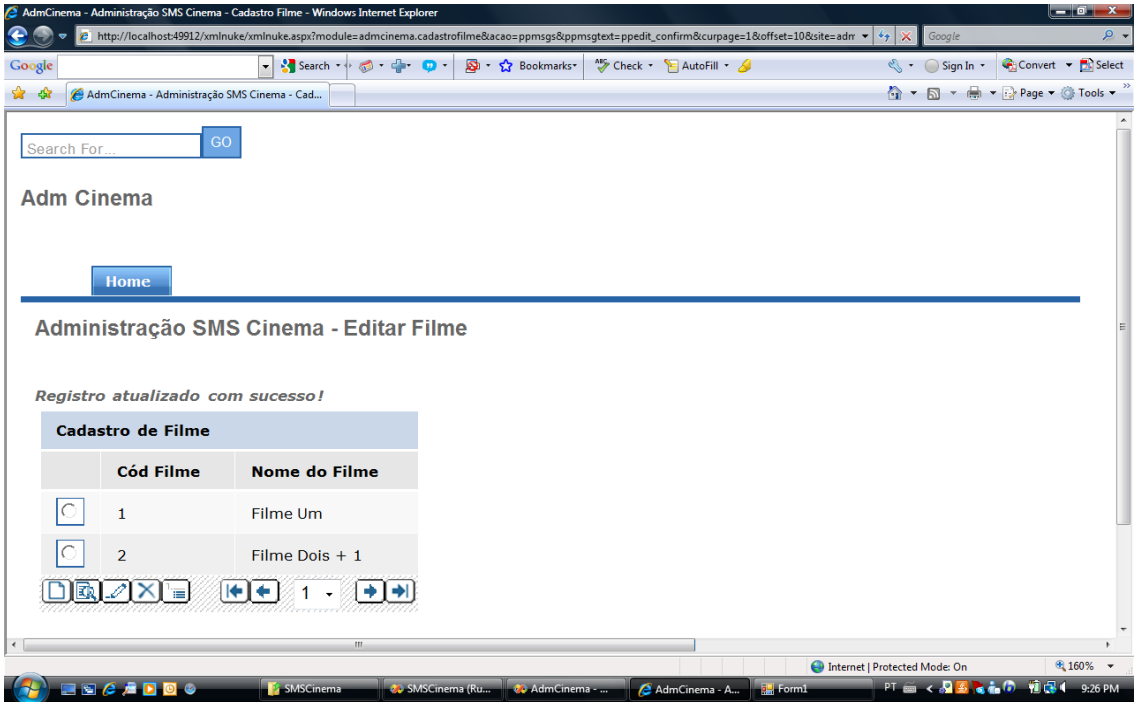
O sistema aceita comprar ingressos até 99 bilhetes conforme registro na figura acima.



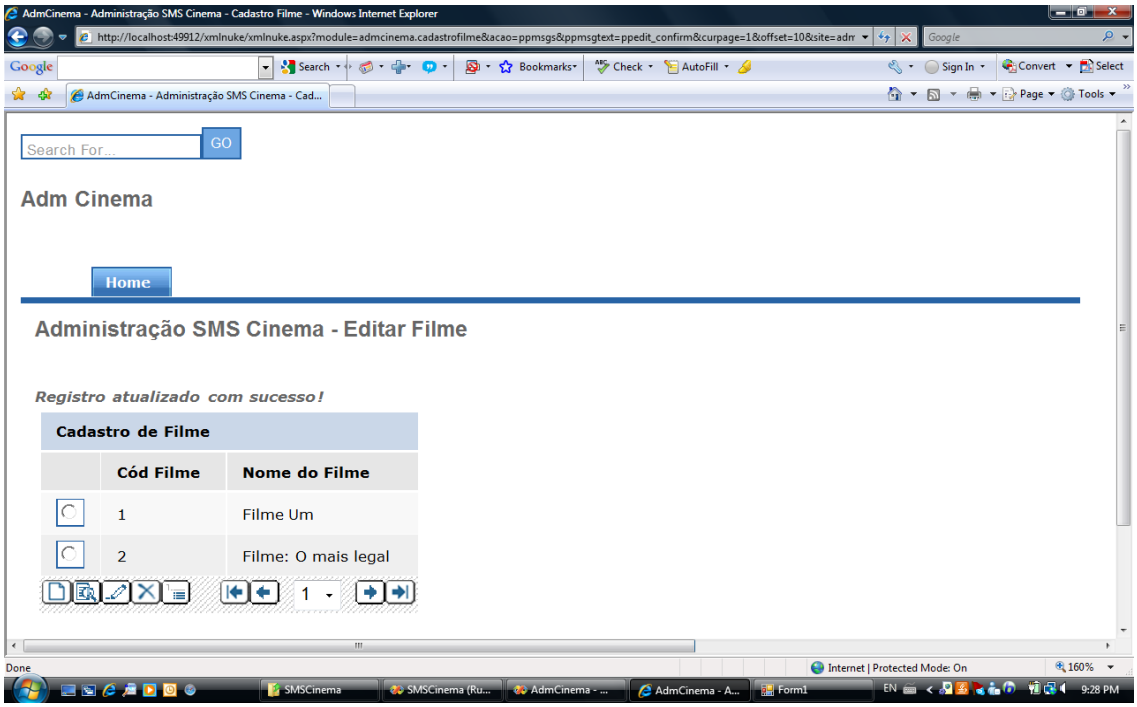
A cada compra efetuada o sistema atualiza a quantidade de ingressos disponíveis.



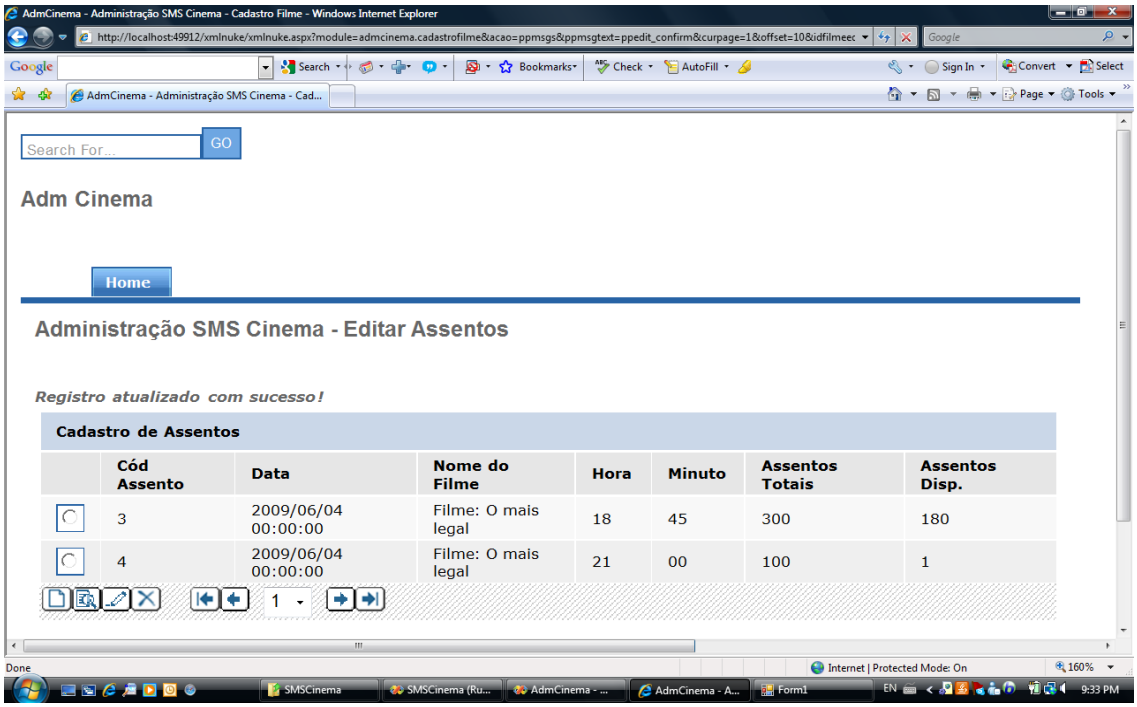
A figura mostra “Registro atualizado com sucesso” porque o sistema permite alterar o horário e o número de assentos totais ou disponíveis para um filme já cadastrado.



