UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

PROTÓTIPO DE SISTEMA PARA INTEGRAÇÃO DE NOTA FISCAL ELETRÔNICA COM APLICAÇÕES B2B ON-LINE UTILIZANDO WEBSERVICES

RICARDO MOMM

BLUMENAU
2008

2008/2-21

RICARDO MOMM

PROTÓTIPO DE SISTEMA PARA INTEGRAÇÃO DE NOTA FISCAL ELETRÔNICA COM APLICAÇÕES B2B ON-LINE UTILIZANDO WEBSERVICES

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciências da Computação — Bacharelado.

Prof. Marcel Hugo, M.Eng - Orientador

BLUMENAU
2008

2008/2-21

PROTÓTIPO DE SISTEMA PARA INTEGRAÇÃO DE NOTA FISCAL ELETRÔNICA COM APLICAÇÕES B2B ON-LINE UTILIZANDO WEBSERVICES

Por

RICARDO MOMM

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

	_____
Presidente:	Prof. Marcel Hugo, M.Eng – Orientador, FURB
	_____
Membro:	Prof. Alexander R. Valdameri, Mestre – FURB
	_____
Membro:	Prof. Everaldo Artur Grahl, Mestre – FURB

Blumenau, 11 de fevereiro de 2009

Dedico este trabalho a todos os meus familiares e amigos, especialmente aqueles que me ajudaram diretamente na realização deste.

AGRADECIMENTOS

A Deus, pelo seu imenso amor e graça.

À minha família que sempre me incentivou.

À minha namorada pela paciência e compreensão.

Ao meu orientador que me guiou no desenvolvimento deste trabalho.

Aos meus colegas de empresa que me agüentaram em momentos de estresse.

A mente que se abre para uma nova idéia jamais voltará ao seu tamanho original.

Albert Einstein

RESUMO

O objetivo deste trabalho é construir um protótipo de sistema para permitir que aplicações *business-to-business* (B2B) *on-line* possam fazer a emissão de Notas Fiscais Eletrônicas (NF-E). O protótipo demonstra também a utilização do padrão de desenvolvimento *Model View-Presenter* (MVP) e é desenvolvido utilizando a linguagem de programação *C#.Net*. O protótipo tem a finalidade de servir como componente para adicionar a possibilidade de emitir NF-Es e abstrair a integração entre aplicação B2B e o sistema da SEcretaria da FAZenda (SEFAZ). A NF-E é um documento exclusivamente eletrônico que tem por objetivo substituir os modelos 1 e 1A das notas fiscais em papel e a opção da empresa em adotar a NF-E implica na proibição da emissão destes modelos em papel. Com este empecilho a SEFAZ criou o Documento Auxiliar da Nota Fiscal Eletrônica (DANFE) que é uma versão para impressão da NF-E sem valor fiscal. O protótipo possui uma interface *web* para consulta das requisições de emissão da NF-E, um histórico de NF-Es emitidas e a impressão do DANFE. A implementação do protótipo serviu para entender o mecanismo proposto pela SEFAZ para emissão de NF-E, as boas práticas para servir e consumir *webservices* e uma aplicação prática do padrão de desenvolvimento MVP.

Palavras-chave: Nota fiscal eletrônica. *Webservices*. *Model view-presenter*.

ABSTRACT

This work objective is to build a prototype of a system to allow on-line business-to-business (B2B) applications to use Nota Fiscal Eletrônica (NF-e). The prototype demonstrates the usage of the Model View-Presenter (MVP) design pattern and his development is using the C#.Net programming language. The finality of the prototype is to serve has an intermediate and an abstract component to allow B2B applications to use Nota Fiscal Eletrônica integrated with the SEcretaria da FAZenda (SEFAZ) system. The NF-e is a document exclusively electronic and has the objective to replace the 1 and 1A model of paper invoices. The option for a company to adopt the NF-e implies in the prohibition to emit model 1 and 1A paper invoices. With this obstacle the SEFAZ has created the Documento Auxiliar da Nota Fiscal Eletrônica (DANFE) that is the printed version of NF-e without fiscal value. The prototype has a web interface to search in an historic of issued NF-e and to print DANFEs. The implementation of the system served to understand the SEFAZ proposed mechanism for issue NF-e, the best practices to serve and consume webservices and a practice application of the MVP design pattern.

Key-words: Nota fiscal eletrônica. Webservices. Model view-presenter.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

ASP – *Application Service Provider*

B2B – *Business-to-Business*

CNPJ – Cadastro Nacional de Pessoa Jurídica

DANFE – Documento Auxiliar da Nota Fiscal Eletrônica

DLL – *Dynamic Link Library*

EA – *Entrerprise Architect*

HTML – *HyperText Markup Language*

HTTP – *HyperText Transfer Protocol*

HTTPS – *HyperText Transfer Protocol Secure*

ICMS – Imposto sobre Circulação de Mercadorias e Serviços

ICP-BR – Infra-estrutura de Chaves Públicas Brasileira

IDE – *Integrated Development Environment*

IPI – Imposto sobre Produtos Industrializados

MER – Modelo Entidade-Relacional

MVP – *Model View-Presenter*

NF – Nota Fiscal

NF-E – Nota Fiscal Eletrônica

ORM – *Object-Relational Mapping*

PDF – *Portable Document Format*

RF – Requisitos Funcionais

RFB – Receita Federal do Brasil

RNF – Requisitos Não Funcionais

SEFAZ – SEcretaria da FAZenda

SOAP – *Simple Object Access Protocol*

SPC – Serviço de Proteção ao Crédito

SPED – Sistema Público de Escrituração Digital

SUFRAMA – SUperintendência da zona FRAnca de MAnaus

SSL – *Secure Socket Layer*

UML – *Unified Modeling Language*

URI – *Unified Resource Identifier*

XML – *eXtensible Markup Language*

WSDL – *WebService Description Language*

SUMÁRIO

1 INTRODUÇÃO

O desafio atual do governo, no que se refere à administração tributária, é adaptar-se aos avanços tecnológicos e à informatização dos comércios e transações entre contribuintes. O número de transações efetuadas e o montante de recursos movimentados crescem em um ritmo intenso em paralelo com os custos do Estado em detectar e prevenir a sonegação de impostos (SECRETARIA DE ESTADO DA FAZENDA, 2004).

O projeto da NF-E é coordenado pelo Encontro Nacional dos Coordenadores e Administradores Tributários Estaduais e é o substituto da Nota Fiscal (NF) formatada em papel utilizada atualmente. As NFs geram um custo elevado para os contribuintes, pois seu arquivamento demanda muitos recursos e é importante para históricos, auditorias, contabilidade, entre outros. A NF-E por ser em meio digital visa resolver este problema, além de diminuir a burocracia e facilitar o controle e a fiscalização tributária.

A NF-E é um documento digital elaborado no padrão *eXtensible Markup Language* (XML) e deve ser enviado para o *site* da SEFAZ estadual assinado digitalmente para garantir a autenticidade. Atualmente a NF-E engloba somente os modelos 1 e 1-A utilizados em transações com mercadorias e somente entre empresas.

Quando uma empresa se credencia para usar a NF-E, ela está impedida de fazer a impressão da mesma em papel. Para resolver este problema, a SEFAZ estabeleceu um arquivo de retorno para cada NF-E emitida, o Documento Auxiliar da Nota Fiscal Eletrônica (DANFE). O DANFE é a representação em papel da NF-E, possuindo informações da NF-E e um código de barras para autenticação.

A SEFAZ escolheu os *webservices* como padrão para disponibilização do sistema da NF-E. Os *webservices* são uma tecnologia simples de implementar e de integrar pois utilizam padrões abertos como o XML, o *Simple Object Access Protocol* (SOAP) e rodam sobre o protocolo *HyperText Transfer Protocol* (HTTP). Segundo Borges Júnior (2005, p. 1), "Os *webservices* representam um fragmento de informação que pode ser acessado por qualquer um, em qualquer lugar, utilizando qualquer tipo de dispositivo."

O protótipo desenvolvido neste trabalho também contém *webservices* para fornecer uma integração ao sistema *Business-to-Business* (B2B) on-line com as regras de negócios. O foco de integração são sistemas B2B on-line cuja operação de venda é efetuada via internet. A disponibilização do protótipo é feita utilizando a modalidade *Application Service Provider* (ASP), pois com toda informatização no setor fiscal, as empresas têm que investir pesado para

poderem se adaptar e utilizar estes novos serviços. Os investimentos vão desde programadores para atualizar o sistema até a aquisição de servidores para armazenar dados e rodar o sistema. O ASP é uma modalidade na qual o sistema é alugado barateando o custo para a empresa contratante uma vez que toda a estrutura será terceirizada.

A configuração vai ser feita de acordo com as necessidades da empresa contratante através uma interface *web* implementada com *HyperText Markup Language* (HTML) e C#.Net utilizando um banco de dados SQL Server Express 2005. Esta interface vai ser acessada através de autenticação com nome de usuário, senha e uma conexão segura utilizando *Secure Socket Layers* (SSL). Na configuração a empresa vai definir parâmetros e variáveis para o funcionamento do protótipo.

Utilizando destas tecnologias, o principal objetivo do trabalho é fornecer um serviço para emissão da NF-E centralizando as regras de negócio, centralizando a comunicação com a SEFAZ, disponibilizando uma interface para envio dos dados e por fim fornecendo uma solução barata e robusta.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é desenvolver um protótipo baseado em *webservices* com configurações personalizadas utilizando o paradigma ASP para fazer com que sistemas B2B on-line possam emitir a NF-E.

Os objetivos específicos do trabalho são:

a) desenvolver interfaces para envio de dados pelos sistemas B2B on-line através dos *webservices*;
b) desenvolver uma biblioteca no formato de *Dynamic Link Library* (DLL) para centralizar as regras de negócio;
c) desenvolver uma interface web para configurações do sistema, cadastro de contas de autenticação e consulta do DANFE;
d) fornecer uma forma segura de comunicação com os *webservices* e com a interface web através de SSL e autenticação com criptografia;
e) utilizar o paradigma ASP para fornecer o serviço;
f) possibilitar a exportação do DANFE em *Portable Document Format* (PDF) através da interface web.

1.2 JUSTIFICATIVA

Com a popularização dos sistemas de comércio eletrônico ficou muito fácil para uma empresa possuir um sistema B2B on-line, cuja venda é feita através da internet. O problema dos sistemas B2B on-line comercializados é que até então não existia a emissão de NF por meio eletrônico e, portanto não havia um suporte para esta função. Os sistemas B2B on-line por sua maioria fornecem as empresas somente à função de emissão de pedidos com dados simples como informações básicas do cliente, código e descrição do produto assim como informações de entrega.

A adoção das regras para emissão da NF-E é um processo custoso e demorado e a proposta deste protótipo é oferecer uma alternativa barata para solucionar estes problemas. Uma das soluções é fornecer uma interface mais simples via *webservice* para que o pedido do sistema B2B on-line seja enviado diretamente para o protótipo. Com os dados do pedido no protótipo o emitente pode complementar as informações através da interface *web* e então converter o pedido em uma NF-E para enviar a SEFAZ posteriormente.

O protótipo permite que o sistema B2B on-line abstraia a implementação das regras para emissão da NF-E e fornece uma estrutura automatizada para controlar a emissão destas. Por tratar-se de um protótipo que roda no modelo ASP também poupa o emitente de adquirir novos equipamentos para hospedar a solução e de adquirir mão de obra para atualizar seus legados.

1.3 ESTRUTURA DO TRABALHO

O trabalho está organizado em quatro capítulos. No capítulo 2 são descritas as principais tecnologias utilizadas no trabalho assim como uma análise das principais ferramentas para emissão de NF-E existentes no mercado. No capítulo 3 é descrito de forma detalhada toda a especificação do protótipo contendo requisitos, diagramas e demais informações relevantes ao trabalho. Ainda no capítulo 3 é detalhada a implementação dos componentes do protótipo bem como a sua operacionalidade. No último capitulo são apresentados os resultados obtidos com o protótipo, as conclusões e sugestões para extensão do protótipo em trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Nas seções seguintes são detalhadas as tecnologias essenciais para o entendimento deste trabalho. Primeiramente é explicada a definição da NF-E, seus principais objetivos e é apresentado o modelo operacional da NF-E. Na seqüência é feita uma breve explicação do DANFE e um conceito de *webservices* focando nas suas principais utilizações e características. Em seguida são mostrados os conceitos de B2B e apresentadas às tecnologias de segurança que serão utilizadas no trabalho. Por final são detalhadas algumas ferramentas similares e trabalhos correlatos.

2.1 NOTA FISCAL ELETRÔNICA

> É um documento emitido e armazenado eletronicamente, de existência apenas digital, com o intuito de documentar uma operação de circulação de mercadorias ou prestação de serviços ocorrida entre as partes, cuja validade jurídica é garantida pela assinatura digital do emitente e recepção, pelo fisco, antes da ocorrência do Fato Gerador. (ENCONTRO NACIONAL DOS COORDENADORES E ADMINISTRADORES TRIBUTÁRIOS ESTADUAIS, 2005).

De acordo com a Receita Federal (2005), um dos objetivos da NF-E é "facilitar a vida do contribuinte e as atividades de fiscalização sobre operações e prestações tributadas pelo Imposto sobre Circulação de Mercadorias e Serviços (ICMS) e pelo Imposto sobre Produtos Industrializados (IPI)."

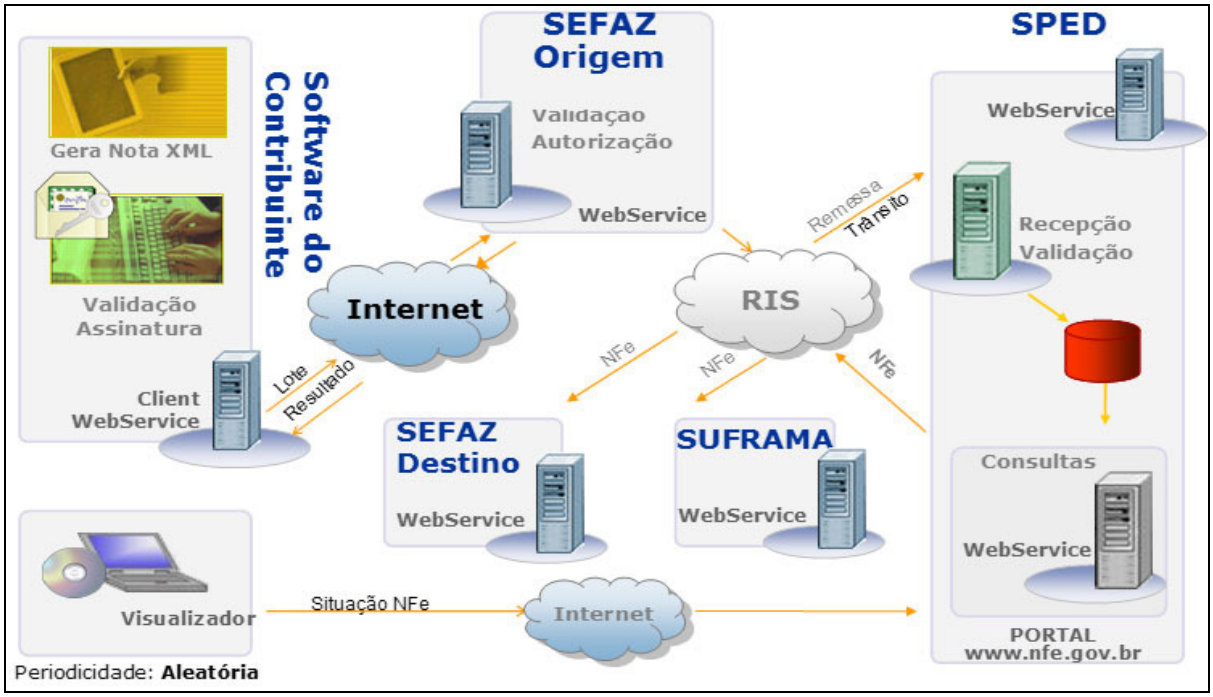
Segundo uma pesquisa com várias empresas efetuada pelo Conselho Privado da Nota Fiscal Eletrônica do Brasil, 24% dos entrevistados concordou que a redução de custos com papéis e impressão é o principal benefício que a NF-E irá trazer. Esta pesquisa apontou ainda, que 80% das empresas entrevistadas têm interesse em aderir ao projeto da NF-E.

Com a NF-E, a Receita Federal do Brasil (RFB) espera diminuir o número de fraudes e sonegação de impostos, pois, este método facilita a auditoria, sendo que todas as NF-Es estarão nos servidores da própria RFB.

Como medida de segurança, toda e qualquer NF-E deve ser assinada digitalmente através de um certificado digital adquirido pela empresa participante seguindo as normas da Infra-estrutura de Chaves Públicas BRasileira (ICP-BR). Após a assinatura digital a NF-E é enviada à SEFAZ de origem, que no caso é a SEFAZ do estado no qual a empresa é

autorizada a emitir NF-E. A SEFAZ de origem faz a validação do número da NF-E, entidade emitente, entre outros, e repassa para o Sistema Público de Escrituração Digital (SPED) que armazena a NF-E nos servidores da RFB.

Se a NF-E passar nas validações, o SPED a envia para a SUperintendência da zona FRAnca de MAnaus (SUFRAMA) e para a SEFAZ de destino, que é a pertencente ao estado de destino das mercadorias contidas na NF-E. Caso ela não venha a passar nas validações, uma resposta de erro é enviada ao software do contribuinte. O funcionamento da NF-E pode ser observado conforme a Figura 1.



Fonte: Freitas (2006, p. 9).

Figura 1 – Modelo operacional da NF-E

2.2 DANFE

> O DANFE não é uma nota fiscal, nem substitui uma nota fiscal, servindo apenas como instrumento auxiliar para consulta da NF-e, pois contém a chave de acesso da NF-e, que permite ao detentor desse documento confirmar a efetiva existência da NF-e através do Ambiente Nacional (RFB) ou site da SEFAZ na Internet. (SECRETARIA DE ESTADO DA FAZENDA, 2004).