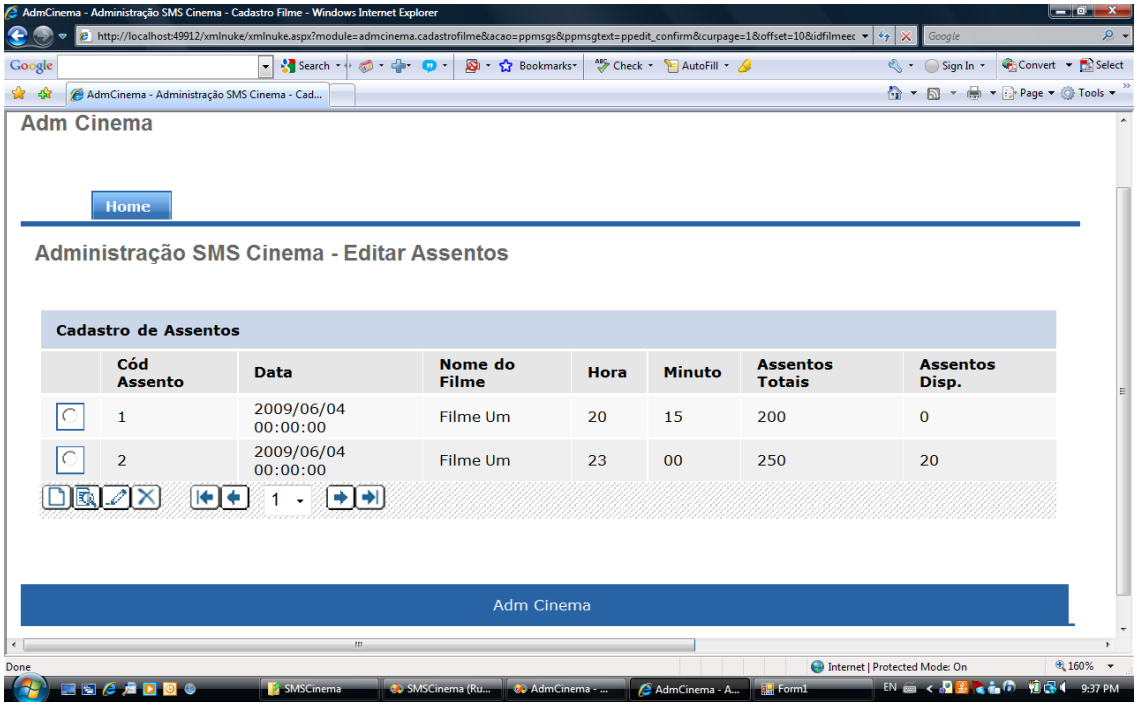
Este exemplo mostra que foi alterado o nome do Filme Dois para Filme Dois + 1.



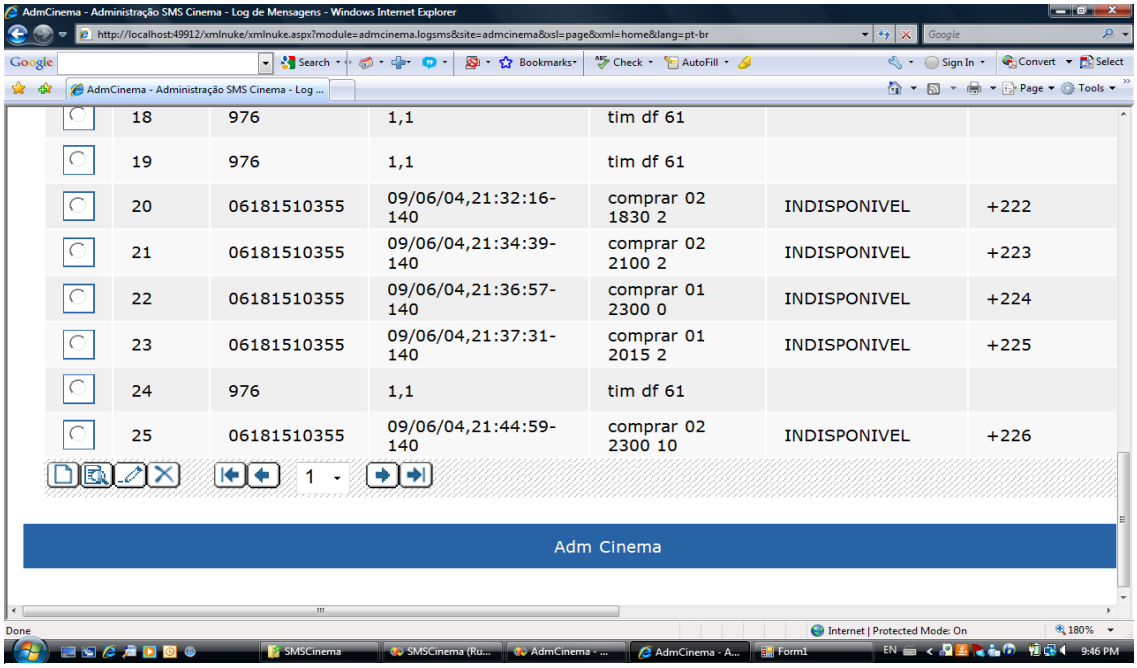
Este exemplo mostra que foi alterado o nome do Filme Dois + 1 para Filme: O mais legal.



Esses foram os registros utilizados para fazer o teste do comando INDISPONIVEL para casos 1 e 2 descritos na lista no item 5.3 do capítulo 5.

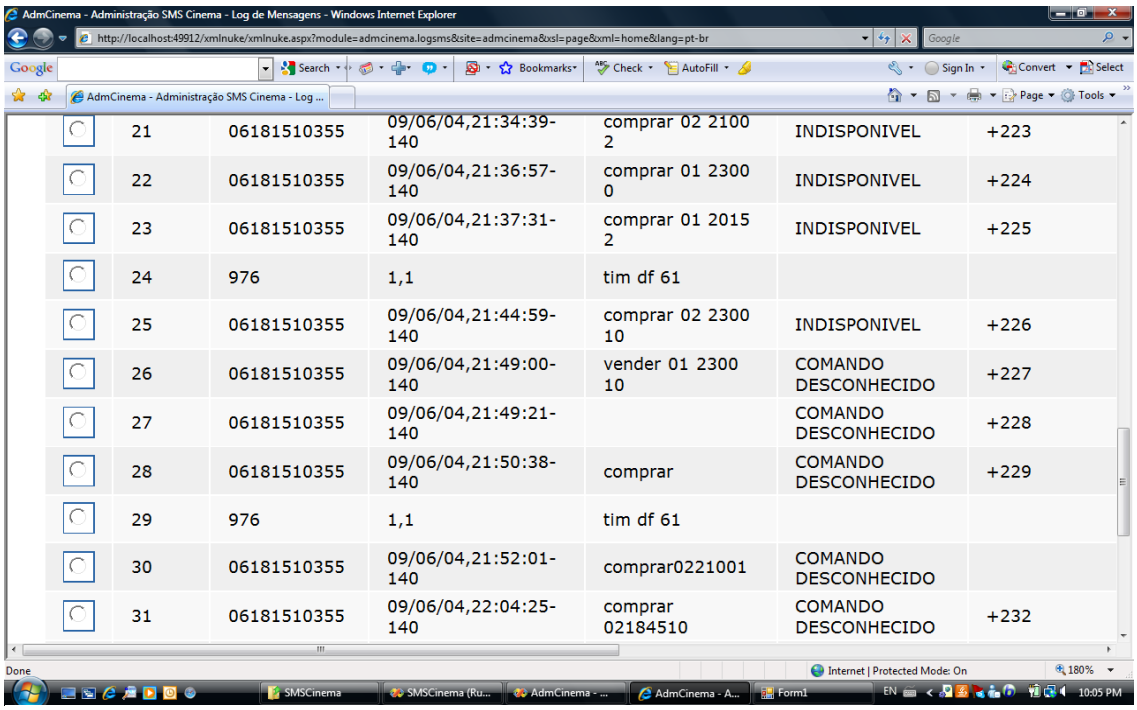


Esses foram os registros utilizados para fazer o teste do comando INDISPONIVEL para casos 3 e 4 descritos na lista no item 5.3 do capítulo 5.



	18	976	1,1	tim df 61		
	19	976	1,1	tim df 61		
	20	06181510355	09/06/04,21:32:16-140	comprar 02 1830 2	INDISPONIVEL	+222
	21	06181510355	09/06/04,21:34:39-140	comprar 02 2100 2	INDISPONIVEL	+223
	22	06181510355	09/06/04,21:36:57-140	comprar 01 2300 0	INDISPONIVEL	+224
	23	06181510355	09/06/04,21:37:31-140	comprar 01 2015 2	INDISPONIVEL	+225
	24	976	1,1	tim df 61		
	25	06181510355	09/06/04,21:44:59-140	comprar 02 2300 10	INDISPONIVEL	+226

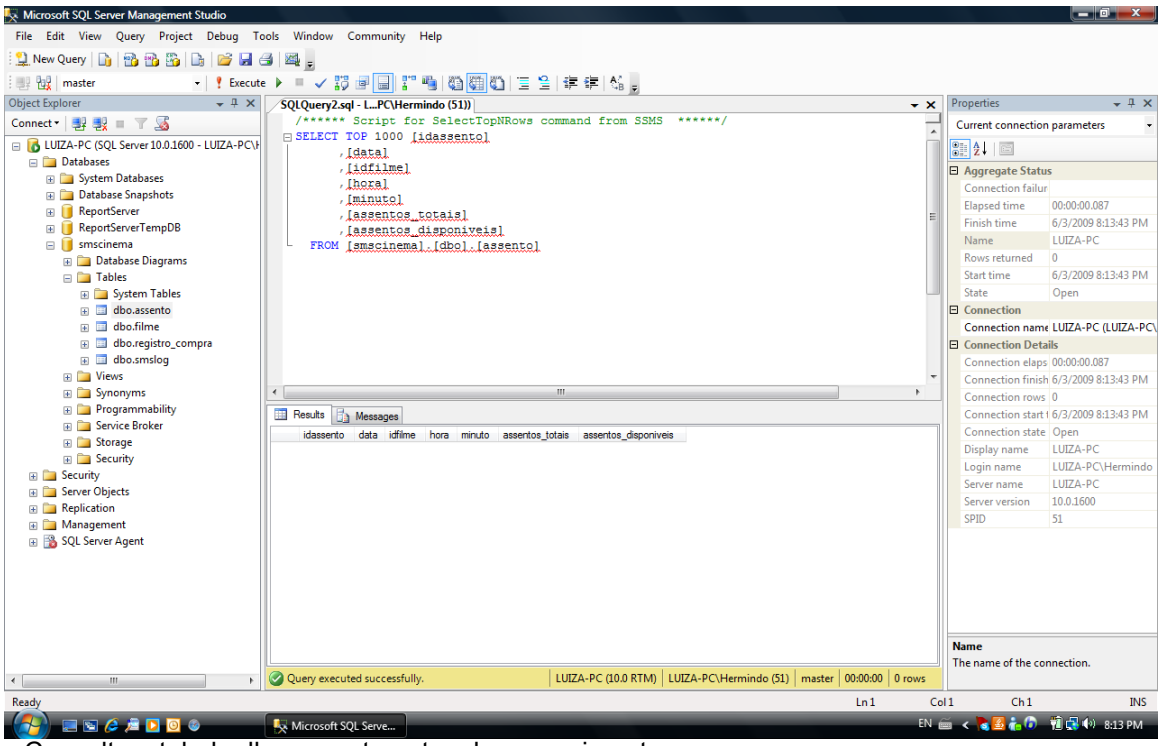
Do menu principal “Ver Registro SMS” mostra a tabela com o Log de Mensagens. Observe que na última linha a sessão está correta e o código do filme é inválido porque valeria se fosse para o código filme 01.



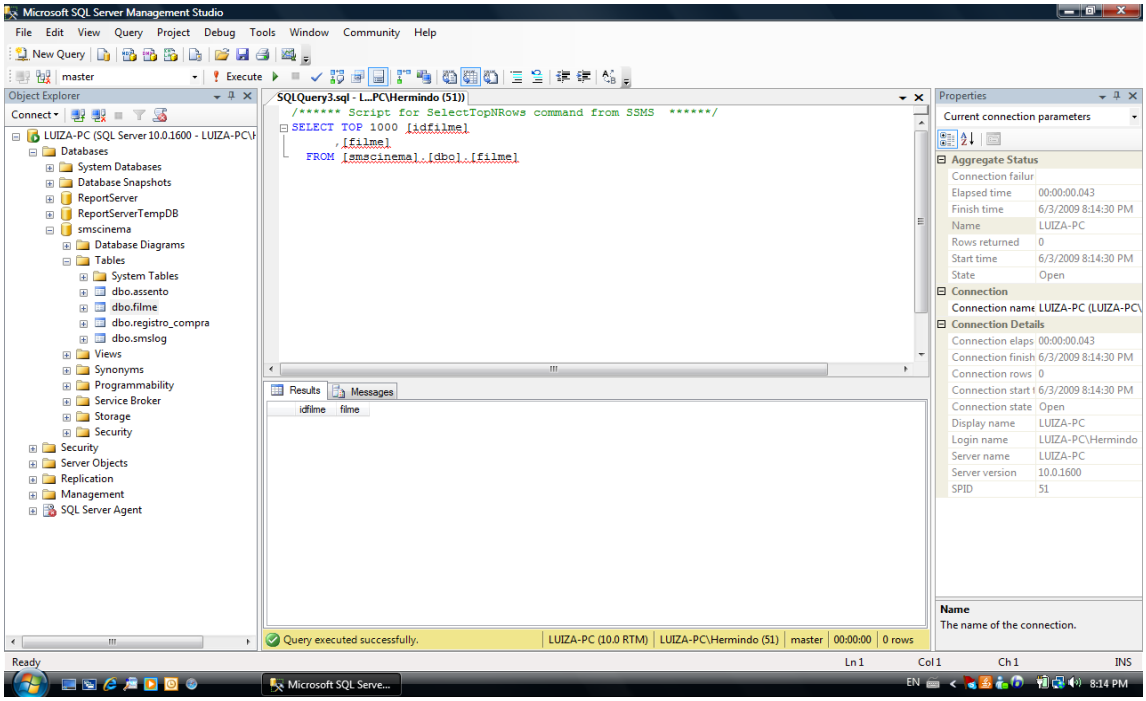
	21	06181510355	09/06/04,21:34:39-140	comprar 02 2100 2	INDISPONIVEL	+223
	22	06181510355	09/06/04,21:36:57-140	comprar 01 2300 0	INDISPONIVEL	+224
	23	06181510355	09/06/04,21:37:31-140	comprar 01 2015 2	INDISPONIVEL	+225
	24	976	1,1	tim df 61		
	25	06181510355	09/06/04,21:44:59-140	comprar 02 2300 10	INDISPONIVEL	+226
	26	06181510355	09/06/04,21:49:00-140	vender 01 2300 10	COMANDO DESCONHECIDO	+227
	27	06181510355	09/06/04,21:49:21-140		COMANDO DESCONHECIDO	+228
	28	06181510355	09/06/04,21:50:38-140	comprar	COMANDO DESCONHECIDO	+229
	29	976	1,1	tim df 61		
	30	06181510355	09/06/04,21:52:01-140	comprar0221001	COMANDO DESCONHECIDO	
	31	06181510355	09/06/04,22:04:25-140	comprar 02184510	COMANDO DESCONHECIDO	+232