O DANFE será a representação em papel para ser utilizado no transporte de mercadorias. Além de conter informações da NF-E, ele ainda possui um código de barras para autenticação e a chave para consulta da NF-E correspondente no *site* da SEFAZ. Quando o

uma mercadoria chega a seu destino ou passa por um posto fiscal a passagem da mesma é registrada evitando o trânsito da mesma DANFE duas vezes.

O DANFE, assim como a NF-E, também é um arquivo XML e será armazenado em um servidor para posteriormente a empresa que contratar o serviço, ter acesso à impressão da mesma independente de localidade. O único requisito para acessar o DANFE é um computador com Internet. O formato do DANFE impresso é ilustrado na Figura 2.

Fonte: Freitas (2006, p.10).

Figura 2 – DANFE impresso pela Souza Cruz para testes

2.3 *WEBSERVICE*

Webservice é um sistema de software identificado por uma *Unified Resource Identifier* (URI), na qual sua interface e métodos são definidos e descritos usando XML. Sua definição pode ser encontrada por outros sistemas e estes sistemas podem então, interagir com o *webservice* trocando mensagens em XML utilizando a Internet (WORLD WIDE WEB CONSORTIUM, 1994). Suas principais funcionalidades são:

a) participar em transações B2B;
b) expor funcionalidades do sistema aos clientes;
c) integrar linguagens de programação e plataformas heterogêneas;
d) prover uma plataforma simplificada para desenvolvimento de produtos.

Mclaughlin (2001) diz que os *webservices* são sinônimos de interoperabilidade, pois usam as mais variadas tecnologias da web e mesmo assim possuem uma grande abstração para quem os utiliza. Os *webservices* encapsulam toda a sua definição para saber quais métodos chamar, quais parâmetros passar e quais respostas receber utilizando o *WebService Definition Language* (WSDL) para a descrição dos seus serviços.

2.4 B2B

O B2B é uma sigla atual criada para representar a fatia de mercado do *e-Business* voltada para o comércio entre empresas. O B2B é a modalidade que mais cresce na internet nos últimos anos. Segundo Franco Jr. (2005, p. 33) a dimensão econômica mundial do B2B em 2004 era de 7,29 trilhões de dólares por ano, sendo este valor comparável ao PIB dos Estados Unidos da América.

O B2B, embora baseado na internet, usa predominantemente recursos de extranet que tem por característica fornecer aos parceiros, distribuidores, prestadores de serviços entre outros, uma conexão remota com a rede interna da empresa, a fim de agilizar a troca de informação em ambos os sentidos.

2.5 SEGURANÇA

Conforme Kurose e Ross (2006, p. 555), o protocolo SSL é amplamente usado no comércio pela internet, sendo implementado em quase todos os navegadores populares e servidores web. O SSL será responsável pela comunicação segura entre a empresa contratante, os *webservices* e a interface *web* para configuração do sistema.

A autenticação acontece quando o navegador envia ao servidor uma solicitação utilizando o SSL. O servidor gera uma chave simétrica compartilhada e envia para o

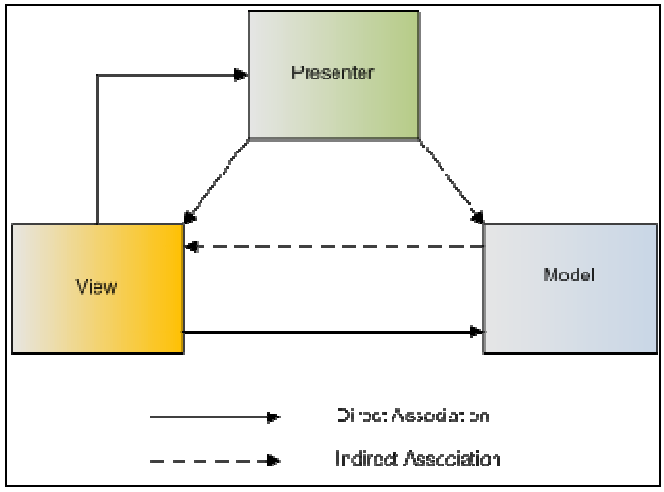
navegador que, ao receber, gera uma nova chave simétrica aleatória e criptografia com a chave pública enviada pelo servidor. A chave gerada pelo navegador é enviada ao servidor que extrai a chave simétrica aleatória e conclui a autenticação iniciando então a sessão SSL.

2.6 *MODEL VIEW-PRESENTER*

Segundo Boodhoo (2008), sem uma separação clara de responsabilidades, a camada de interface torna-se um depósito de lógica que pertence a outras camadas da aplicação. Boodhoo (2008) disse ainda que o código fonte de uma camada de interface é muito difícil de testar sem que a aplicação esteja executando, e ainda que isto seja um problema, o maior problema são os códigos duplicados espalhados entre as diferentes interfaces que prejudicam a manutenção.

Conhecendo estes problemas e sabendo que um *webform* em uma aplicação ASP.Net padrão possui a lógica da interface na própria interface, o padrão MVP foi adotado visando a separação de responsabilidades e permitindo então a testabilidade dos diferentes componentes de forma individual. Greer (2007) disse que o MVP separa as responsabilidades de manipulação de dados, coordenação da aplicação, interação do usuário e apresentação em componentes especializados.

Em Boodhoo (2008) consta que o padrão MVP torna mais fácil a fatoração da lógica para fora da camada de interface em códigos menores, reusáveis e mais fáceis de testar. O padrão MVP segue a arquitetura conforme a Figura 3.



Fonte: Greer (2008).

Figura 3 – Arquitetura do padrão MVP

Cada camada do padrão MVP tem uma responsabilidade diferente. Fowler (2006) explica que a *view* é a estrutura de controles e componentes e não contém nenhum comportamento ou evento. Fowler (2006) também disse que a *view* repassa os eventos para que o *presenter* atualize o *model* e define a ação a ser tomada para a interação do usuário.

Snyder (2007) disse que as principais diferenças entre o MVP e o *Model View Controller* (MVC) são:

a) o MVP tem a *view* menos acoplada ao *model*;
b) o MVP facilita os testes unitários pois a *view* está ligada ao *presenter* através de uma *interface*;
c) no MVP o mapeamento entre *view* e *presenter* é de um para um enquanto que no MVC o *controller* é baseado em eventos compartilhados entre as *views* e ele que determina qual *view* exibir.

2.7 TRABALHOS CORRELATOS

Existem várias ferramentas para a integração da NF-E atualmente, porém, estas ferramentas são módulos de sistemas complexos que na maioria das vezes são inacessíveis para testes. Foram escolhidas algumas ferramentas que oferecem uma breve descrição de suas funcionalidades a fim de verificar pontos fortes e fracos neste trabalho.

2.7.1 Optio Software

A ferramenta distribuída no Brasil pela Consultema Consultoria (2005) é o Optio Software (Figura 4) que possui a NF-E apenas como um módulo da ferramenta que engloba vários outros módulos, como distribuição de relatórios, nota fiscal a laser, envio automático via fax, gerenciador de impressões, entre outros.

O módulo de NF-E apresenta várias características positivas, como é o caso do suporte a vários bancos de dados tornando o sistema acessível para a maioria das empresas que não desejam migrar seus sistemas para outro banco de dados. Outra característica marcante é o suporte aos sistemas operacionais Windows, Unix, Linux e AS400.

Como ponto negativo pode ser levado em consideração o fato da aplicação fazer parte

de uma ferramenta extensa e que não pode ser utilizado separadamente forçando as empresas que querem integrar seus sistemas com a NF-E a migrar seus sistemas legados para está nova ferramenta.

Fonte: Consultema Consultoria (2005).

Figura 4 – Tela principal do Optio Software

2.7.2 NF-Express

O NF-Express é um componente *activex* apresentado pela Alfasig Consultoria e Sistemas (2007). Sua distribuição é em forma de biblioteca aonde o desenvolvedor utiliza o componente que já contém as regras para adequar uma solução existente as normas da NF-E.

Por ser um componente *activex* o NF-Express apresenta sérias limitações como a sua utilização que só pode ser feita em programas que suportam *activex*. Outra limitação é que para fazer uso do componente o cliente deve ter acesso ao código fonte do programa. Portanto a integração deve ser feita por alguém com conhecimentos avançados impedindo que o próprio cliente, através de uma interface, integre o componente.

3 DESENVOLVIMENTO DO PROTÓTIPO

O protótipo desenvolvido neste trabalho adiciona o suporte à emissão de NF-E para aplicações B2B on-line. O suporte a emissão de NF-E é adicionado através da disponibilização de *webservices* para a aplicação B2B. O protótipo ainda fornece uma área administrativa para fazer consultas as NF-Es emitidas, impressão do DANFE e configurações adicionais.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

A seguir são definidos os Requisitos Funcionais (RF) e Requisitos Não Funcionais (RNF) do protótipo:

a) gerar o arquivo XML da NF-E com base nos dados enviados pela aplicação B2B on-line (RF);
b) criar os *webservices* para a integração dos sistemas de B2B on-line (RF);
c) disponibilizar uma aplicação *web* para que o administrador responsável pela aplicação B2B configure e acesse o sistema;
d) fazer a assinatura digital do arquivo XML para posterior encaminhamento a SEFAZ (RF);
e) fazer as validações dos dados enviados pelos sistemas B2B on-line (RF);
f) enviar as NF-Es ao sistema da SEFAZ e tratar as respostas recebidas (RF);
g) armazenar as informações recebidas da SEFAZ para posterior emissão do DANFE (RF);
h) disponibilizar uma área na aplicação *web* para exportar o DANFE no formato PDF (RF);
i) implementar o protótipo utilizando o ambiente de desenvolvimento Visual Studio Professional 2008 (RNF);
j) usar a linguagem de programação C#.Net 3.0 (RNF);
k) utilizar o paradigma ASP (RNF);
l) fornecer medidas de segurança tais como SSL e criptografia (RNF).

3.2 ESPECIFICAÇÃO

Como técnica para especificação do protótipo foi escolhido o padrão *Unified Modeling Language* (UML) em conjunto com as ferramentas Enterprise Architect (EA) da Sparx Systems e Microsoft Visual Studio 2008. Para criação do Modelo Entidade-Relacional (MER) do banco de dados foi utilizada a ferramenta Microsoft Visual Studio 2008.

Nas seções seguintes serão apresentados os casos de uso, diagramas de seqüência e diagramas de classes para especificação do protótipo e MER para especificação do modelo de banco de dados.

3.2.1 Diagrama de Caso de Uso

O diagrama de caso de uso ilustra as ações disponíveis no protótipo. Os atores do protótipo são compostos por Usuário, Emitente e Aplicação B2B on-line. O ator Usuário é qualquer visitante que acessa o protótipo e ainda não possui um cadastro, o ator Emitente é um usuário cadastrado no sistema após a autenticação e a Aplicação B2B on-line é o sistema do emitente a ser integrado com o protótipo.

Na Figura 5 são apresentadas as ações do sistema, o papel de cada ator e como as ações se relacionam entre si.

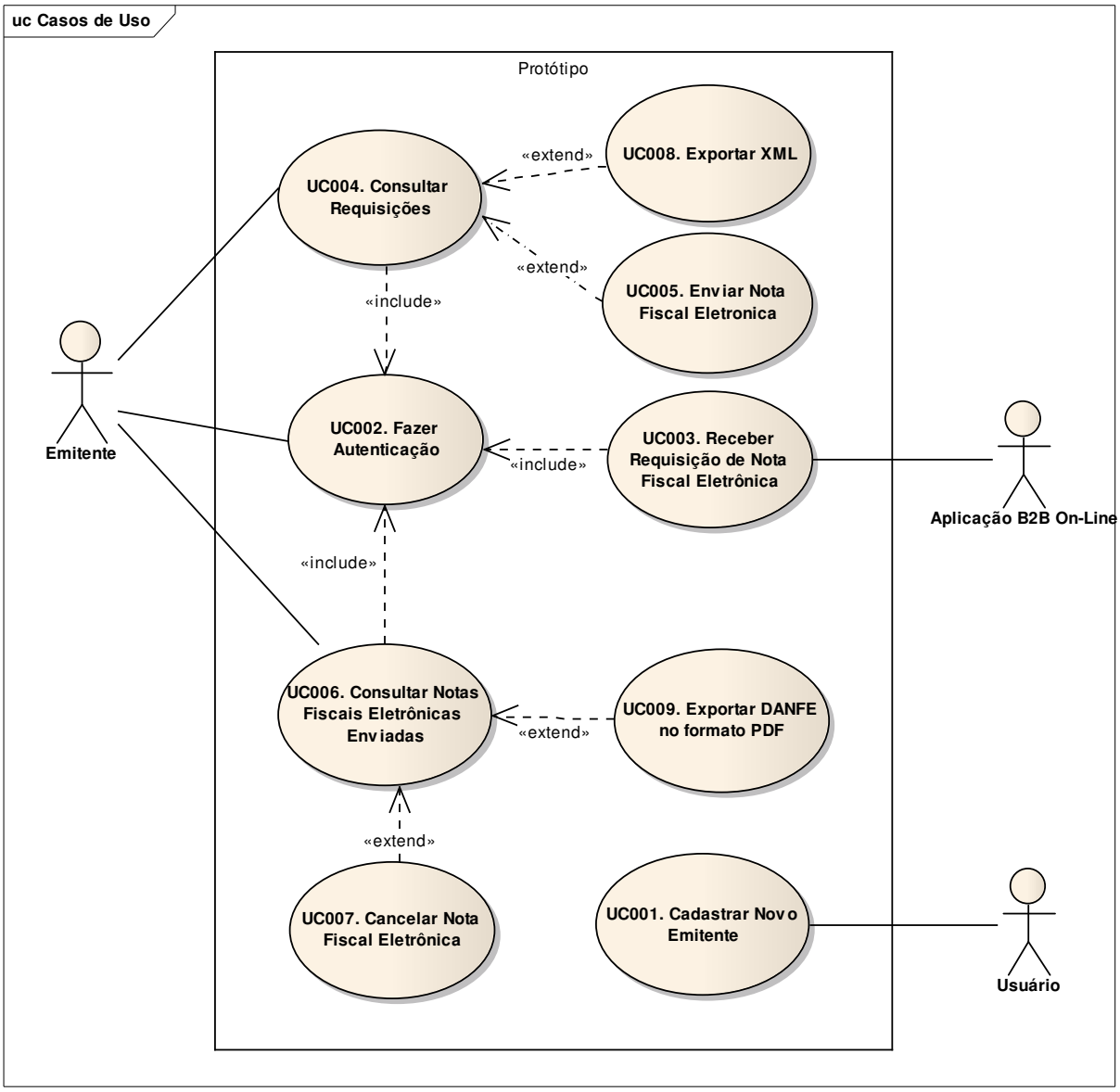


Figura 5 – Diagrama de Caso de Uso

A seguir são descritos individualmente cada um dos casos de uso apresentados no diagrama da Figura 5:

a) `cadastrar novo emitente`: para um usuário integrar sua aplicação B2B on-line ele necessita inicialmente efetuar seu cadastro como emitente. Para efetuar este cadastro o usuário precisa informar sua razão social, nome fantasia, número do Cadastro Nacional de Pessoa Jurídica (CNPJ), endereço de e-mail, nome de usuário e senha para acesso ao sistema. O número do CNPJ não pode estar cadastrado no sistema previamente. Ao final do cadastro o usuário recebe no endereço de e-mail informado os dados para acessar o sistema;

b) `fazer autenticação`: o usuário pode fazer a autenticação no sistema utilizando o usuário e senha informados durante o cadastro. A autenticação é um pré-requisito

para que o usuário possa acessar as funcionalidades de um emitente assim como fazer requisições de NF-E. A autenticação é feita sobre o protocolo *HyperText Transfer Protocol Secure* (HTTPS) e utilizando SSL para garantir a segurança da aplicação;

c) `receber requisição de nota fiscal eletrônica`: a aplicação B2B on-line do emitente faz a autenticação no protótipo via *webservice* através de HTTPS e SSL, e envia uma requisição de NF-E ao *webservice*. A aplicação do emitente deve enviar na requisição os itens que compõem a NF-E, as informações do cliente e opcionalmente as informações da transportadora. O sistema recebe a requisição e grava no banco de dados do sistema para que o emitente acesse posteriormente estas informações, visualize, complemente informações e exclua as requisições. Este caso de uso inclui o caso de uso `fazer autenticação`, pois, é necessário efetuar a autenticação para o protótipo receber requisições de NF-E;

d) `consultar requisições`: o protótipo exibe ao emitente a lista de todas as requisições que a aplicação B2B on-line enviou ao *webservice*. Na lista estão informações como data de envio, quantidade de itens, valor total e informações de contato do cliente. Através desta lista, o emitente tem a opção de selecionar uma requisição para visualizar e complementar seus detalhes, opção de selecionar uma ou mais requisições para excluir e a opção de selecionar uma ou mais requisições para enviar a como uma NF-E a SEFAZ;

e) `enviar nota fiscal eletrônica`: com as requisições selecionadas o protótipo faz a validação das informações e verifica todos os pré-requisitos para a criação de uma NF-E. Após a verificação o protótipo converte a requisição em um arquivo XML conforme o padrão estabelecido pela SEFAZ demonstrado no Apêndice A. Com o arquivo XML criado o protótipo faz a assinatura digital utilizando o certificado do emitente e envia este arquivo ao sistema da SEFAZ;

f) `consultar notas fiscais eletrônicas enviadas`: o protótipo exibe ao emitente a lista de todas as NF-Es enviadas ao sistema da SEFAZ. Nesta lista estão presentes a data da NF-E, informações do cliente, quantidade de itens, valor total e informações da transportadora. Através desta lista o emitente consegue ver a situação atual de uma NF-E no sistema da SEFAZ, efetuar o cancelamento da NF-E, efetuar a impressão e a visualização do DANFE;

g) `cancelar nota fiscal eletrônica`: o protótipo envia uma requisição de cancelamento de uma ou mais NF-Es selecionadas ao sistema da SEFAZ. O

protótipo aguarda a resposta do sistema da SEFAZ e se o cancelamento ocorreu com sucesso marca a situação das NF-Es como canceladas. Se o cancelamento acarretou em algum erro o protótipo exibe uma mensagem para o usuário;

h) `exportar XML`: o protótipo gera a requisição convertida para o formato XML proposto pela SEFAZ. Com o XML o usuário pode verificar os dados formatados da NF-E e conferir caso ocorrer algum erro de validação na SEFAZ de destino;

i) `exportar DANFE no formato PDF`: o protótipo gera uma DANFE da NF-E já enviada para a SEFAZ no formato PDF. Permite ao usuário a visualização da DANFE, impressão e envio através de e-mails e outros meios de comunicação.