A tabela Log de Mensagens mostra os registros para os casos de COMANDO DESCONHECIDO.

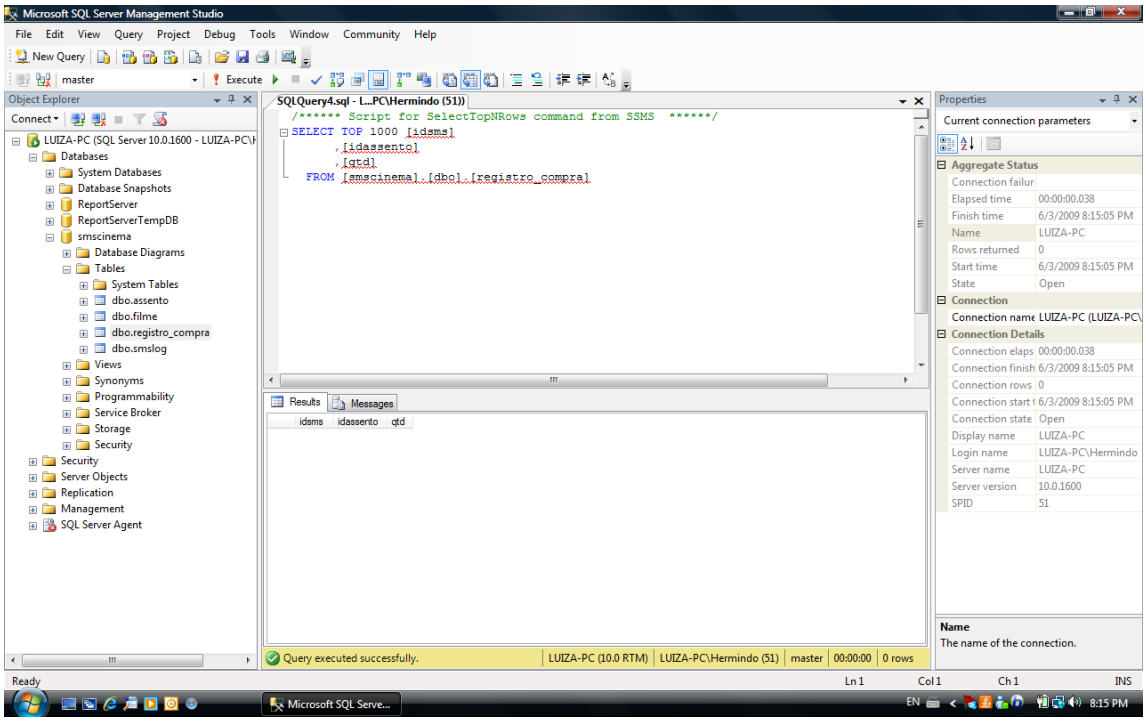
COMPARANDO CONSULTA SQL VERSUS INTERFACE



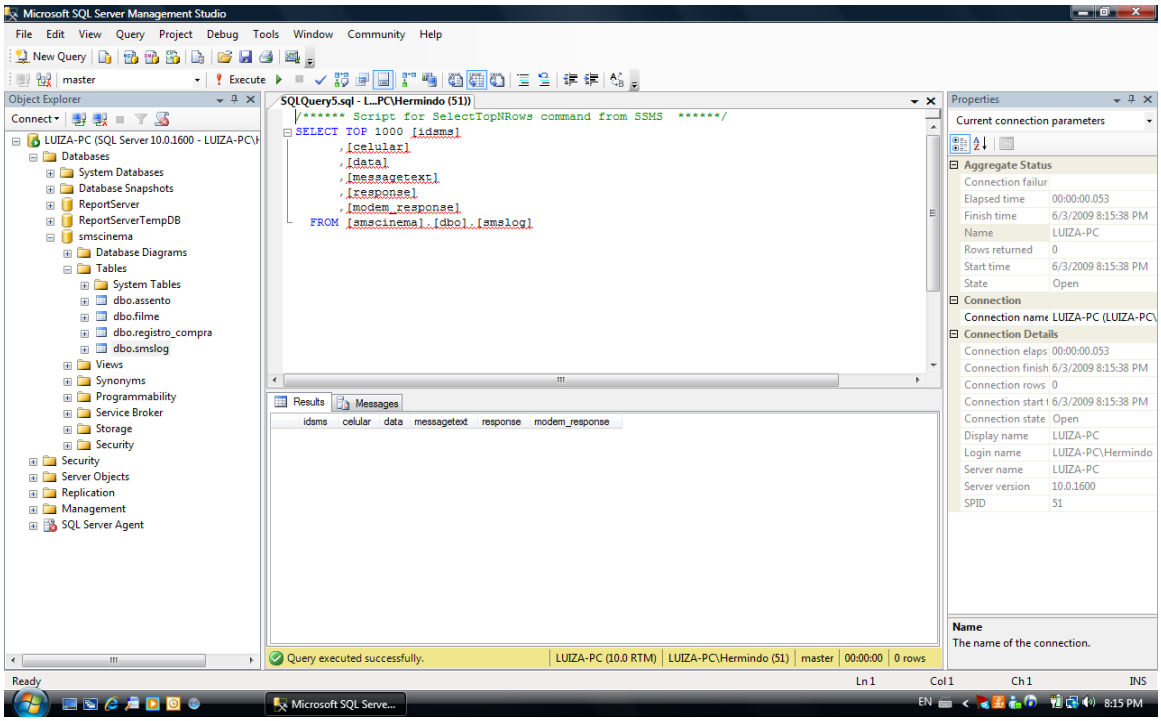
Consulta a tabela dbo.assento antes dos experimentos.



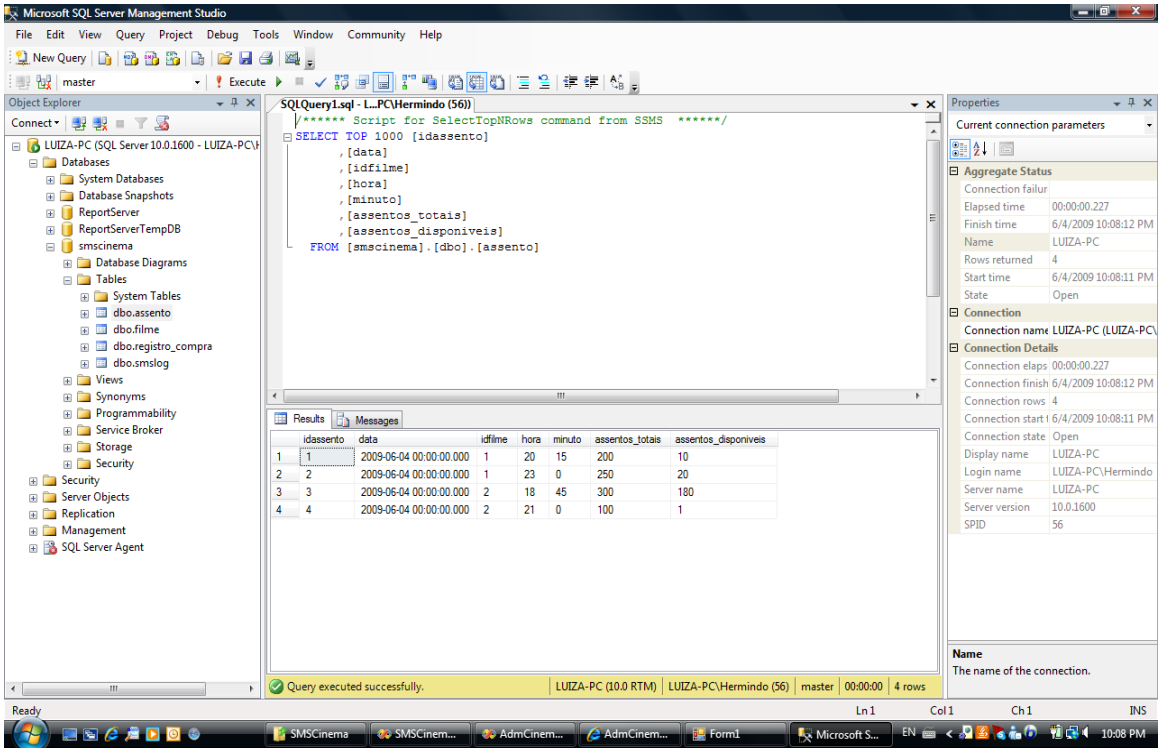
Consulta a tabela dbo.filme antes dos experimentos.



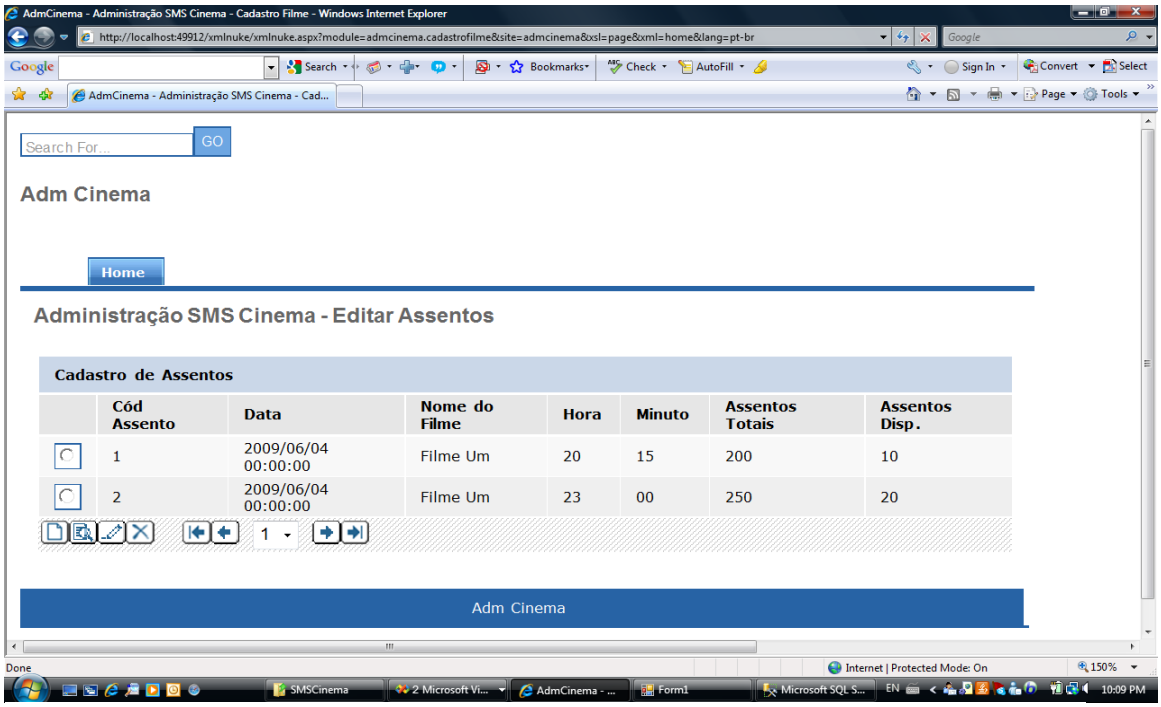
Consulta a tabela dbo.registro_compra antes dos experimentos.



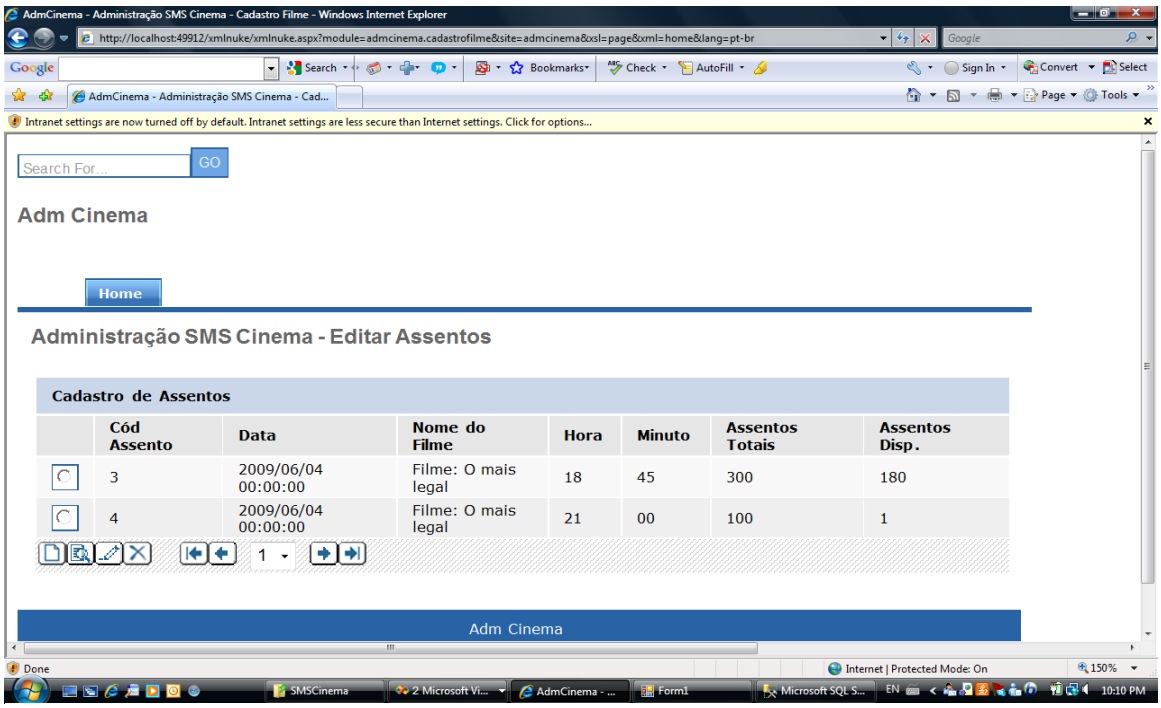
Consulta a tabela dbo.*smslog* antes dos experimentos.