3.2.2 Diagrama de Classes

Na Figura 6 é mostrado o diagrama de pacotes (*packages*) contendo todas as classes de cada *package* do protótipo e como os *packages* interagem entre si. Cada *package* representa uma DLL no protótipo e nas seções a frente eles serão descritos individualmente detalhando suas classes. Para o modelo de classes foi utilizada a ferramenta Microsoft Visual Studio 2008 que possui um recurso para sincronizar o modelo com as classes facilitando na hora de fazer alterações em um ou em outro. A descrição resumida da responsabilidade de cada pacote esta relacionada abaixo:

a) `TCC.Data`: pacote que contém todo o mapeamento objeto-relacional, fornece as classes que representam as tabelas e faz a persistência no banco de dados;

b) `TCC.Model`: pacote que contém as classes com as mesmas propriedades do pacote `TCC.Data` porém não possuem suporte a operações de banco de dados e servem somente para passagem de dados entre as camadas;

c) `TCC.Service`: pacote que contém centralizada todas as regras de negócio do sistema, também é responsável pela ligação entre o pacote `TCC.Presenter` com a base de dados;

d) `TCC.View`: pacote que contém as interfaces do MVP a ser implementadas pela camada `TCC.Web`;

e) `TCC.Presenter`: pacote que contém todas as regras de interface, é responsável pela ligação entre a interface do usuário, tratamento de eventos e comunicação com o banco de dados através do pacote `TCC.Service`;

f) `TCC.Web`: pacote que contém todas as implementações das interfaces definidas no pacote `TCC.View`.

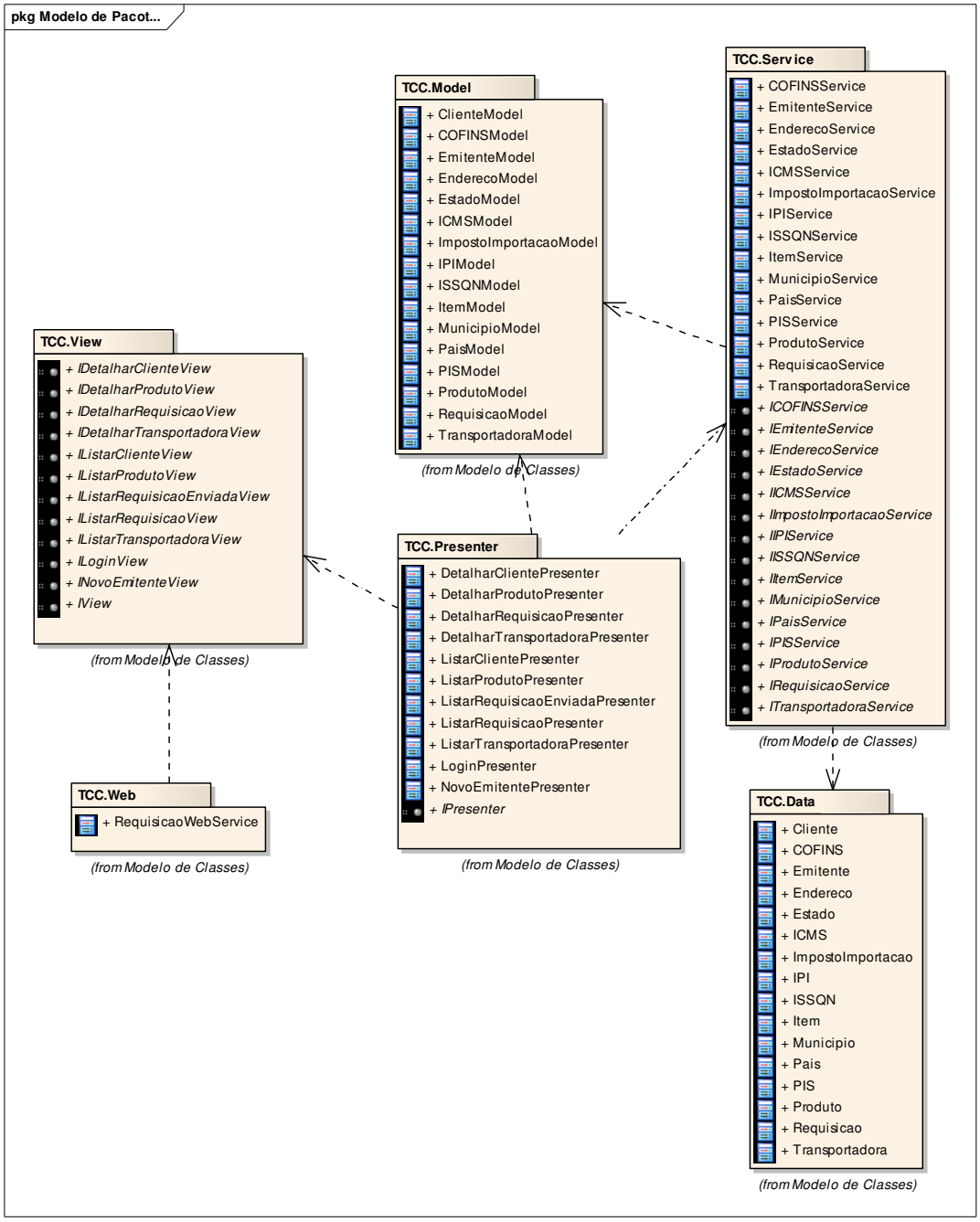


Figura 6 – Diagrama de pacotes do protótipo

3.2.2.1 Diagrama de classes: `TCC.Data`

As classes do *package* `TCC.Data`, apresentadas na Figura 7, representam a camada de acesso a dados e utilizam a tecnologia de *Object Relational Mapping* (ORM). O ORM é o

mapeamento das tabelas do banco de dados para as classes no *package* `TCC.Data` que é responsável por todas as operações que envolvem o banco de dados assim como a conexão com o mesmo. As classes que compões o *package* `TCC.Data` são:

a) `Emitente`: classe que possui todas as propriedades correspondentes às colunas da tabela `Emitente` no banco de dados;
b) `Requisicao`: classe que possui todas as propriedades correspondentes às colunas da tabela `Requisicao` no banco de dados. Esta é a classe que representa uma NF-E no protótipo;
c) `Cliente`: classe que possui todas as propriedades correspondentes às colunas da tabela `Cliente` no banco de dados;
d) `Produto`: classe que possui todas as propriedades correspondentes às colunas da tabela `Produto` no banco de dados;
e) `Endereco`: classe que possui todas as propriedades correspondentes às colunas da tabela `Endereco` no banco de dados;
f) `Pais`: classe que possui todas as propriedades correspondentes às colunas da tabela `Pais` no banco de dados;
g) `Estado`: classe que possui todas as propriedades correspondentes às colunas da tabela `Estado` no banco de dados;
h) `Municipio`: classe que possui todas as propriedades correspondentes às colunas da tabela `Municipio` no banco de dados;
i) `Transportadora`: classe que possui todas as propriedades correspondentes às colunas da tabela `Transportadora` no banco de dados;
j) `TCCEntities`: classe que gerencia a conexão com o banco de dados e é responsável por efetuar as inserções, alterações e exclusões de cada entidade;
k) `Item`: classe que possui todas as propriedades correspondentes às colunas da tabela `Item` no banco de dados;
l) `PIS`: classe que possui todas as propriedades correspondentes às colunas da tabela `PIS` no banco de dados;
m) `IPI`: classe que possui todas às colunas da tabela `IPI` no banco de dados;
n) `ICMS`: classe que contém todas as propriedades correspondentes às colunas da tabela `ICMS` no banco de dados;
o) `ImpostoImportacao`: classe que contém todas as propriedades correspondentes às colunas da tabela `ImpostoImportacao` no banco de dados;

p) `COFINS`: classe que contém todas as propriedades correspondentes às colunas da tabela `COFINS` no banco de dados;

q) `ISSQN`: classe que contém todas as propriedades correspondentes às colunas da tabela `ISSQN` no banco de dados.

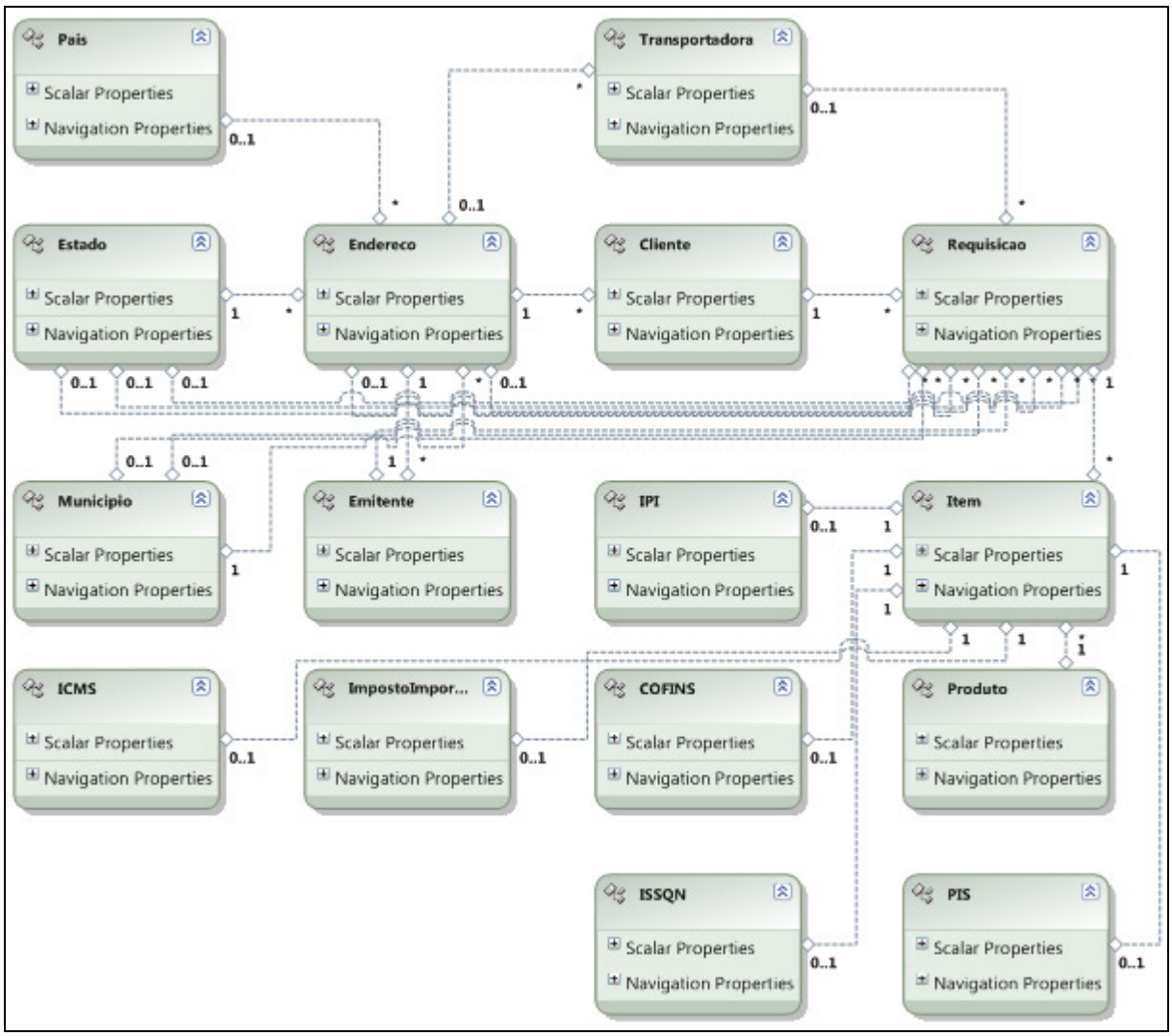


Figura 7 – Diagrama de classe gerado pelo *wizard* do Visual Studio 2008 do *package* `TCC.Data`

Para criação das tabelas foi utilizado o *wizard* do Microsoft Visual Studio 2008 a partir do arquivo que contém o *schema* da NF-E fornecido pela SEFAZ. A persistência das informações no banco de dados é abstrata para o desenvolvedor e para o usuário.

3.2.2.2 Diagrama de classes: `TCC.Model`

As classes do *package* `TCC.Model`, apresentadas na Figura 8, fazem parte da camada de apresentação e representam o *model* do modelo de desenvolvimento MVP. As classes

deste *package* são um espelho das classes do *package* `TCC.Data` porém não possuem métodos e nenhuma outra funcionalidade além de armazenar os valores de seus atributos.

O propósito das classes deste *package* é de serem trafegadas entre os *packages* `TCC.Service`, `TCC.Presenter`, `TCC.View` e `TCC.Web`. Isto se deve à separação de conceitos proposto pela programação multicamadas em que uma classe da camada de acesso a dados, implementada pelo *package* `TCC.Data`, não deve ser acessada de nenhuma outra camada a não ser da camada de negócios, implementada pelo *package* `TCC.Service`.

As classes do *package* `TCC.Model` são:

a) `EmitenteModel`: classe que possui todos os atributos da tabela de `Emitente`;
b) `RequisicaoModel`: classe que possui todos os atributos da tabela de `Requisicao`;
c) `ClienteModel`: classe que possui todos os atributos da tabela de `Cliente`;
d) `ProdutoModel`: classe que possui todos os atributos da tabela de `Produto`;
e) `EnderecoModel`: classe que possui todos os atributos da tabela de `Endereco`;
f) `PaisModel`: classe que possui todos os atributos da tabela `Pais`;
g) `EstadoModel`: classe que possui todos os atributos da tabela `Estado`;
h) `MunicipioModel`: classe que possui todos os atributos da tabela `Municipio`;
i) `TransportadoraModel`: classe que possui todos os atributos da tabela `Transportadora`;
j) `ICMSModel`: classe que possui todos os atributos da tabela `ICMS`;
k) `PISModel`: classe que possui todos os atributo da tabela `PIS`;
l) `IPIModel`: classe que possui todos os atributos da tabela `IPI`;
m) `COFINSModel`: classe que possui todos os atributos da tabela `COFINS`;
n) `ISSQNModel`: classe que possui todos os atributos da tabela `ISSQN`;
o) `ImpostoImportacaoModel`: classe que possui todos os atributos da tabela de `ImpostoImportacao`;
p) `ItemModel`: classe que possui todos os atributos da tabela de `Item`.

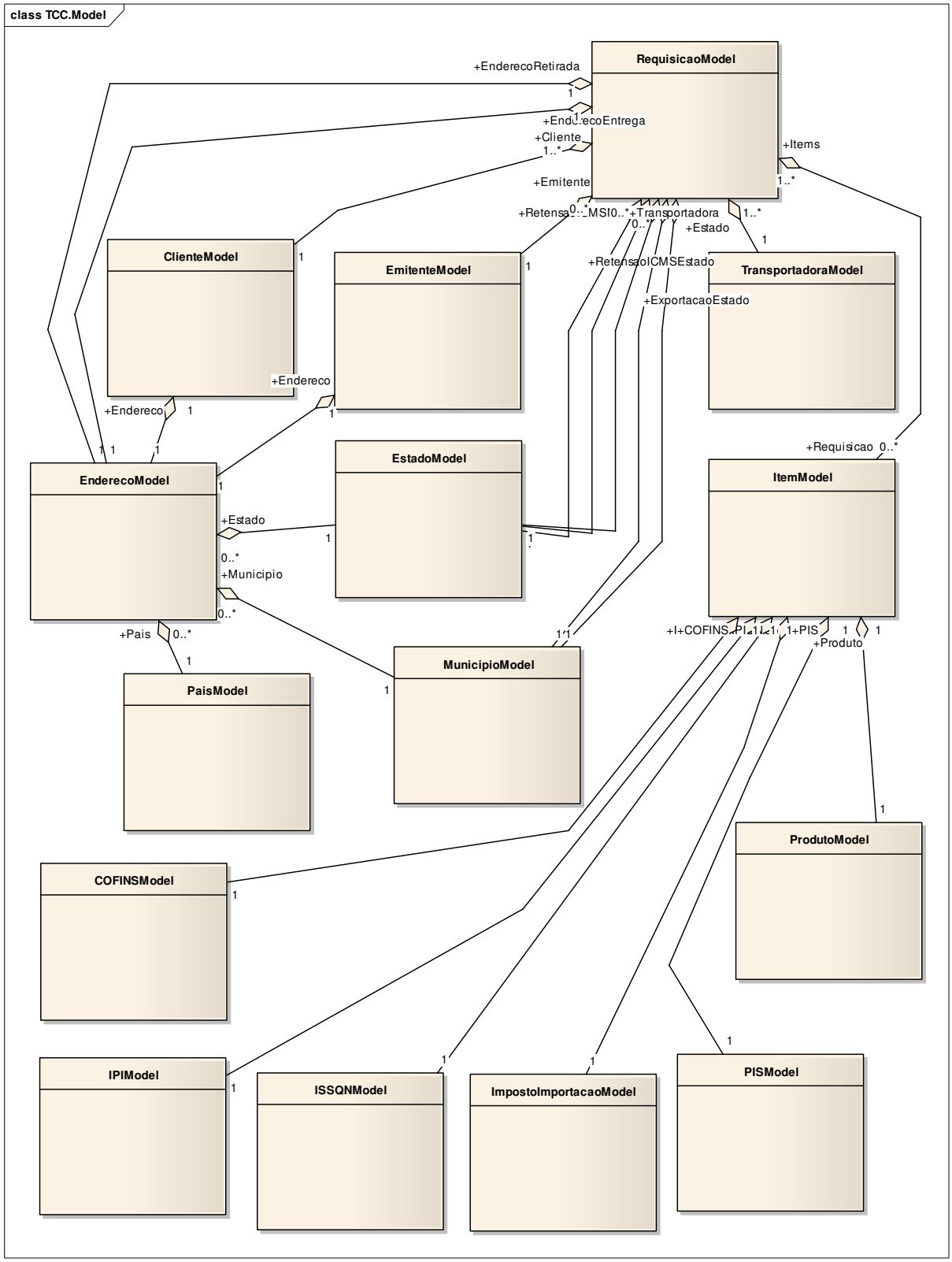


Figura 8 – Diagrama de classes do *package* `TCC.Model`

3.2.2.3 Diagrama de classes: `TCC.Service`

As classes do *package* `TCC.Service`, apresentadas na Figura 9, fazem parte da camada de negócio do protótipo e contém os métodos de consulta na camada de dados assim como os métodos para as operações de inserção, alteração e exclusão. As classes do package `TCC.Service` são:

a) `IEmitenteService`: interface que define os atributos e métodos públicos de um serviço que atende entidades do tipo `Emitente`;
b) `EmitenteService`: classe que implementa os métodos e propriedades definidos na interface `IEmitenteService`;
c) `IRequisicaoService`: interface que define os atributos e métodos públicos de um serviço que atende entidades do tipo `NF-E`;
d) `RequisicaoService`: classe que implementa os métodos e propriedades definidos na interface `IRequisicaoService`;
e) `IClienteService`: interface que define os atributos e métodos públicos de um serviço que atende a entidade do tipo `Cliente`;
f) `ClienteService`: classe que implementa os métodos e propriedades definidos na interface `IClienteService`;
g) `IProdutoService`: interface que define os atributos e métodos públicos de um serviço que atende a entidade do tipo `Produto`;
h) `ProdutoService`: classe que implementa os métodos e propriedades definidos na interface `IProducoService`;
i) `IEnderecoService`: interface que define os atributos e métodos públicos de um serviço que atende a entidade do tipo `Endereco`;
j) `EnderecoService`: classe que implementa os métodos e propriedades definidos na interface `IEnderecoService`;
k) `IPaisService`: interface que define os atributos e métodos públicos de um serviço que atende a entidade do tipo `Pais`;
l) `PaisService`: classe que implementa os métodos e propriedades definidos na interface `IPaisService`;
m) `IEstadoService`: interface que define os atributos e métodos públicos de um serviço que atende a entidade do tipo `Estado`;

n) `EstadoService`: classe que implementa os métodos e propriedades definidos na interface `IEstadoService`;

o) `IMunicipioService`: interface que define os atributos e métodos públicos de um serviço que atende a entidade do tipo `Municipio`;

p) `MunicipioService`: classe que implementa os métodos e propriedades definidos na interface `IMunicipioService`;

q) `ITransportadoraService` interface que define os atributos e métodos públicos de um serviço que atende a entidade do tipo `Transportadora`;

r) `TransportadoraService`: classe que implementa os métodos e propriedades definidos na interface `ITransportadoraService`;

s) `IItemService`: interface que define os atributos e métodos públicos de um serviço que atende a entidade do tipo `Item`;

t) `ItemService`: classe que implementa os métodos e propriedades definidos na interface `IItemService`;

u) `IIPIService`: interface que define os atributos e métodos públicos de um serviço que atende a entidade do tipo `IPI`;

v) `IPIService`: classe que implementa os métodos e propriedades definidos na interface `IIPIService`;

w) `ICOFINSService`: interface que define os atributos e métodos públicos de um serviço que atende a entidade do tipo `COFINS`;

x) `COFINSService`: classe que implementa os métodos e propriedades definidos na interface `ICofinsService`;

y) `IICMSService`: interface que define os atributos e métodos públicos de um serviço que atende a entidade do tipo `ICMS`;

z) `ICMSService`: classe que implementa os métodos e propriedades definidos na interface `IICMSService`;

aa) `IImpostoImportacaoService`: interface que define os atributos e métodos públicos de um serviço que atende a entidade do tipo `ImpostoImportacao`;

bb) `ImpostoImportacaoService`: classe que implementa os métodos e propriedades definidos na interface `IImpostoImportacaoService`;

cc) `IPISService`: interface que define os atributos e métodos públicos de um serviço que atende a entidade do tipo `PIS`;

dd) `PISService`: classe que implementa os métodos e propriedades definidos na

interface `IPISService`;

ee) `IISSQNService`: interface que define os atributos e métodos públicos de um serviço que atende a entidade do tipo `ISSQN`;

ff) `ISSQNService`: classe que implementa os métodos e propriedades definidos na interface `IISSQNService`.

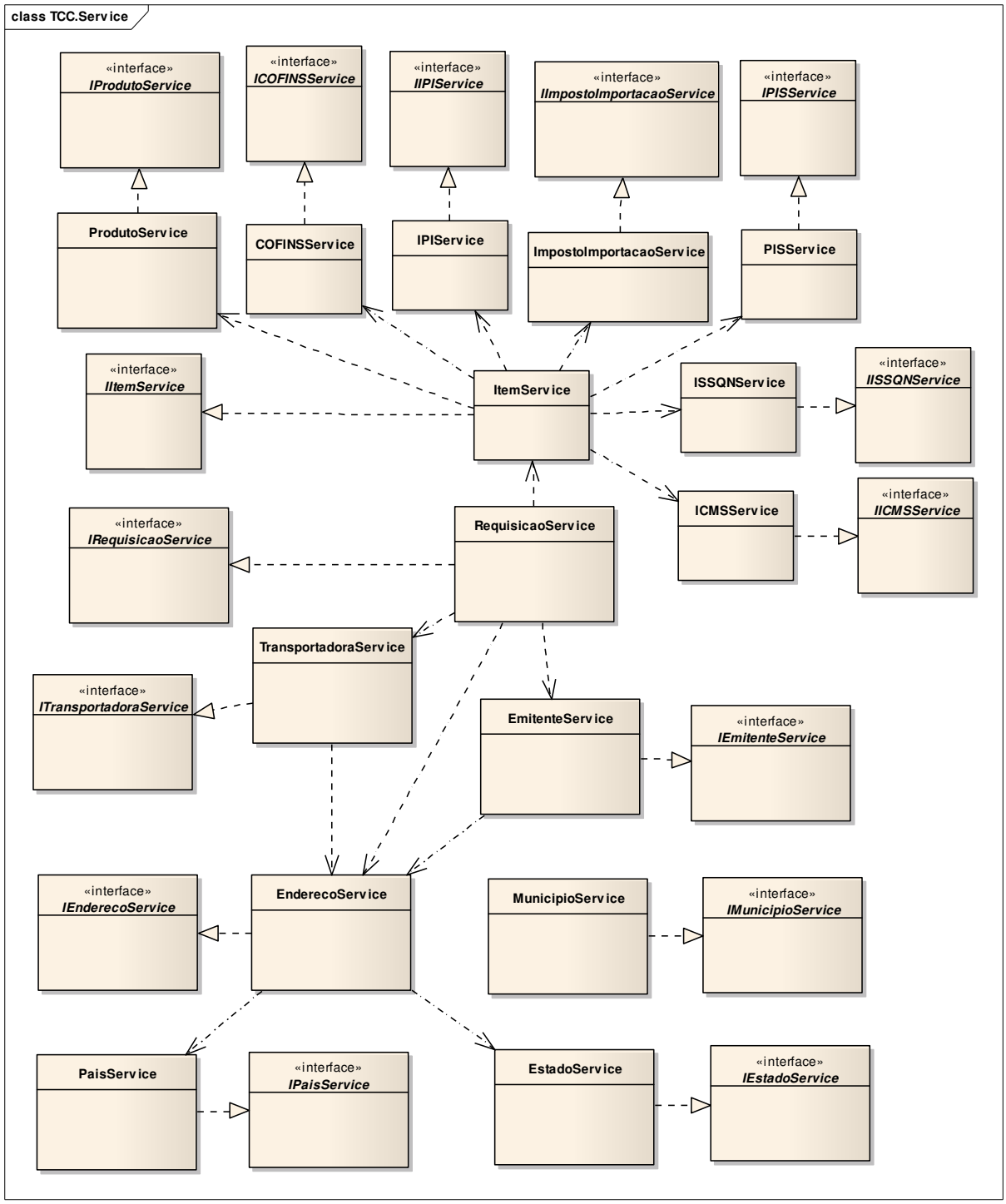


Figura 9 – Diagrama de classes do *package* `TCC.Service`

3.2.2.4 Diagrama de classes: `TCC.View`

As classes do *package* `TCC.View`, apresentadas na Figura 10, fazem parte da camada de apresentação e representam o *view* do modelo de desenvolvimento MVP. Nas classes deste *package* estão as definições das telas do protótipo e de suas operações que são:

a) `IView`: interface que define as propriedades e ações comuns a todas as telas do protótipo;
b) `ILoginView`: interface que define as propriedades e ações disponíveis na tela de login do protótipo;
c) `INovoEmitenteView`: interface que define as propriedades e ações disponíveis na tela de cadastro de novo emitente para utilização do sistema;
d) `IListarRequisicaoView`: interface que define as propriedades e ações disponíveis na tela que lista as requisições de um emitente;
e) `IDetalharRequisicaoView`: interface que define as propriedades e ações disponíveis na tela que exibe uma requisição em detalhes;
f) `IListarRequisicaoEnviadaView`: interface que define as propriedades e ações disponíveis na tela que lista as requisições convertidas em NF-E, validadas e assinadas que foram enviadas a SEFAZ;
g) `IListarClienteView`: interface que define as propriedades e ações disponíveis na tela que lista os clientes de um emitente;
h) `IDetalharClienteView`: interface que define as propriedades e ações disponíveis na tela que exibe um cliente em detalhes;
i) `IListarProdutoView`: interface que define as propriedades e ações disponíveis na tela que lista os produtos de um emitente;
j) `IDetalharProdutoView`: interface que define as propriedades e ações disponíveis na tela que exibe um produto em detalhes;
k) `IListarTransportadoraView`: interface que define as propriedades e ações disponíveis na tela que lista as transportadoras de um emitente;
l) `IDetalharTransportadoraView`: interface que define as propriedades e ações disponíveis na tela que exibe uma transportadora em detalhes.

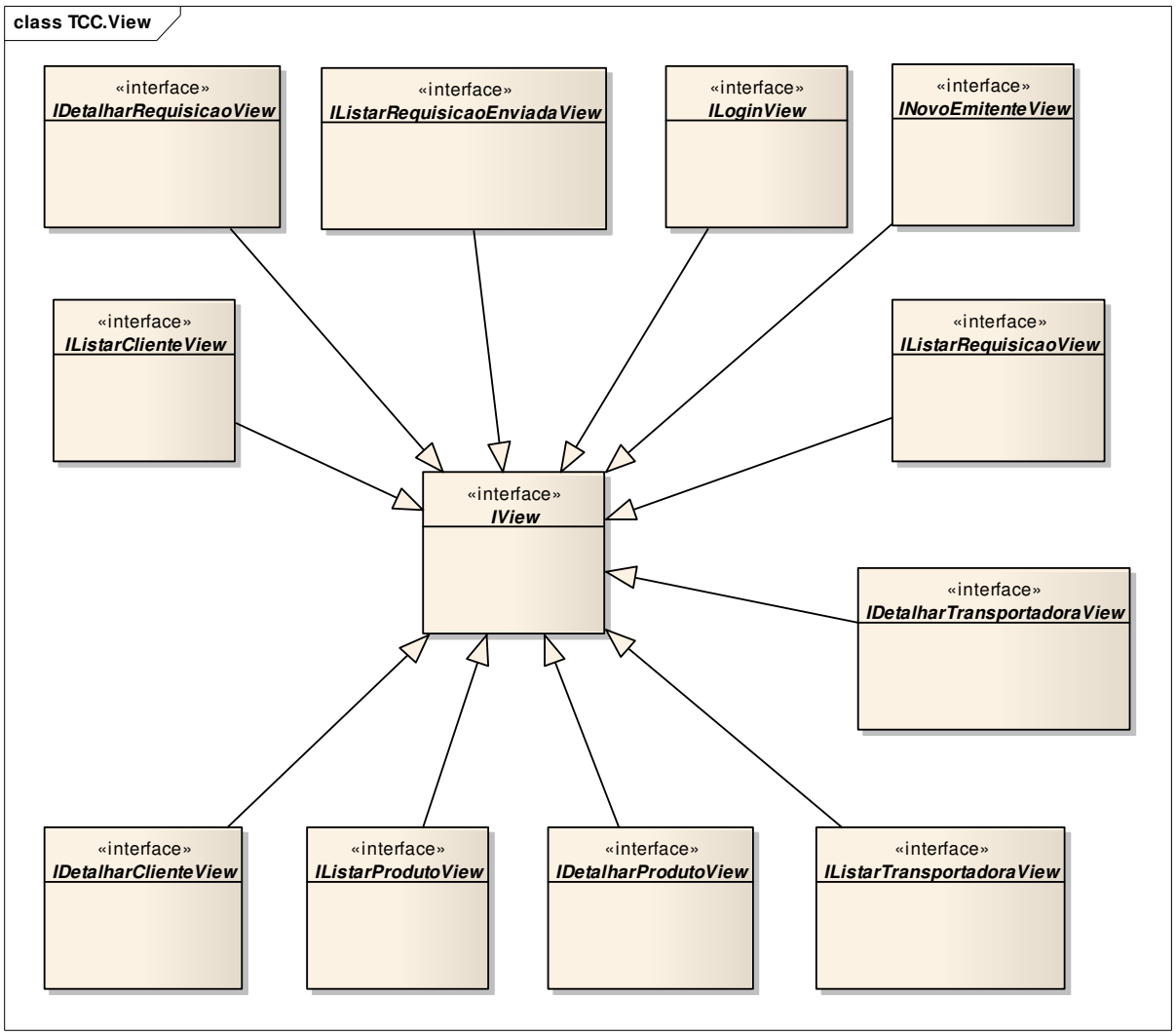


Figura 10 – Diagrama de classes do *package* `TCC.View`

3.2.2.5 Diagrama de classes: `TCC.Presenter`

As classes do *package* `TCC.Presenter`, apresentadas na Figura 11, fazem parte da camada de apresentação e representam o *presenter* do modelo de desenvolvimento MVP. Nas classes deste *package* estão regras de negócio e eventos das telas. Este *package* é responsável pela ligação entre a camada de negócio e a camada de apresentação, e é por ele que o *package* `TCC.Service` recebe dados do `TCC.View`. As classes que compões este *package* são:

a) `IPresenter`: interface que define as propriedades e ações comuns a todas as classes deste *package*;

b) `LoginPresenter`: classe responsável por tratar as regras da *view* `ILoginView` e fazer a ligação desta com a classe de negócio `IEmitenteService`;

c) `NovoEmitentePresenter`: classe responsável por tratar as regras da *view* `INovoEmitenteView` e fazer a ligação desta com a classe de negócio `IEmitenteService`;

d) `ListarRequisicoesPresenter`: classe responsável por tratar as regras da *view* `IListarRequisicoesView` e fazer a ligação desta com a classe de negócio `IRequisicaoService`;

e) `DetalharRequisicaoPresenter`: classe responsável por tratar as regras da *view* `IDetalharRequisicoesView` e fazer a ligação desta com a classe de negócio `IRequisicaoService`;

f) `ListarRequisicaoEnviadaPresenter`: classe responsável por tratar as regras da *view* `IListarRequisicaoEnviadaView` e fazer a ligação desta com a classe de negócio `IRequisicaoService`;

g) `ListarClientePresenter`: classe responsável por tratar as regras da *view* `IListarClienteView` e fazer a ligação desta com a classe de negócio `IClienteService`;

h) `DetalharClientePresenter`: classe responsável por tratar as regras da *view* `IDetalharClienteView` e fazer a ligação desta com a classe de negócio `IClienteService`;

i) `ListarProdutoPresenter`: classe responsável por tratar as regras da *view* `IListarProdutoView` e fazer a ligação desta com a classe de negócio `IProdutoService`;

j) `DetalharProdutoPresenter`: classe responsável por tratar as regras da *view* `IDetalharProdutoView` e fazer a ligação desta com a classe de negócio `IProdutoService`;

k) `ListarTransportadoraPresenter`: classe responsável por tratar as regras da *view* `IListarTransportadoraView` e fazer a ligação desta com a classe de negócio `ITransportadoraService`;

l) `DetalharTransportadoraPresenter`: classe responsável por tratar as regras da *view* `IDetalharTransportadoraView` e fazer a ligação desta com a classe de negócio `ITransportadoraService`.