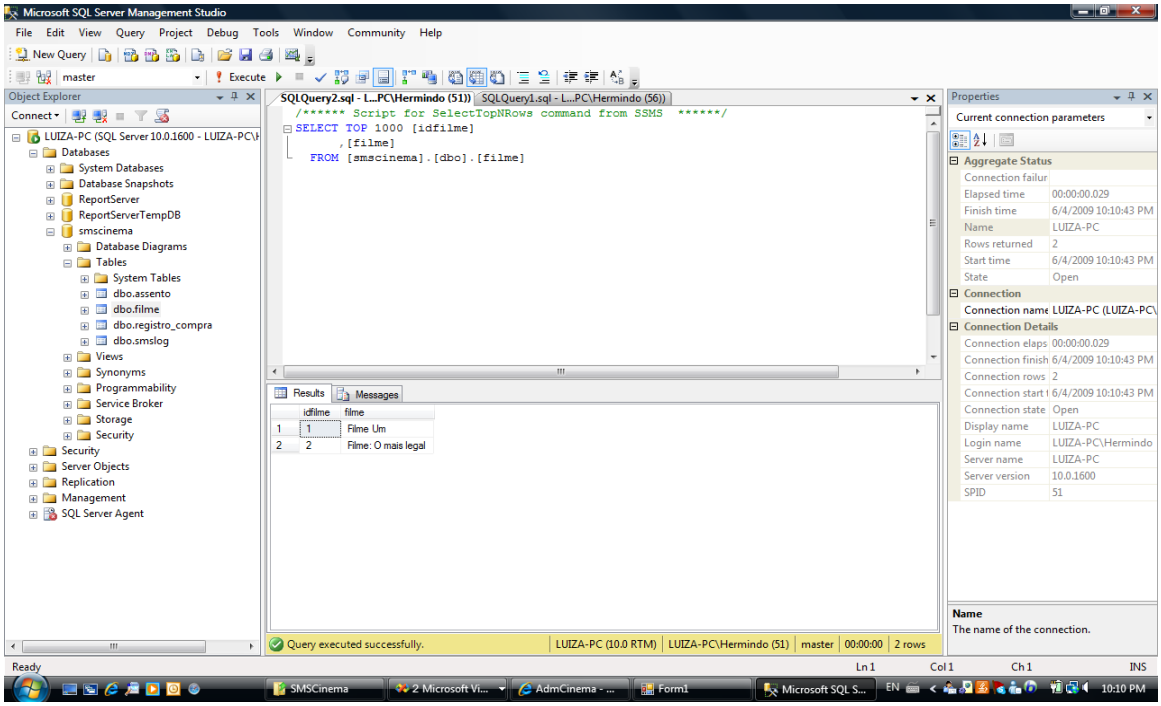
Consulta pelo SQL *Server Management Studio* a tabela dbo.assento.



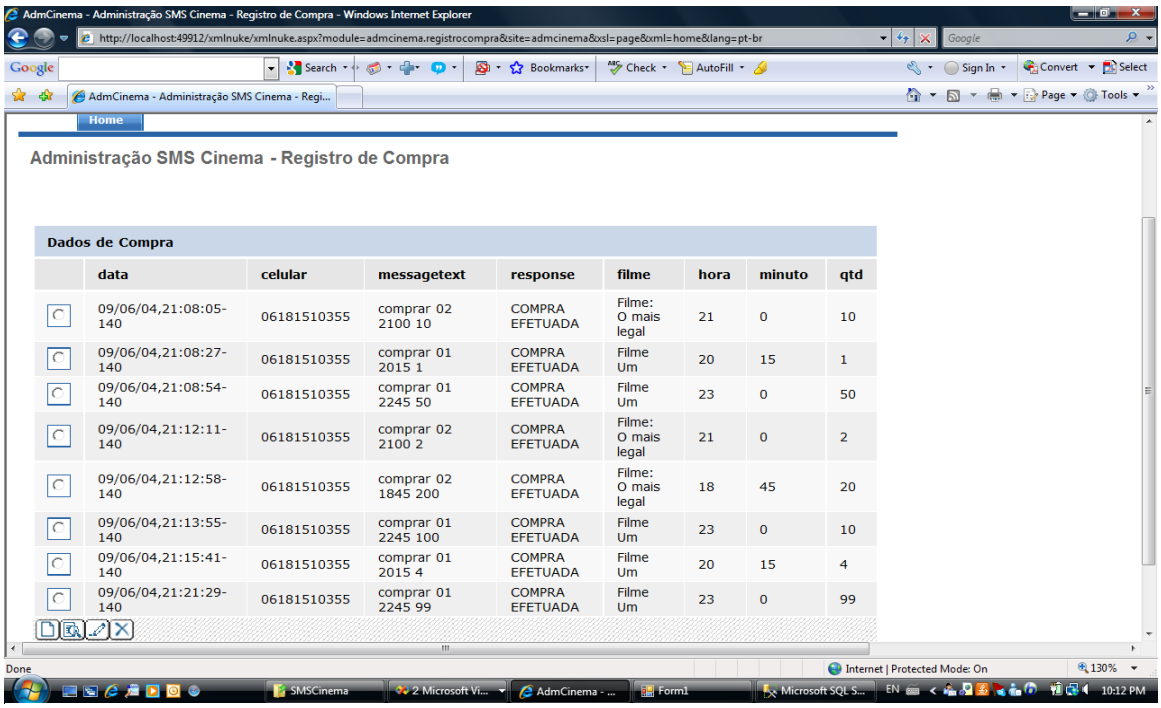
Pela interface para o Filme 1 tem a primeira sessão têm 10 assentos disponíveis para compra usando SMS e para sessão 2 têm 20 assentos disponíveis.



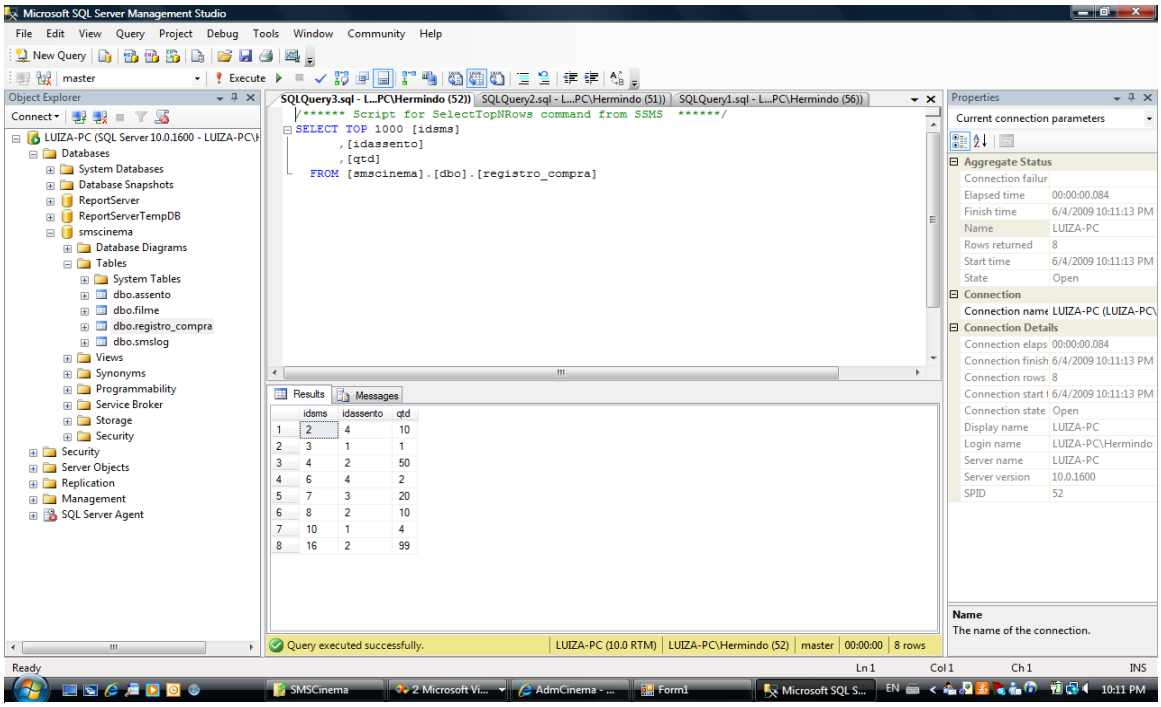
Para o Filme: O mais legal para a primeira sessão têm 180 assentos disponíveis e para segunda sessão tem apenas um 1 ingresso disponível no momento.



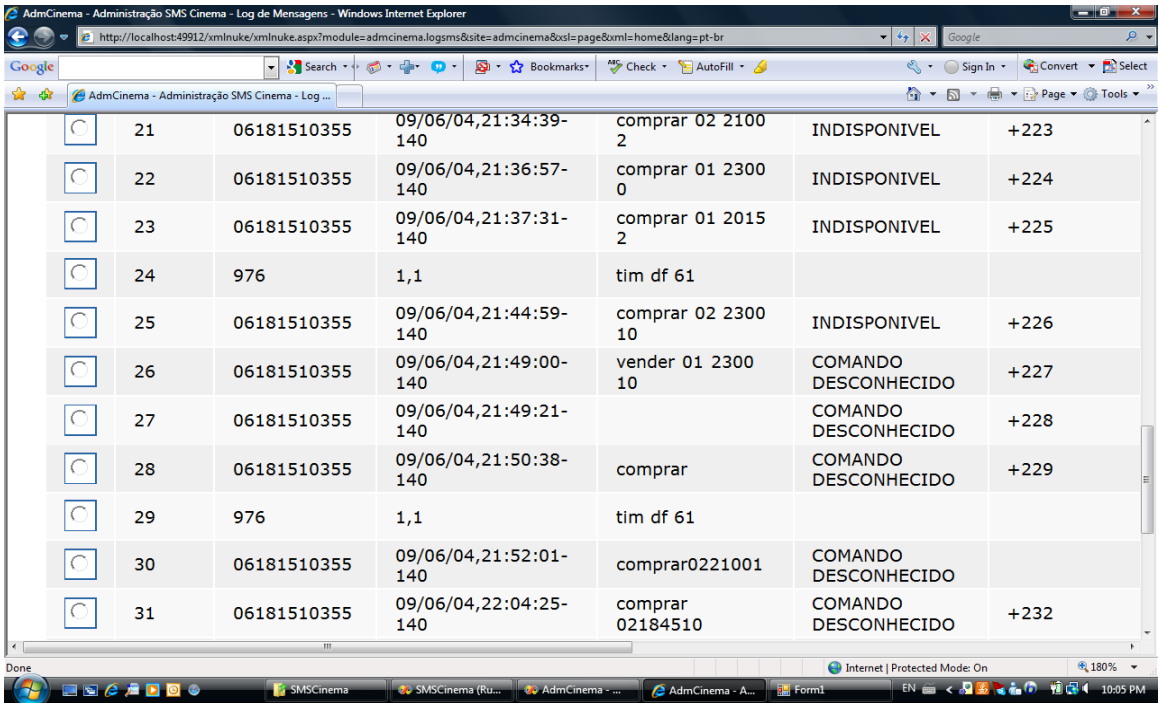
Consulta pelo SQL *Server Management Studio* a tabela dbo.filme mostra que existem dois filmes cadastrados no sistema.



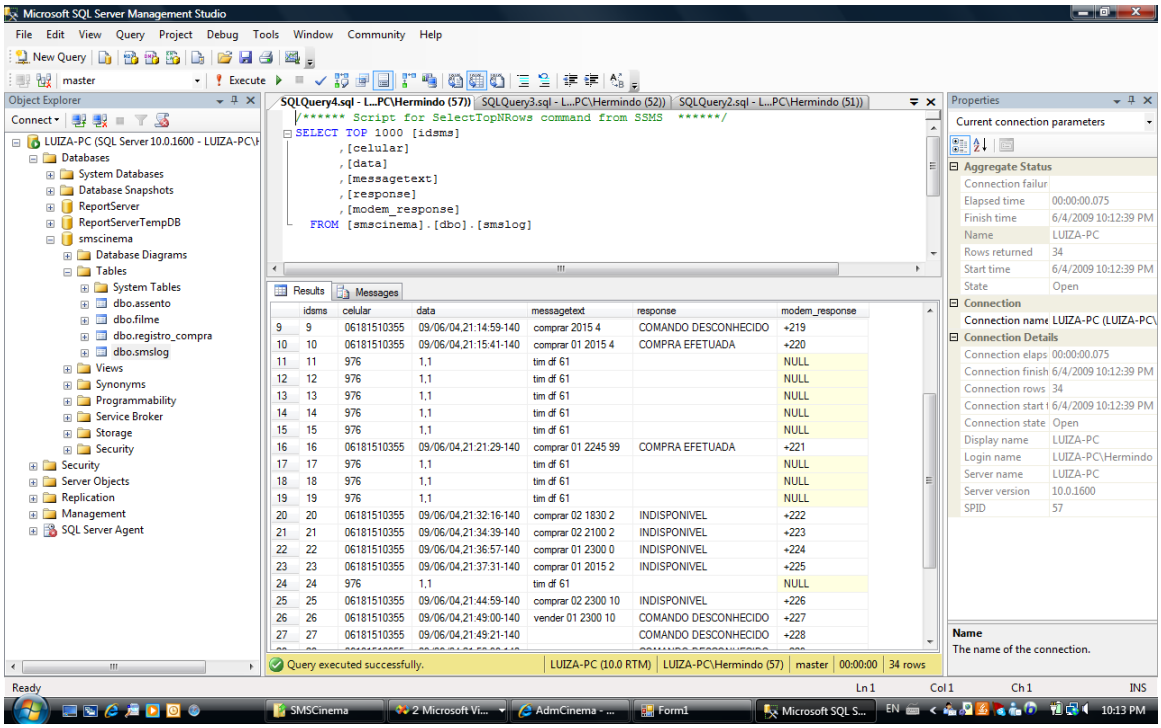
Visualização da tabela “Registro de Compra” pela interface web.



Consulta pelo SQL *Server Management Studio* a tabela dbo.registro_compra.



Visualização da tabela “Log de Mensagens” pela interface web.