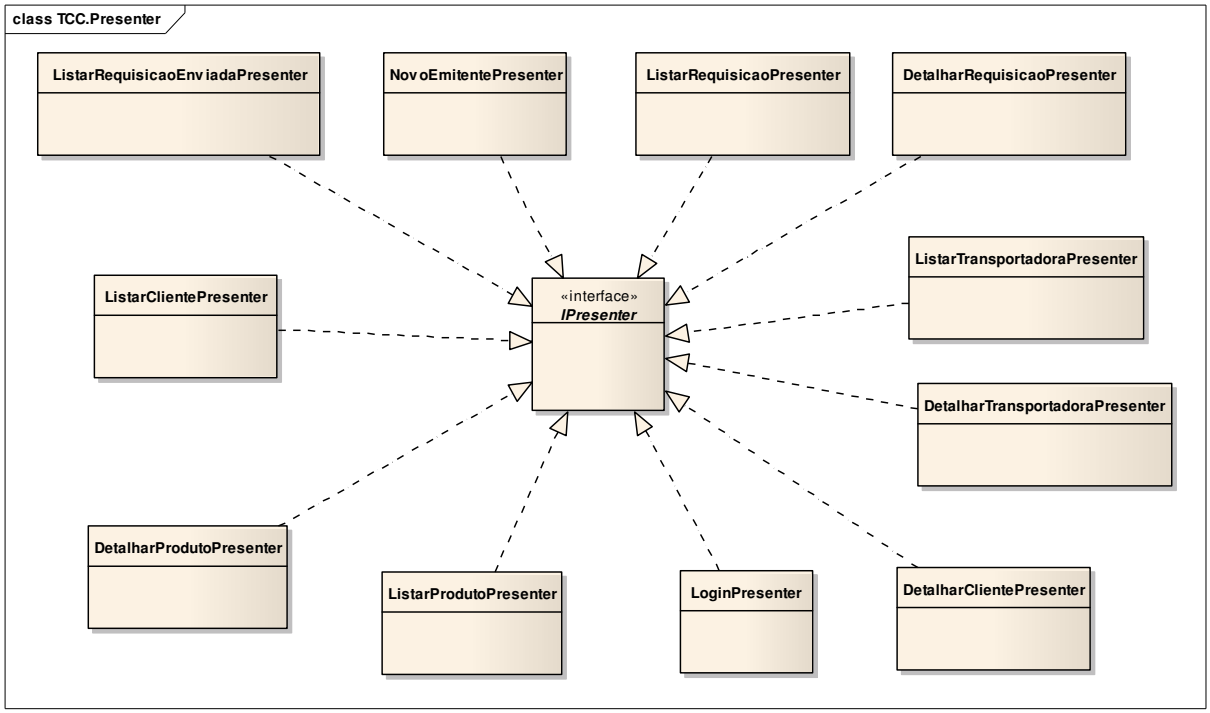


Figura 11 – Diagrama de classes do *package* `TCC.Presenter`

3.2.2.6 Diagrama de classes: `TCC.Web`

O *package* `TCC.Web` faz parte da camada de apresentação e fornece os formulários e telas para utilização do protótipo. As classes das telas implementam as *interfaces* das *views* definidas na *package* `TCC.View` conforme a Figura 12 e a principal classe deste pacote é a interface do *webservice* disponível para consumo pelo protótipo:

`RequisicaoWebService`: classe que define o *webservice* responsável por autenticar uma aplicação B2B on-line de um emitente previamente cadastrado, receber uma requisição e fornecer o status de uma requisição já enviada a SEFAZ.

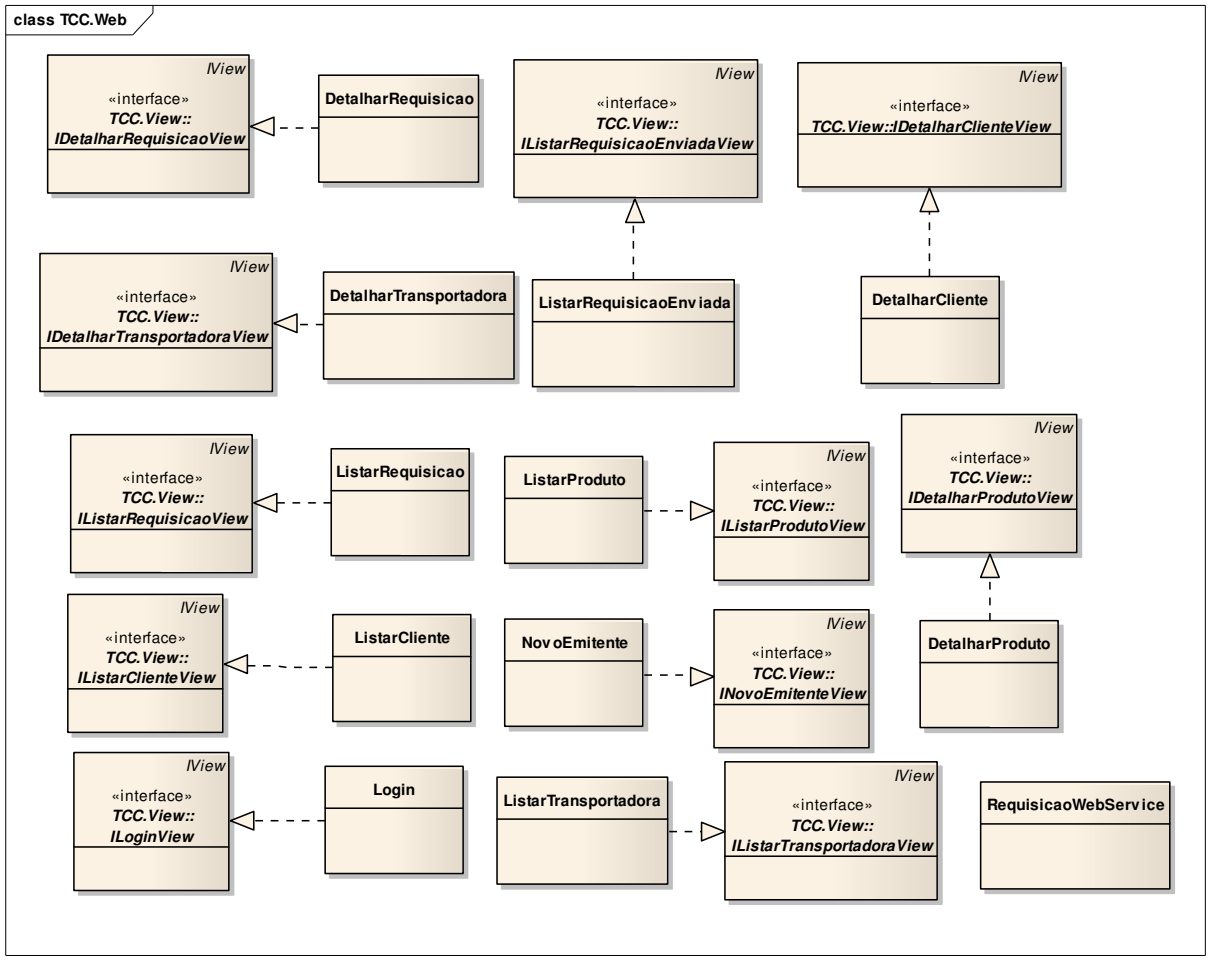


Figura 12 - Diagrama de classes do *package* `TCC.Web`

3.2.3 Diagrama de Sequência

O diagrama de sequência serve para ilustrar de uma forma mais detalhada a troca de mensagens entre objetos durante o tempo de execução de um programa. Para detalhar a utilização da troca de mensagens entre os objetos do protótipo são apresentados a seguir o diagrama de seqüência para os principais casos de uso descritos neste trabalho.

3.2.3.1 Diagrama de seqüencia: `cadastrar novo emitente`

O cadastro de um emitente é o primeiro passo para utilização do protótipo, sem o cadastro inicial não é possível definir os dados do emitente de uma NF e nem controlar as requisições e NF-Es geradas por ele. Conforme a Figura 13 o processo inicia quando um

usuário acessa o protótipo e não possui usuário e senha para fazer autenticação. Ao preencher os campos obrigatórios e enviar o formulário a classe `NovoEmitentePresenter` recebe os dados e com auxilio da classe `EmitenteService` faz as validações necessárias.

A validação é responsável por garantir a integridade do CNPJ, do e-mail e do nome de usuário escolhido pelo emitente para acessar o protótipo. Após a validação ser concluída com sucesso os dados são passados a classe `EmitenteService` que insere no banco de dados através da classe `TCCEntitites` no *package* `TCC.Data`.

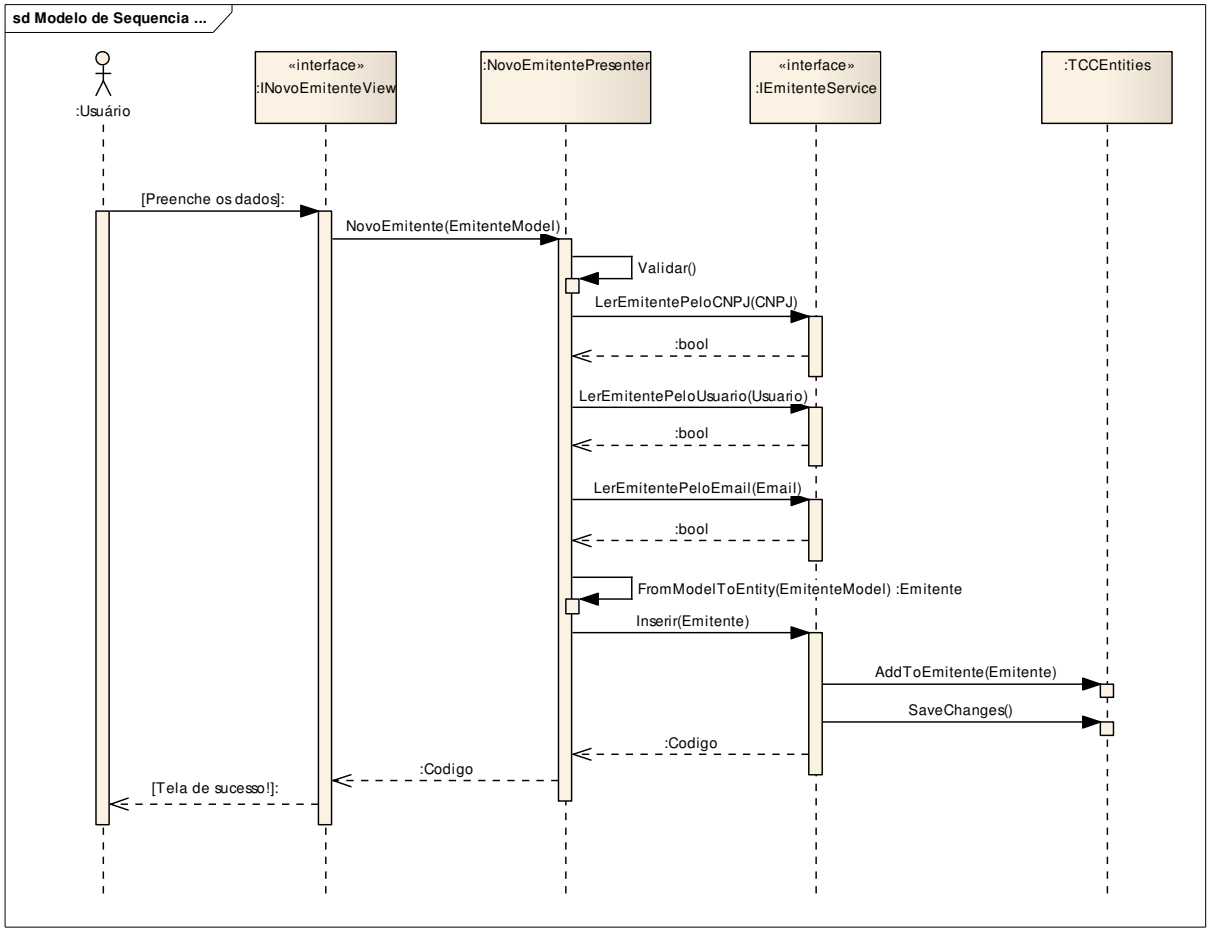


Figura 13 - Diagrama de seqüência: `cadastrar novo emitente`

3.2.3.2 Diagrama de seqüência: `fazer autenticação`

O processo de autenticação é utilizado na maioria dos casos e é o processo que assegura segurança no acesso ao protótipo. O diagrama na Figura 14 é iniciado no envio da solicitação de autenticação pelo usuário que informa um nome de usuário e uma senha. A *view* recebe a requisição e repassa a classe `LoginPresenter` que com auxílio da classe `EmitenteService` efetua a validação do usuário e senha.

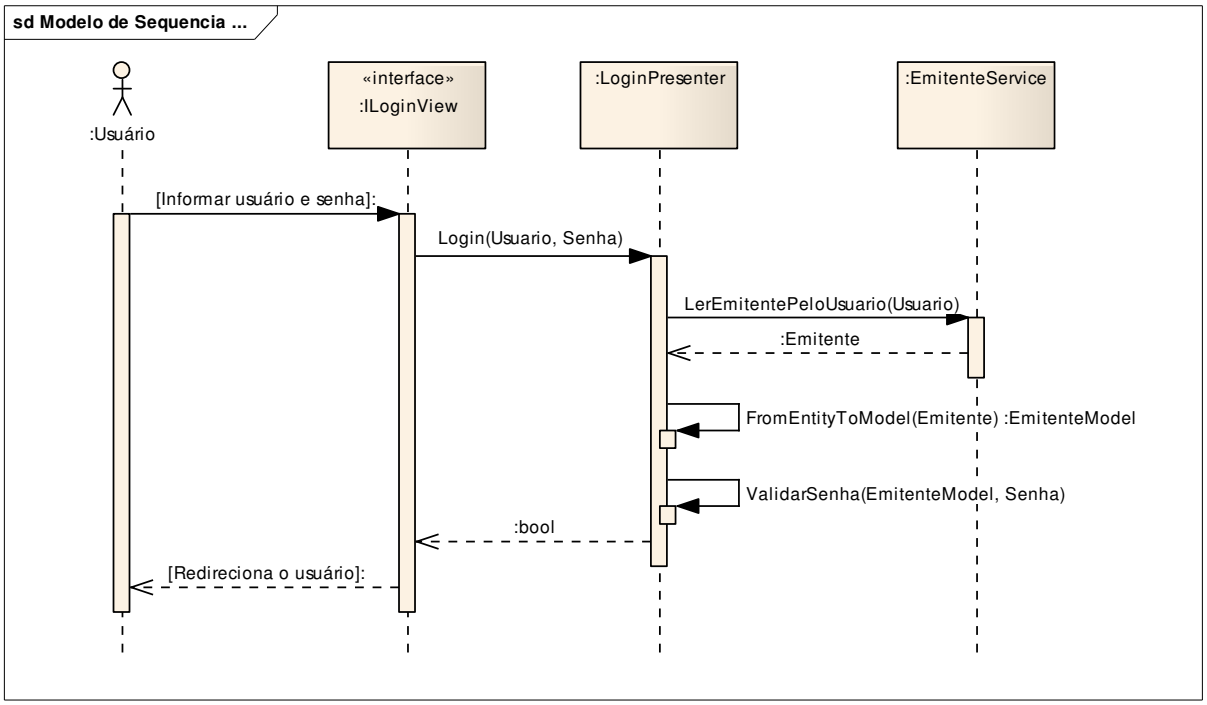


Figura 14 - Diagrama de seqüência: `fazer autenticação`

3.2.3.3 Diagrama de seqüência: `receber requisição de NF-E`

Este diagrama ilustra a interação em que o *webservice* recebe uma requisição de NF-E do sistema B2B on-line de um emitente previamente cadastrado conforme o caso de uso `receber requisição de nota fiscal eletrônica`. Na Figura 15 é exibido o diagrama ilustrando a troca de mensagens entre as classes que tem participação neste caso de uso, onde o processo é iniciado pela aplicação B2B on-line que envia um pedido ao *webservice* que valida o pedido e efetua a autenticação, permitindo ou não a inserção da nova requisição pela aplicação B2B on-line.

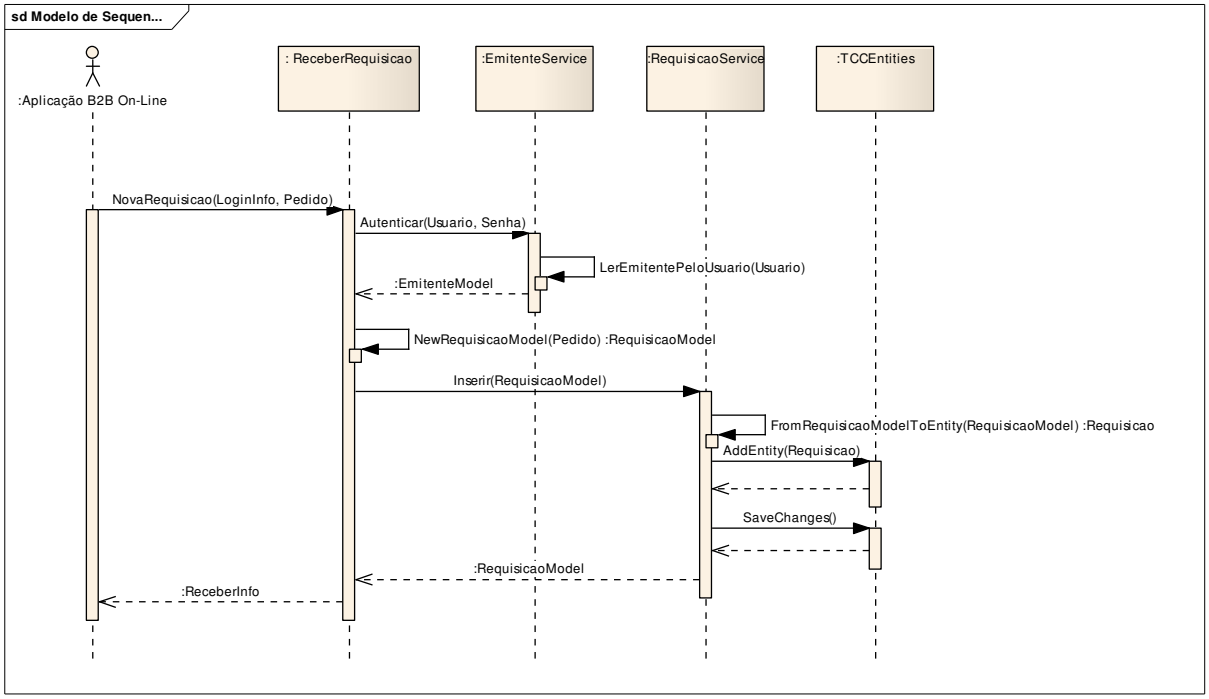


Figura 15 – Diagrama de seqüência: `receber requisição de NF-E`

Ainda na Figura 15, após a autenticação o *webservice* recebe os dados e efetua a montagem de uma entidade de `RequisicaoModel` que é enviada a classe `RequisicaoService` através das classes do *package* `TCC.Service` e `TCC.Data`. Com a entidade criada, ela é adicionada a classe `TCCEntities` e em seguida as alterações são salvas. Após a inserção dos dados o *webservice* retorna uma instância de `ReceberInfo` com o código da requisição ou com a descrição do erro, caso houver, ao sistema B2B on-line para posteriormente poder acessar o andamento da requisição.