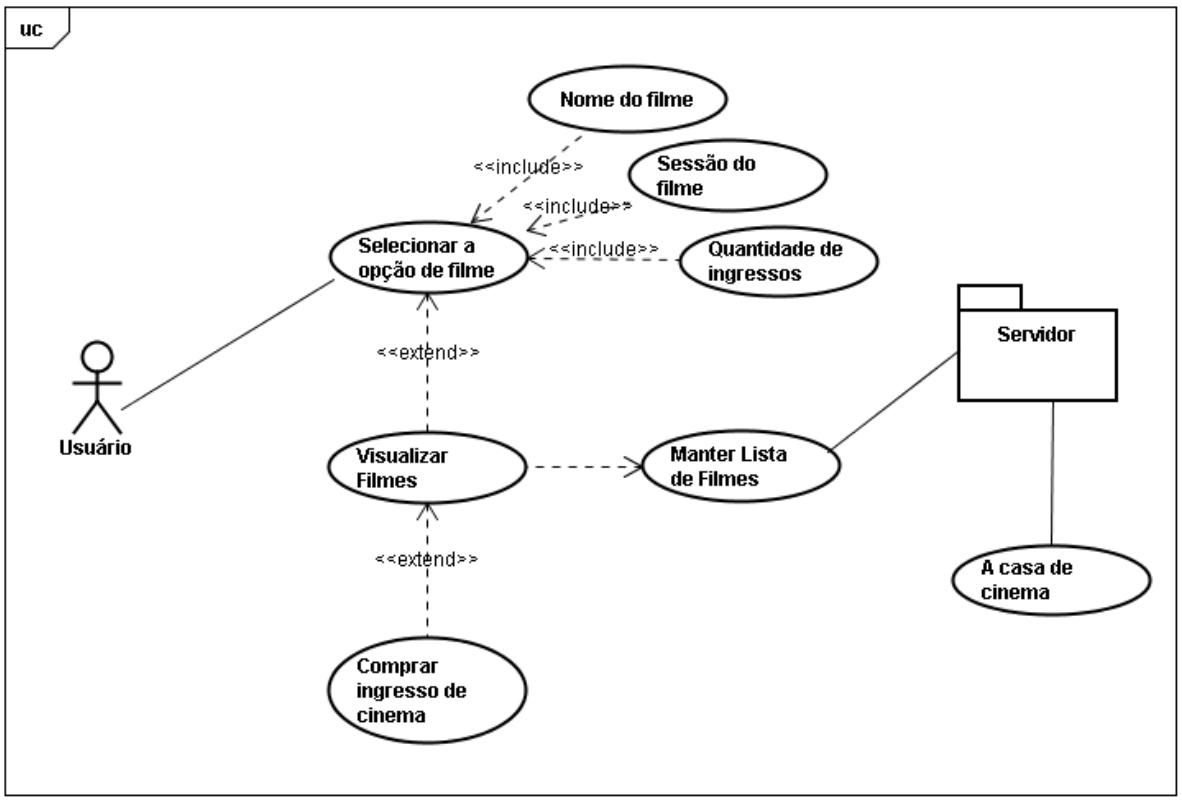


Consulta pelo SQL *Server Management Studio* a tabela dbo.*smslog* nas linhas equivalentes.

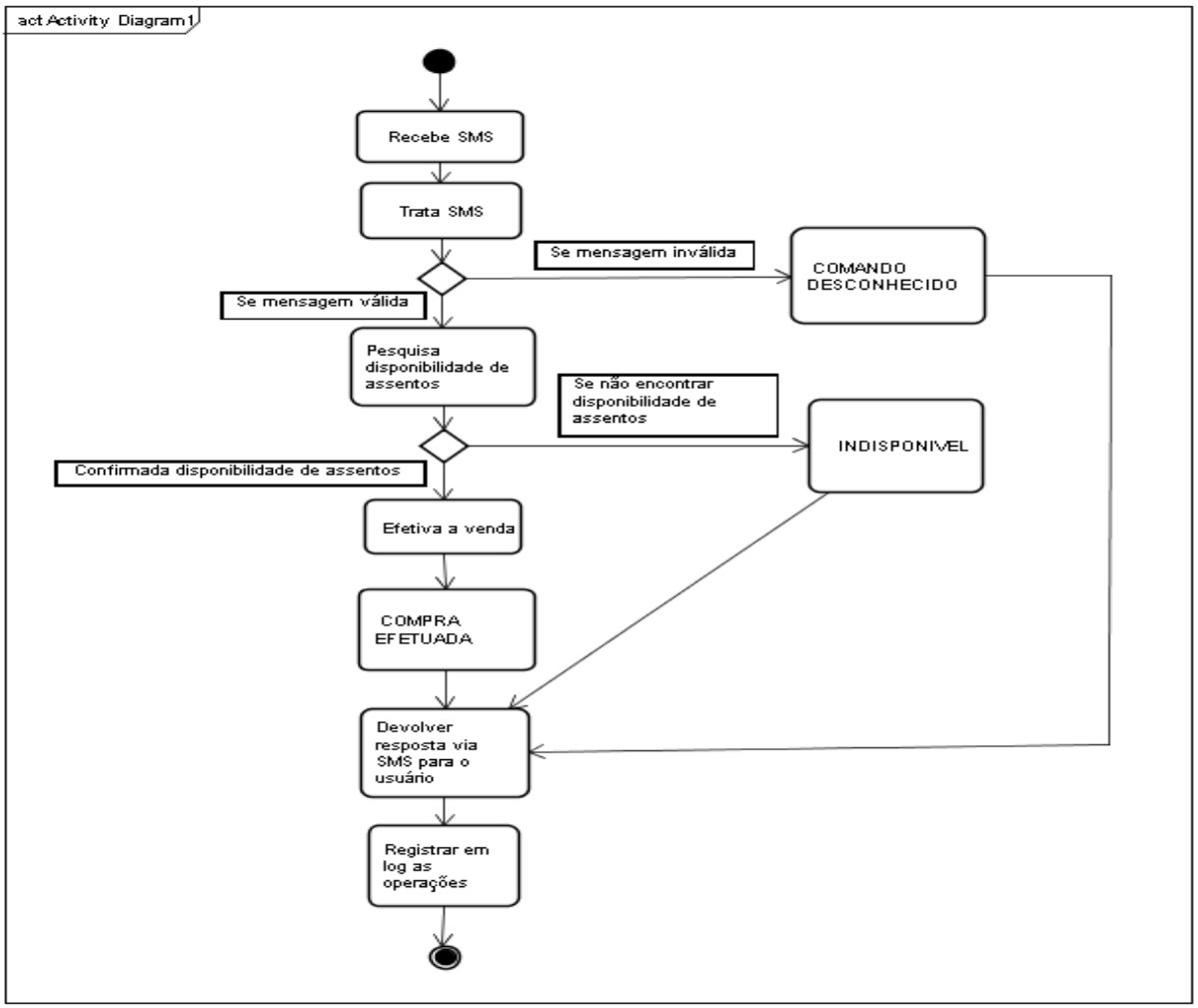
APÊNDICE V – DIAGRAMAS

A Linguagem de Modelagem Unificada (UML) surgiu como a notação diagramática padrão para a modelagem orientada a objetos. É uma linguagem para especificar, visualizar, construir e documentar os artefatos de sistema de software, bem como modelar os negócios.

O diagrama de caso de uso mostra a perspectiva do usuário e modela as interações do usuário com o sistema. O objetivo é demonstrar como o sistema se comporta e o que ele oferece para o usuário.



O diagrama de atividade preocupa descrever os passos percorridos para a conclusão de uma atividade específica. Concentra-se na representação do fluxo de controle de uma atividade.



O diagrama de classes modela classes que pertencem ao domínio principal do problema.