3.2.3.4 Diagrama de seqüência: `consultar requisições`

A consulta de requisições fornece ao usuário uma lista de todas as requisições enviadas por sua aplicação B2B on-line. O diagrama da Figura 16 demonstra a seqüência de mensagens entre os objetos para efetuar a consulta de requisições. Conforme o padrão MVP a *view* recebe a requisição do usuário, repassa ao `ListarRequisicaoPresenter` que com auxilio da classe `RequisicaoService` busca as requisições no banco de dados e retorna ao usuário.

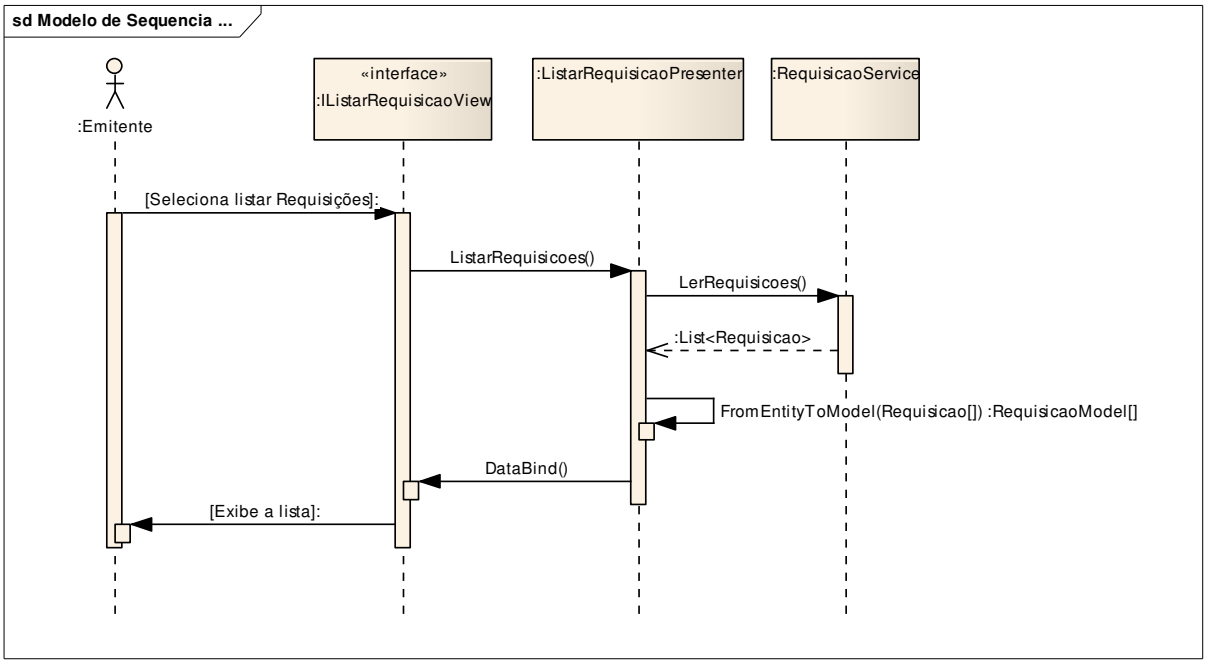


Figura 16 - Diagrama de seqüência: `consultar requisições`

3.2.3.5 Diagrama de seqüência: `enviar NF-E`

Este diagrama ilustra a interação em que o usuário solicita ao protótipo o envio de uma ou mais requisições como NF-E conforme o caso de uso `enviar nota fiscal eletrônica`. Na Figura 17 é exibido o diagrama ilustrando a troca de mensagens entre as classes que têm participação neste caso de uso, onde o processo é iniciado pela interação do usuário que seleciona previamente uma ou mais requisições a enviar.

O protótipo utiliza a classe `IRequisicaoService` para, através dela, efetuar a validação de todos os campos marcados como obrigatório pela SEFAZ, formatar a requisição em um arquivo XML e assinar digitalmente.

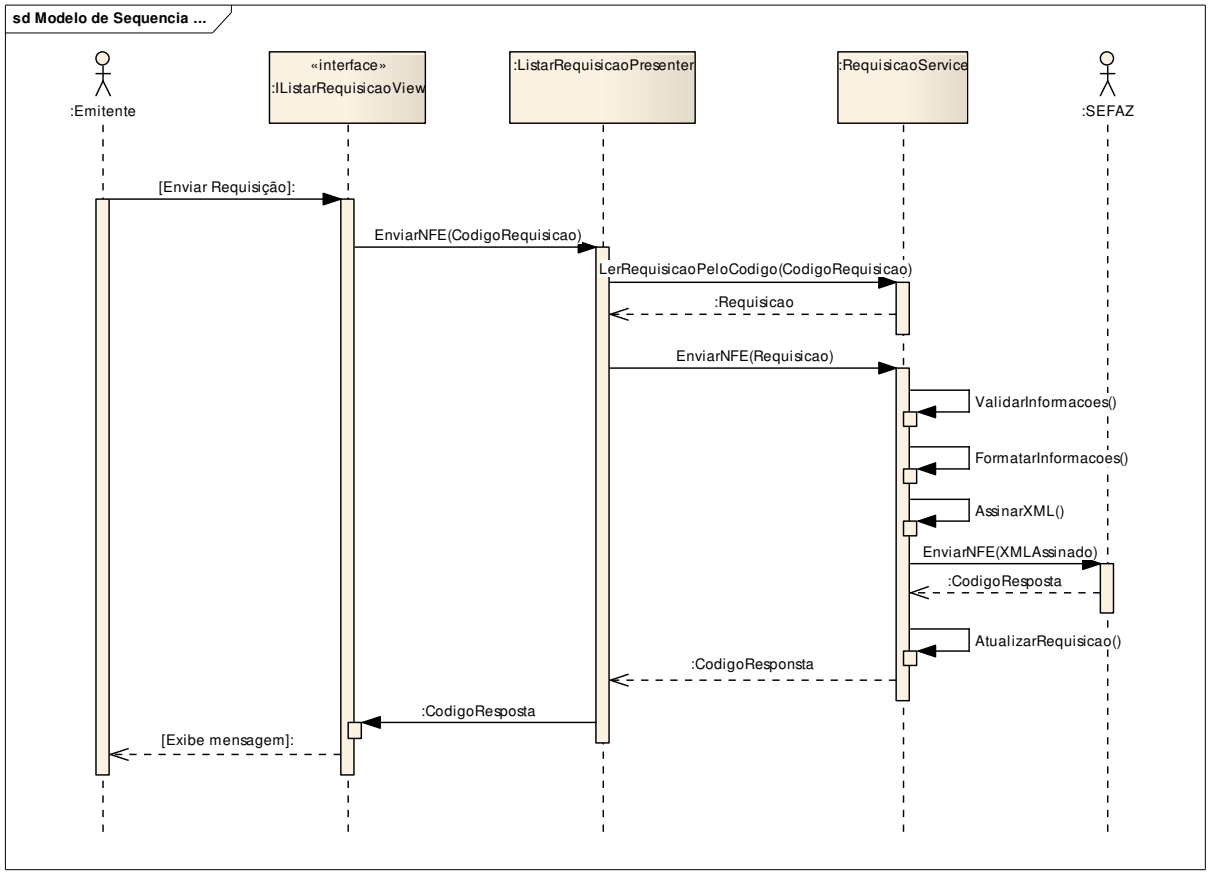


Figura 17 – Diagrama de seqüência: `enviar NF-E`

Após as validações necessárias é enviada uma mensagem ao *webservice* do SEFAZ pertencente ao estado do emitente juntamente com o XML da NF-E. Após o envio a requisição é marcada como enviada e a alteração de seus campos passa a ser vetada.

3.2.3.6 Diagrama de seqüência: `consultar NF-Es enviadas`

A consulta de NF-Es enviadas fornece ao usuário uma lista de todas as requisições convertidas para NF-E e enviadas à SEFAZ. O diagrama da Figura 18 demonstra a seqüência de mensagens entre os objetos para efetuar a consulta de NF-Es enviadas. Conforme o padrão MVP a *view* recebe a requisição do usuário, repassa a classe do *presenter* de tipo `ListarRequisicaoEnviadaPresenter` que com auxílio da classe `RequisicaoService` busca as requisições marcadas como enviadas no banco de dados e retorna ao usuário.

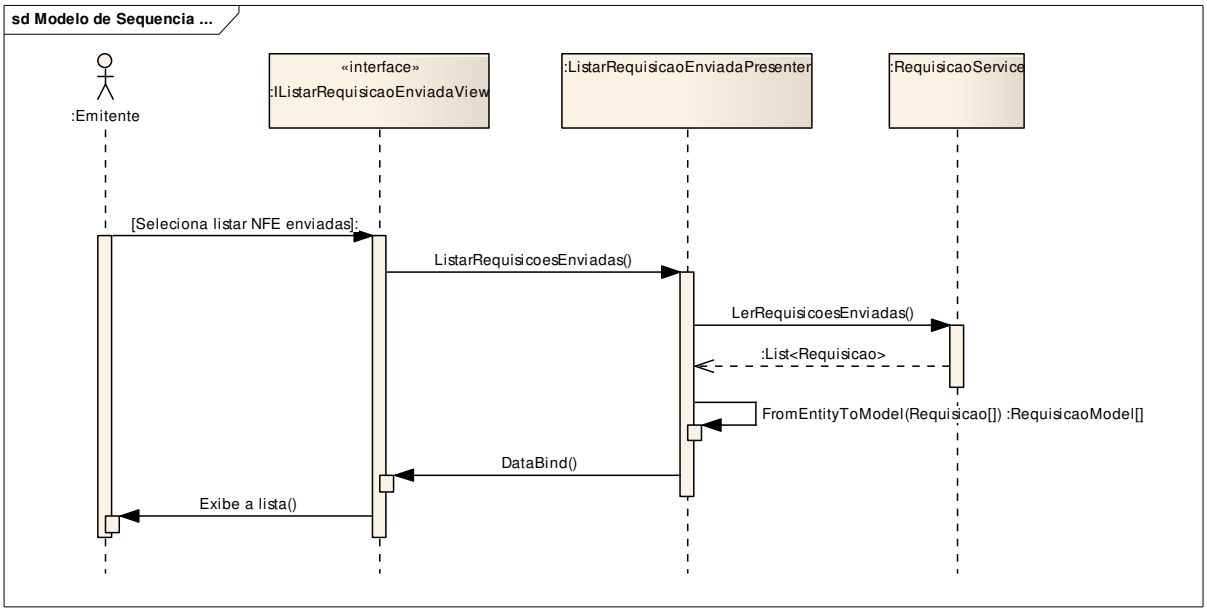


Figura 18 - Diagrama de seqüência: `consultar NF-Es enviadas`

3.2.3.7 Diagrama de seqüência: `cancelar NF-E`

Este diagrama ilustra a interação em que o usuário solicita ao protótipo o envio de uma ou mais NF-Es para cancelamento conforme o caso de uso `cancelar nota fiscal eletrônica`. Na Figura 19 é exibido o diagrama ilustrando a troca de mensagens entre as classes que têm participação neste caso de uso, onde o processo é iniciado pela interação do usuário que seleciona previamente uma ou mais NF-Es a cancelar. O protótipo utiliza a classe `RequisicaoService` para, através dela, efetuar o cancelamento de todas as NF-Es selecionadas enviando-as para o *webservice* da SEFAZ.

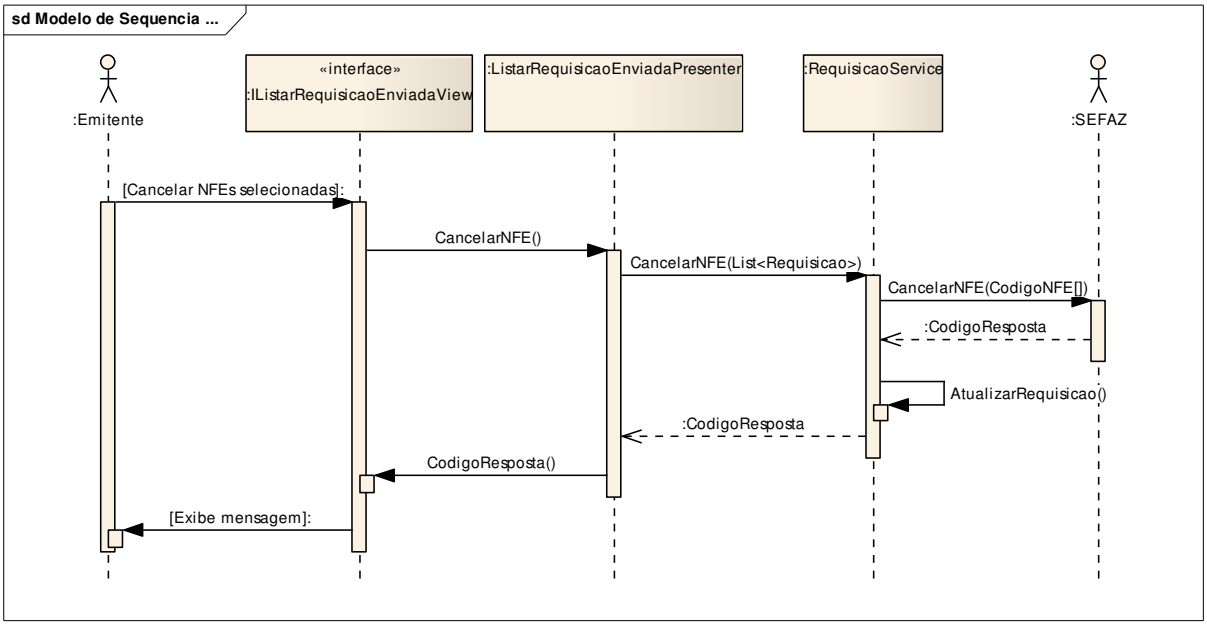


Figura 19 - Diagrama de seqüência: `cancelar NF-E`

3.3 IMPLEMENTAÇÃO

A seguir são mostradas as técnicas e ferramentas utilizadas e a operacionalidade da implementação.

3.3.1 Técnicas e ferramentas utilizadas

O protótipo proposto foi desenvolvido utilizando o *Integrated Development Environment* (IDE) Visual Studio 2008 por ter melhor suporte para desenvolvimento baseado no *framework* .NET versão 3.5 utilizando a linguagem de programação C#.Net 3.0. O desenvolvimento foi separado em vários projetos distintos de modo que atendesse ao modelo de implementação em três camadas. Para a camada de acesso a dados a tecnologia escolhida foi o *LINQ-to-Entities* que abstrai o acesso ao banco de dados fazendo o papel de um ORM. Para a camada de interface foi escolhido o *design pattern* MVP que separa esta camada em três outras camadas distribuindo de forma distinta as responsabilidades de cada uma.

3.3.2 Implementação do Protótipo

Neste capítulo será exibido as principais implementações do trabalho no que diz respeito ao MVP e as regras de negócios para cumprir com os objetivos do trabalho, tais como: *webservice* de recepção de requisição, *webservice* de envio de NF-E e assinatura do XML.

3.3.2.1 Implementação do MVP

Para a implementação do protótipo utilizando o MVP foi criada uma nova solução no Visual Studio 2008. Uma solução agrega vários projetos de vários formatos. Em seguida foi adicionado um novo projeto do tipo *class library* nesta solução para a camada de acesso a dados.

Como a camada de acesso a dados utiliza a tecnologia *LINQ-to-Entities* para abstrair o banco de dados foi adicionado um item do tipo *ADO.NET Entity Data Model* no projeto. Após a adição deste item foram importadas as tabelas do banco de dados utilizando um *wizard* da ferramenta que por fim gera as classes para acesso conforme o diagrama na Figura 7.

Para a camada de serviço de dados, foi adicionado um novo projeto do tipo *class library* e em seguida adicionada uma referência ao projeto de acesso a dados. A camada de serviço de dados é responsável pelas regras de negócios e o acesso à camada de dados. Neste projeto foram implementadas classes de serviço e interfaces para cada entidade da camada de acesso a dados.

A camada de interface é implementada utilizando o *design pattern* MVP destrinchando esta camada em quatro novos projetos, sendo, três do tipo *class library* e um do tipo *web application*. Os três projetos do tipo *class library* formam o *design pattern* MVP e são subdivididos em *model*, *view* e *presenter*.

O *model* é um projeto com classes semelhantes às entidades do projeto de acesso a dados. Sua única diferença é ser uma classe com o propósito de armazenar o estado atual de uma *view*, portanto não possui métodos conforme o Quadro 1.

```
namespace TCC.Model
{
    /// <summary>
    /// Modelo de Emitente
    /// </summary>
    public class EmitenteModel
    {
        public int Codigo { get; set; }
        public string RazaoSocial { get; set; }
        public string NomeFantasia { get; set; }
        public string Email { get; set; }
        public string CNPJ { get; set; }
        public string IE { get; set; }
        public string IEST { get; set; }
        public string CNAEFiscal { get; set; }
        public string IM { get; set; }
        public string Usuario { get; set; }
        public string Senha { get; set; }
        public bool Ativo { get; set; }
        public EnderecoModel Endereco { get; set; }
    }
}
```

Quadro 1 – Exemplo de *model* para a classe `Emitente`

A *view* por sua vez é um projeto que só possui *interfaces* que devem ser implementadas pelos *webforms* do projeto *web application*. As *interfaces* definidas na *view* devem implementar a *interface* `IView` que possui a estrutura básica a ser implementada conforme o Quadro 2. Um exemplo de *interface* que define uma entidade do tipo `Emitente` e implementa a *interface* `IView` pode ser vista no Quadro 3.

```
namespace TCC.View
{
    public interface IView
    {
        event EventHandler Init;
        event EventHandler Load;
        bool IsPostBack { get; }
        void DataBind();
        bool IsValid { get; }
    }
}
```

Quadro 2 – *Interface* base `IView`

```
namespace TCC.View
{
    public interface INovoEmitenteView : IView
    {
        string MensagemUsuario { get; set; }
        string RazaoSocial { get; set; }
        string NomeFantasia { get; set; }
        string CNPJ { get; set; }
        string IE { get; set; }
        string IESubstTributario { get; set; }
        string CNAEFiscal { get; set; }
        string IM { get; set; }
        string Email { get; set; }
        string Usuario { get; set; }
        string Senha { get; set; }
        string CEP { get; set; }
        string Estado { get; set; }
        string Cidade { get; set; }
        string Bairro { get; set; }
        string Endereco { get; set; }
        string Numero { get; set; }
        string Complemento { get; set; }
        string Telefone { get; set; }
        event EventHandler Inserir;
        event EventHandler ValidarCNPJ;
        event EventHandler ValidarEmail;
        event EventHandler ValidarUsuario;
    }
}
```

Quadro 3 – Interface para implementar uma *view* do tipo `INovoEmitenteView`

O *presenter* tem a função de receber os eventos e decidir o comportamento a ser aplicado para cada ação solicitada pelo usuário. Como exibido no trecho de código do Quadro 4 o construtor do *presenter* recebe como parâmetro a *view* que ele observa assim como a classe de serviço de dados que ele deve utilizar para persistir o que o usuário informar. O construtor do *presenter* é instanciado pela *view* que o utiliza.

Como segue a definição do MVP uma classe *presenter* atende a somente uma classe *view* ao contrário do MVC aonde uma classe *controller* pode atender a várias classes *view*.

```
namespace TCC.Presenter
{
    public class EmitentePresenter : IPresenter
    {

        IEmitenteView _view;
        IEmitenteService _service;

        public EmitentePresenter(IEmitenteView view, IEmitenteService service)
        {
            _view = view;
            _service = service;
            SubscribeViewToEvents();
        }

        void SubscribeViewToEvents()
        {
            _view.Load += OnViewLoad;
            _view.Atualizar += OnAtualizar;
            _view.Inserir += OnInserir;
            _view.Excluir += OnExcluir;
        }
```

Quadro 4 - Trecho de código contendo o construtor do *presenter* da entidade `Emitente`

Conforme o Quadro 4 o *presenter* possui um uma classe de serviço associada a ele. A classe de serviço é acionada quando o *presenter* recebe a solicitação da *view* e precisa interagir com o banco de dados. Conforme o Quadro 5 a classe de serviço é responsável pela interação com o banco de dados e nenhuma outra camada consegue acessá-lo sem passar pelas classes de serviço.

```
namespace TCC.Service
{
    public class EstadoService : IEstadoService
    {

        public EstadoModel LerEstadoPeloCodigo(int codigo)
        {
            using (TCCEntities tcc = new TCCEntities())
            {
                return EstadoService.FromEstadoEntityToModel(LerEstadoPeloCodigo(codigo, tcc));
            }
        }

        public EstadoModel LerEstadoPelaSigla(string estado)...

        public ICollection<EstadoModel> LerEstado()...

        #region Internal

        internal Estado LerEstadoPeloCodigo(int codigo, TCCEntities dataContext)
        {
            var estado = (from e in dataContext.Estado
                          where e.Codigo == codigo
                          select e).FirstOrDefault();
            return estado;
        }
```

Quadro 5 - Trecho de código da classe de serviço da entidade `Estado`

O último item a ser adicionado na solução é o projeto do tipo *web application* que é composto pela aplicação *web* disponível no modelo ASP e os *webservices* a serem consumidos pela aplicação B2B on-line integradora. Os *webforms* que compõem a aplicação *web* implementam as *interfaces* definidas no projeto da *view* conforme o Quadro 6.

```
namespace TCC.Web
{
    public partial class NovoEmitente : System.Web.UI.Page, INovoEmiteenteView
    {

        NovoEmitentePresenter _presenter = null;

        protected void Page_PreInit(object sender, EventArgs e)
        {
            _presenter = new NovoEmitentePresenter(this, new EmiteenteService());
        }
```

Quadro 6 - Trecho de código do *webform* que implementa a *view* `INovoEmiteenteView`

O MVP permite ao desenvolvedor uma maior liberdade no desenvolvimento do *webform* retirando as regras de negócio e fazendo com que os eventos do *webform* sejam tratados pelo *presenter*. Este então repassa as informações para a camada de serviço de dados responsável pelas regras de negócio e acesso a camada de dados.

3.3.2.2 Implementação do *webservice* `ReceberRequisicao`

O *webservice* `ReceberRequisicao` é o serviço oferecido para que aplicações B2B se integrem ao protótipo. Ele é composto por dois serviços conforme a Figura 20. O serviço `Autenticar` é responsável pela autenticação e é obrigatório para poder utilizar o serviço `NovaRequisicao`.

O serviço `NovaRequisicao` é responsável por receber um objeto da classe `Pedido`, converter em um objeto da classe `Requisicao` e salvar no banco de dados. Ambos os serviços possuem um objeto no retorno chamado `Erro` que contém qualquer erro ocorrido durante o processo. Um trecho do arquivo WSDL do *webservice* pode ser visto na Figura 21.

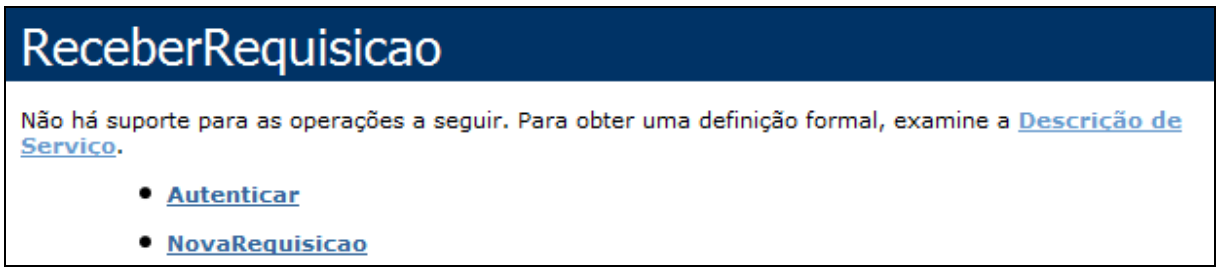


Figura 20 - Serviços oferecidos pelo *webservice* `ReceberRequisicao`

```
- <wsdl:definitions targetNamespace="http://tempuri.org/">
  - <wsdl:types>
    - <s:schema elementFormDefault="qualified" targetNamespace="http://tempuri.org/">
      - <s:element name="NovaRequisicao">
        - <s:complexType>
          - <s:sequence>
              <s:element minOccurs="0" maxOccurs="1" name="pedido" type="tns:Pedido"/>
            </s:sequence>
          </s:complexType>
        </s:element>
      - <s:complexType name="Pedido">
        - <s:sequence>
            <s:element minOccurs="1" maxOccurs="1" name="CodigoPedido" type="s:int"/>
            <s:element minOccurs="1" maxOccurs="1" name="DataPedido" type="s:dateTime"/>
            <s:element minOccurs="0" maxOccurs="1" name="CodigoCliente" type="s:string"/>
            <s:element minOccurs="0" maxOccurs="1" name="RazaoCliente" type="s:string"/>
            <s:element minOccurs="0" maxOccurs="1" name="CNPJCliente" type="s:string"/>
            <s:element minOccurs="0" maxOccurs="1" name="IECliente" type="s:string"/>
            <s:element minOccurs="0" maxOccurs="1" name="LogradouroCliente" type="s:string"/>
            <s:element minOccurs="0" maxOccurs="1" name="NumeroCliente" type="s:string"/>
            <s:element minOccurs="0" maxOccurs="1" name="BairroCliente" type="s:string"/>
            <s:element minOccurs="0" maxOccurs="1" name="CidadeCliente" type="s:string"/>
            <s:element minOccurs="0" maxOccurs="1" name="EstadoCliente" type="s:string"/>
            <s:element minOccurs="0" maxOccurs="1" name="CEPCliente" type="s:string"/>
            <s:element minOccurs="0" maxOccurs="1" name="Itens" type="tns:ArrayOfItem"/>
            <s:element minOccurs="0" maxOccurs="1" name="InfoEntrega" type="tns:Entrega"/>
          </s:sequence>
        </s:complexType>
      - <s:complexType name="ArrayOfItem">
        - <s:sequence>
            <s:element minOccurs="0" maxOccurs="unbounded" name="Item" nillable="true"
            type="tns:Item"/>
          </s:sequence>
        </s:complexType>
      - <s:complexType name="Item">
        - <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="CodigoItem" type="s:string"/>
            <s:element minOccurs="0" maxOccurs="1" name="DescricaoItem" type="s:string"/>
            <s:element minOccurs="1" maxOccurs="1" name="PrecoUnitarioItem"
            type="s:double"/>
            <s:element minOccurs="1" maxOccurs="1" name="QuantidadeItem" type="s:int"/>
          </s:sequence>
        </s:complexType>
```

Figura 21 - Trecho do WSDL do *webservice* `ReceberRequisicao`

3.3.2.3 Implementação do envio da NF-E

A solicitação de envio de NF-E é executada manualmente pelo emitente que acessa o sistema e na tela de listagem de requisições seleciona a requisição a ser enviada. Quando uma requisição é enviada a classe `IListarRequisicaoView` aciona o evento `EnviarNFE` passando como parâmetro o código de uma ou mais requisições. Como é utilizado o padrão MVP no desenvolvimento da aplicação, a *view* dispara um evento no *presenter*, representado pela classe `ListarRequisicaoPresenter`.

Quando o *presenter* recebe o evento, ele executa o método responsável pelo envio da NF-E no serviço que atende a esta *view*. O serviço, que é implementado pela classe `RequisicaoService`, inicia o processo fazendo uma validação dos campos obrigatórios conforme o Quadro 7. Caso ocorrer alguma inconsistência uma exceção é disparada e a mensagem é tratada pelo *presenter* e exibida ao usuário. No procedimento de envio em lote, ocorrendo uma exceção em qualquer requisição o processo inteiro é abortado.

```
private void ValidarInformacoes(RequisicaoModel requisicao)
{
    // Testa obrigatoriedade dos campos
    if (requisicao.Serie == string.Empty)
        throw new Exception("Série é um campo obrigatório");
    if (requisicao.NumeroNFE == string.Empty)
        throw new Exception("Numero da NF-E é obrigatório");
    if (requisicao.DataEmissao == null)
        throw new Exception("Data de Emissão é obrigatório");
    if (requisicao.NaturezaOperacao == string.Empty)
        throw new Exception("Natureza da Operação é obrigatório");
    if (requisicao.TipoDocumento == null)
        throw new Exception("Tipo de documento é obrigatório");
    if (requisicao.TipoImpressaoDanfe == null)
        throw new Exception("Tipo de Impressão do DANFE é obrigatório");
    if (requisicao.FormaPagamento == null)
        throw new Exception("Forma de Pagamento é obrigatório");
    if (requisicao.FormaEmissao == null)
        throw new Exception("Forma de Emissão é obrigatório");
    if (requisicao.FinalidadeEmissao == null)
        throw new Exception("Finalidade de Emissão é obrigatório");
```

Quadro 7 - Trecho de código da validação de uma requisição

Após a validação, o *service* transforma as requisições em XML dentro do padrão estabelecido pela SEFAZ conforme o Anexo A Neste processo cada campo é formatado e encapsulado em uma marcação conforme especifica o XML. Um trecho de código para formatação em XML é exibido no Quadro 8.

```
private XmlDocument FormatarInformacoes(Requisicao requisicao)
{
    XmlDocument xmlNFE = new XmlDocument();
    // <Nodo Raiz>
    XmlElement nfe = xmlNFE.CreateElement("NFe");
    // <Nodo Raiz> - <Informações da NFE>
    XmlElement nfeInfo = xmlNFE.CreateElement("infNFe");
    // <Nodo Raiz> - <Informações da NFE> - <Versao>
    XmlAttribute attr = xmlNFE.CreateAttribute("versao");
    attr.Value = ConfigurationSettings.AppSettings["Versao"];
    nfeInfo.Attributes.Append(attr);
    // <Nodo Raiz> - <Informações da NFE> - <ID>
    attr = xmlNFE.CreateAttribute("Id");
    attr.Value = String.Format("NFe{0}", requisicao.Codigo);
    nfeInfo.Attributes.Append(attr);
    // <Nodo Raiz> - <Informações da NFE> - <Identificação>
    XmlElement nfeIdent = xmlNFE.CreateElement("ide");
    // <Nodo Raiz> - <Informações da NFE> - <Identificação> - <UF Emitente>
    XmlElement info = xmlNFE.CreateElement("cUF");
    info.InnerText = requisicao.Emitente.Endereco.Estado.Codigo.ToString();
    nfeIdent.AppendChild(info);
    // <Nodo Raiz> - <Informações da NFE> - <Identificação> - <Chave Acesso>
    info = xmlNFE.CreateElement("cNF");
    info.InnerText = new Random().Next(999999999).ToString();
    nfeIdent.AppendChild(info);
    // <Nodo Raiz> - <Informações da NFE> - <Identificação> - <Natureza Operacao>
    info = xmlNFE.CreateElement("natOp");
    info.InnerText = requisicao.NaturezaOperacao;
    nfeIdent.AppendChild(info);
    // <Nodo Raiz> - <Informações da NFE> - <Identificação> - <Forma Pagamento>
    info = xmlNFE.CreateElement("indPag");
    info.InnerText = requisicao.FormaPagamento.ToString();
    nfeIdent.AppendChild(info);
    // <Nodo Raiz> - <Informações da NFE> - <Identificação> - <Cod. Modelo Doc.>
    info = xmlNFE.CreateElement("mod");
    info.InnerText = requisicao.Modelo.ToString();
    nfeIdent.AppendChild(info);
    // <Nodo Raiz> - <Informações da NFE> - <Identificação> - <Série>
    info = xmlNFE.CreateElement("serir");
    info.InnerText = requisicao.Serie.ToString();
```

Quadro 8 - Trecho de código da formação da requisição em XML

Com as requisições convertidas em XML o *service* assina digitalmente as notas fiscais utilizando um certificado válido fornecido pelo emitente e instalado no servidor conforme o trecho de código no Quadro 9.

```
public static void Assinar(ref XmlDocument xml, X509Certificate2 certificado, string tag)
{
    try
    {
        string x;
        x = certificado.GetKeyAlgorithm().ToString();
        // Verifica se a tag a ser assinada existe é única
        int qtdeRefUri = xml.GetElementsByTagName(tag).Count;
        if (qtdeRefUri == 0)
            throw new Exception("A tag de assinatura " + tag + " inexiste");
        else if (qtdeRefUri > 1)
            throw new Exception("A tag de assinatura " + tag + " não é unica");
        // Cria um documento SignedXml e adiciona a chave privada
        SignedXml signedXml = new SignedXml(xml);
        signedXml.SigningKey = certificado.PrivateKey;
        // Cria uma referencia a ser assinada e adiciona a tag ID nela
        Reference reference = new Reference();
        XmlAttributeCollection _tag = xml.GetElementsByTagName(tag).Item(0).Attributes;
        foreach (XmlAttribute _atributo in _tag)
            if (_atributo.Name == "Id")
                reference.Uri = "#" + _atributo.InnerText;
        // Adiciona o envelope da assinatura no documento
        XmlDsigEnvelopedSignatureTransform env = new XmlDsigEnvelopedSignatureTransform();
        reference.AddTransform(env);
        XmlDsigC14NTransform c14 = new XmlDsigC14NTransform();
        reference.AddTransform(c14);
        // adiciona a referencia ao SignedXml
        signedXml.AddReference(reference);
        // Cria um objeto de informações da chave e adiciona o certificado
        KeyInfo keyInfo = new KeyInfo();
        keyInfo.AddClause(new KeyInfoX509Data(certificado));
        // Adiciona as informações da chave no SignedXml e por fim Assina
        signedXml.KeyInfo = keyInfo;
        signedXml.ComputeSignature();
        // Pega o elemento da assinatuda e adiciona no XML original
        XmlElement xmlDigitalSignature = signedXml.GetXml();
        xml.DocumentElement.AppendChild(xml.ImportNode(xmlDigitalSignature, true));
    }
    catch (Exception ex)
```

Quadro 9 - Trecho de código que efetua a assinatura digital no XML da NF-E

Com o XML válido e assinado, o *service* inicia a comunicação com o *webservice* da SEFAZ e envia as informações conforme o Quadro 10. Após o envio, o retorno do *webservice,* conforme o Anexo B, é tratado e se houve erro o *presenter* exibe ao usuário. Caso contrário a requisição passa a ser exibida automaticamente na *view* de *IListarRequisicaoEnviadaView*.

```
public int EnviarNFE(RequisicaoModel requisicao)
{
    using (TCCEntities tcc = new TCCEntities())
    {
        // Valida as Informações
        ValidarInformacoes(requisicao);
        // Tranforma o modelo em entidade
        Requisicao novaRequisicao = RequisicaoService.FromRequisicaoModelToEntity(requisicao,
        // Transforma em XML
        XmlDocument xmlNFE = FormatarInformacoes(novaRequisicao);
        // Assinar
        NFEHelper.Assinar(ref xmlNFE, new System.Security.Cryptography.X509Certificates.X509Ce
        // Envia
        NfeRecepcaoSoapClient client = new NfeRecepcaoSoapClient();
        string ret = client.nfeRecepcaoLote(requisicao.NumeroNFE, xmlNFE.OuterXml);
        // Atualiza a requisiscao
        tcc.AddToRequisicao(novaRequisicao);
        tcc.SaveChanges();
        // retorna
        return Convert.ToInt32(ret);
    }
}
```

Quadro 10 - Método de envio de requisição

3.3.3 Operacionalidade

Nesta seção é apresentada uma demonstração das funcionalidades do protótipo. Para ter acesso à aplicação um usuário deve estar devidamente cadastrado no sistema, e para isso ele deve efetuar seu cadastro conforme a Figura 22.

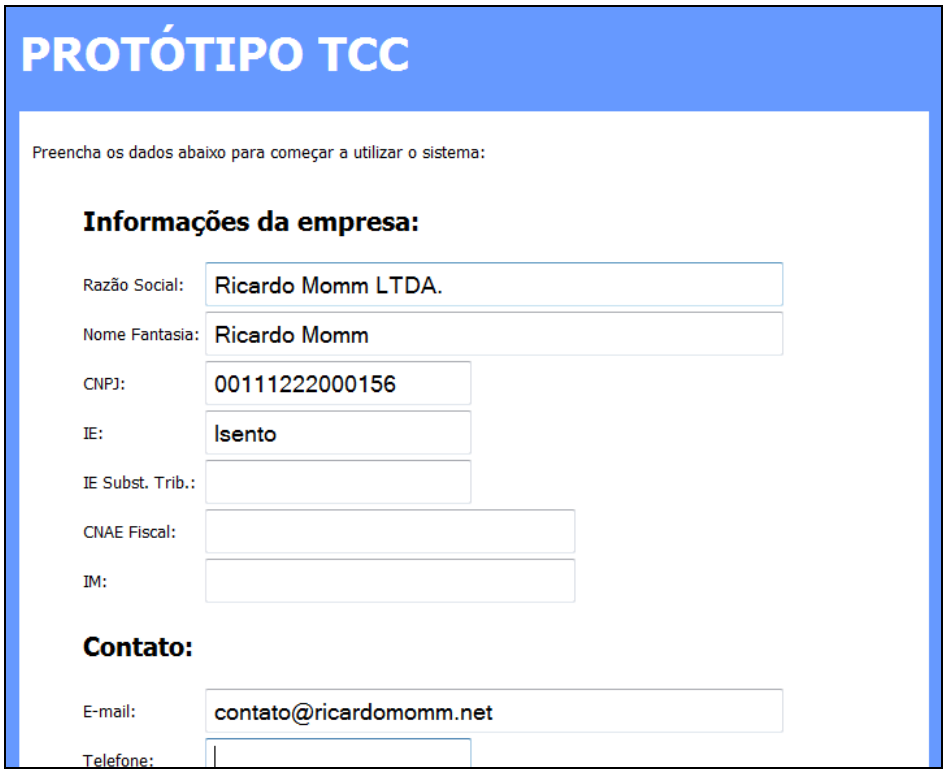


Figura 22 – Parte da tela de cadastro de `Emitente`

Após a conclusão do cadastro o usuário recebe um *e-mail* no endereço utilizado no cadastro com um *hyperlink* para ativar o cadastramento no sistema e iniciar a utilização. Com o cadastro ativo o usuário deve informar o nome de usuário e senha escolhido no cadastro para fazer a autenticação como segue na Figura 23.

Figura 23 - Tela de autenticação

Quando o usuário faz a autenticação, ele é redirecionado para um ambiente protegido com um menu de acesso às funções do protótipo e um resumo da situação das principais operações pendentes no protótipo, conforme a Figura 24.

Figura 24 - Tela inicial de um usuário autenticado

No menu da tela inicial o usuário pode navegar entre as notas fiscais enviadas através

do *hyperlink* `Notas Fiscais`, pode navegar entre as requisições que o protótipo recebeu do sistema B2B mas ainda não enviou a SEFAZ através do *hyperlink* `Requisições`, pode alterar algumas configurações pessoais do sistema através do *hyperlink* `Configurações` e pode retornar a página inicial através do *hyperlink* `Inicio`.

Selecionando a opção `Notas Fiscais` do menu, o usuário navega para uma tela que lista as requisições enviadas a SEFAZ conforme a Figura 25. Nesta tela o usuário pode consultar a situação de uma NF-E na SEFAZ, visualizar os detalhes da NF-E, visualizar o DANFE no formato PDF e exportar o XML da NF-E.

Figura 25 - Tela de NF-Es enviadas

A tela de requisições exibida na Figura 26 é muito semelhante à tela de notas fiscais enviadas. As únicas diferenças são as ações e a situação de uma requisição, que não é exibida pois ainda não foi enviada a SEFAZ. Nesta tela o usuário pode complementar sua requisição de modo que ele possa cumprir os requisitos de uma NF-E exigidos pela SEFAZ através da ação `Editar`.

A ação mais importante da tela de requisições é a ação de enviar uma NF-E. Esta ação é responsável por verificar as regras, as obrigatoriedades, fazer a conversão em XML, assinar digitalmente uma NF-E e enviá-la a SEFAZ.

Figura 26 - Tela de requisições recebidas

Na tela de requisições e na tela de NF-Es enviadas, as funções de editar e de detalhar respectivamente navegam para a mesma página, a página dos detalhes de uma NF-E exibida na Figura 27. Quando ela é acessada utilizando como parâmetro uma NF-E o sistema bloqueia as alterações, pois a NF-E já foi enviada e a tela serve somente para visualizar as informações. Quando é acessada utilizando como parâmetro uma requisição, o usuário tem permissão de alterar todos os dados de acordo com sua necessidade.

Figura 27 – Parte da tela de detalhes de uma NF-E e de uma requisição

Acessando no menu a opção `Configurações` o usuário tem um formulário com os dados de seu cadastro como emitente, conforme a Figura 28. Neste formulário o usuário tem a possibilidade de alterar seus dados de emitente e de autenticação. Qualquer alteração nesta tela reflete nos dados do emitente utilizado para NF-E.

Figura 28 – Parte da tela de configuração

3.4 RESULTADOS E DISCUSSÃO

Os resultados obtidos com o desenvolvimento deste trabalho se mostraram aceitáveis visto que grande parte dos objetivos foi atendida, tais como, o desenvolvimento de uma interface para envio de dados pelos sistemas B2B on-line, desenvolvimento das regras de negócio centralizadas na DLL do pacote `TCC.Service`, desenvolvimento de uma interface *web* para configuração do sistema, cadastro de contas de autenticação, consultas as notas fiscais eletrônicas do emitente e a implementação do protótipo no paradigma ASP.

Como idéia principal, estava o desenvolvimento de um sistema de emissão de NF-E utilizando *webservices,* que fornecesse uma interface simples para empresas que querem adequar seus sistemas B2B on-line ao modelo NF-E. Os *webservices* fornecem a interoperabilidade necessária para este interfaceamento, além de garantir o baixo custo de implementação por tratar-se de uma tecnologia de padrões abertos.

A implementação do protótipo também permitiu o conhecimento necessário quanto às

regras para emissão de NF-Es e toda a sistemática necessária para integrar uma solução a este modelo. A adaptação ao modelo de emissão de NF-E é mais complexo que a simples integração com seus *webservices*.

Todas estas regras e *webservices,* juntamente com a utilização do sistema no modelo ASP e a construção no padrão MVP, tornaram o sistema centralizado. Houve também uma maior separação de responsabilidades entre as camadas, permitindo futuramente uma facilidade maior nas modificações e implementação de testes unitários.

Uma dificuldade encontrada durante a implementação foram os testes de integração diretamente com os *webservices* de homologação da SEFAZ. Para efetuar os testes é necessária a utilização de um certificado digital válido, mesmo fora do ambiente de produção, isto porque os *webservices* utilizam autenticação mútua de servidores. Nesta modalidade de segurança o certificado digital deve ser enviado junto com a requisição para que a SEFAZ verifique se o emitente pode, ou não, utilizar os *webservices*.

Com este problema não foi possível também, mapear diretamente o WSDL dos *webservices* da SEFAZ através dos recursos do Visual Studio, que permite a geração das classes automaticamente informando a URL dos *webservices*. Para ultrapassar este obstáculo, foi efetuada uma pesquisa intensa para encontrar o modelo dos WSDL, que foram localizados via blog da SEFAZ (http://nf-eletronica.com/blog). Porém os testes integrados não puderam ser efetuados devido à falta dos certificados válidos.

A utilização do SSL e do protocolo HTTPS na interface do protótipo também necessita de certificados válidos em ambiente de produção, porém, foi possível utilizar estas tecnologias somente para testes mediante um certificado temporário e sem validade comprovada.

Em comparação com as ferramentas correlatas (Quadro 11), o protótipo se destaca em alguns pontos como a falta de necessidade de aquisição de *hardware* e da implementação das regras de negócio, porém, não possui todas as funcionalidades que as ferramentas estudadas oferecem.

Ferramenta	Funcionalidades					
	Necessita aquisição de hardware para implantação	Emissão de NF-Es para cigarros, remédios, combustíveis e armas	Permite exportação do DANFE em PDF	Permite integração com outros sistemas	Fornece interface para configuração e utilização	Fornece acesso web para consulta de situação e exportação do DANFE
Protótipo	Não	Não	Sim	Sim	Sim	Sim
Optio Software	Sim	Sim	Sim	Sim	Sim	Não
NF-Express	Sim	Sim	Não	Sim	Não	Não

Quadro 11 - Tabela de comparação de funcionalidades

Conforme o Quadro 11, o protótipo não necessita de hardware para implantação, pois ele opera sobre o paradigma ASP que provê o funcionamento do sistema como serviço em um servidor centralizado cabendo ao emitente somente fazer a integração. A opção por não emitir NF-Es para cigarros, remédios, combustíveis, armas e automóveis deve-se às propriedades específicas destes segmentos que normalmente não se encaixam na modalidade de B2B on-line.

A franquia da empresa Checkcheck na cidade de Blumenau é responsável pela comercialização de sistemas para consultas ao Serviço de Proteção ao Crédito (SPC) e ao banco de dados da empresa Serasa. Esta franquia está desenvolvendo um sistema comercial on-line chamado Efficient que deverá ser integrado ao protótipo e comercializado em todo território nacional.

4 CONCLUSÕES

Este trabalho propôs e implementou um protótipo para adequar sistemas B2B on-line à modalidade de emissão de NF-E utilizando *webservices*. Este protótipo surgiu da necessidade de ter uma opção para adoção da NF-E que tende a se tornar obrigatória em breve em todos os ramos de negócios e para empresas de todos os portes.

Durante o desenvolvimento do protótipo foi necessário um estudo mais aprofundado sobre as regras de negócio para emissão de NF-E. Estas regras destacam o protótipo perante as ferramentas correlatas, pois os sistemas B2B on-line não necessitam de modificações nos processos atuais. Toda sistemática de validação, formatação, comunicação com a SEFAZ, gerenciamento de NF-E e assinatura digital é absorvida pelo protótipo, acarretando à empresa responsável pelo sistema B2B somente a responsabilidade de garantir o fornecimento de informações através dos *webservices* disponibilizados.

A centralização das regras de negócio em uma DLL forneceu um melhor aproveitamento de código assim como uma manutenção mais fácil. O desenvolvimento no paradigma ASP foi essencial para alcançar o baixo custo, pois, fornece toda a estrutura para rodar a aplicação evitando gastos de infra-estrutura por parte dos emitentes. A interface criada para integração dos sistemas B2B utilizando *webservices* provê a integração entre sistemas em diferentes linguagens e ambientes.

A interface *web* foi necessária para manter um gerenciamento e um histórico das NF-Es. O armazenamento da NF-E deve ser mantido por pelo menos três anos para posterior consulta pelo fisco. A implementação da comunicação com autenticação mutua e a exportação das NF-Es em formato PDF foi frustrada devido à necessidade de um certificado digital válido e um credenciamento junto a SEFAZ para emissão de NF-E.

O processo de especificação foi crucial prevendo funcionalidades e permitindo um planejamento melhor para o desenvolvimento do protótipo. A ferramenta Visual Studio 2008 forneceu todas as funcionalidades para tornar o desenvolvimento do protótipo mais flexível para implementação do MVP, que provou ser um modelo adequado para desenvolvimento web por fornecer uma estrutura altamente desacoplada facilitando a manutenção e os testes unitários.

O Visual Studio 2008 também provou ter um suporte muito simples e fácil de utilizar para disponibilização e consumo de *webservices*. O *framework* .Net 3.5 e a linguagem C#.Net 3.0 possuem todas as bibliotecas necessárias ao desenvolvimento dispensando a necessidade

da utilização de componentes de terceiros durante o desenvolvimento do protótipo.

A principal limitação do protótipo é de fato não ter a funcionalidade para emissão de NF-E para bebidas, cigarros, remédios, armas e veículos. Porém, este é um mercado muito especifico. Conforme demonstrado na tabela comparativa (Quadro 11), o protótipo não ficou menos funcional que as outras ferramentas correlatas.

4.1 EXTENSÕES

Como sugestão para trabalhos futuros podem ser desenvolvidas as funcionalidades para emissão de NF-E para cigarros, bebidas, remédios, armas e veículos complementando assim seu leque de utilização. Outra funcionalidade é a criação de dois outros componentes que são SPED fiscal e SPED contábil que também estão sendo informatizados pela SEFAZ para serem utilizados em conjunto com a NF-E.

REFERÊNCIAS BIBLIOGRÁFICAS

ALFASIG CONSULTORIA E SISTEMAS. **Alfasig consultoria e sistemas.** [S.l.], 2007. Disponível em: <http://www.alfasig.com.br/index.php?id=10>. Acesso em: 10 set. 2008

BORGES JUNIOR, Maurício P. **Desenvolvendo webservices:** guia rápido C#.NET usando Visual Studio.Net 2003 com banco de dados SQL SERVER. Rio de Janeiro: Ciência Moderna, 2005.

BOODHOO, Jean-Paul. **Design patterns:** model view presenter. [S.l.], 2006. Disponível em: <http://msdn.microsoft.com/en-us/magazine/cc188690.aspx>. Acesso em: 30 out. 2008.

CONSELHO PRIVADO DA NOTA FISCAL ELETRÔNICA DO BRASIL. **CONF-EB – conselho privado da nota fiscal eletrônica do Brasil**. [S.l.], 2005. Disponível em: <http://www.NF-E.org.br>. Acesso em: 27 out. 2008.

CONSULTEMA CONSULTORIA. **Consultema:** consultoria, treinamento e sistemas. [S.l.], 2005. Disponível em: <http://www.consultema.com.br/notafiscaleletronica.htm>. Acesso em: 10 set. 2008.

ENCONTRO NACIONAL DOS COORDENADORES E ADMINISTRADORES TRIBUTÁRIOS ESTADUAIS. **Projeto nota fiscal eletrônica:** especificação do escopo do sistema. [S.l.], 2005. Disponível em: <http://www.portalfiscal.se.gov.br/WebPortalFiscal/notaFiscalEletronica/download/projeto_conceitual_do_sistema.pdf>. Acesso em: 27 out. 2008.

_____. **Manual de integração do contribuinte:** padrões técnicos de comunicação. [S.l.], 2008. Disponível em: <http://www.fazenda.gov.br/confaz/confaz/Diversos/Manual_de_Integracao-Contribuinte-CT-e_versao2_0_2a.pdf>. Acesso em: 10 nov. 2008.

FOWLER, Martin. **GUI architectures**. [S.l.], 2006. Disponível em: <http://martinfowler.com/eaaDev/uiArchs.html>. Acesso em: 30 out. 2008.

FRANCO JR., Carlos F. **E-business:** internet, tecnologia e sistemas de informação na administração de empresas. 3. ed. São Paulo: Atlas, 2005.

FREITAS, Vinicius P. Projeto nota fiscal eletrônica. In: FÓRUM DE CERTIFICAÇÃO DIGITAL, 4., 2006, Brasília. **Anais eletrônicos...** [S.l.], 2006. Disponível em: <http://www.iti.br/twiki/pub/Main/PalesCart2006/3_Painel_1_Modernizacao_Administrativa_nas_Empresas-Vinicius_Pimentel_de_Freitas.pdf>. Acesso em: 27 out. 2008.

GREER, Derek. **Interactive application architecture patterns**. [S.l.], 2007. Disponível em: <http://ctrl-shift-b.blogspot.com/2007/08/interactive-application-architecture.html>. Acesso em: 30 out. 2008.

JSK CONSULTORIA E TREINAMENTO LTDA. **WebServices**. [S.l.], 2003. Disponível em: <http://www.jsk.com.br/webservices.html>. Acesso em: 27 out. 2008.

KUROSE, James F.; ROSS, Keith W. **Redes de computadores e a internet:** uma abordagem top-down. 3. ed. Tradução Arlete Simille Marques. São Paulo: Addison Wesley, 2006.

MCLAUGHLIN, Brett. **Java & XML**. 2nd ed. Beijing: O`Reilly, 2001.

SECRETARIA DE ESTADO DA FAZENDA. **Portal de informações fiscais**. Sergipe, 2004. Disponível em: <http://www.portalfiscal.se.gov.br/WebPortalFiscal/>. Acesso em: 27 out. 2008.

RECEITA FEDERAL. **Portal da nota fiscal eletrônica**. [S.l.], 2005. Disponível em: <http://www.NF-E.fazenda.gov.br/portal/>. Acesso em: 27 out. 2008.

SCRIBNER, Kenn; STIVER, Mark C. **Applies SOAP:** implementing .NET XML web services. Indianapolis: Sams, 2002.

SNYDER, Todd. **MVC or MVP pattern:** whats the diference? [S.l.], 2007. Disponível em: <http://blogs.infragistics.com/blogs/tsnyder/archive/2007/10/17/mvc-or-mvp-pattern-whats-the-difference.aspx>. Acesso em: 30 out. 2008.

SOMMERVILLE, Ian. **Engenharia de software**. 6. ed. Tradução Maurício de Andrade. São Paulo: Addison Wesley, 2003.

VAN-DALL, Sérgio K. **Protótipo para atualização assíncrona de dados utilizando WebService**. 2006. 63f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau. Disponível em: <http://www.bc.furb.br/docs/MO/2006/307291_1_1.pdf>. Acesso em: 27 out. 2008.

WORLD WIDE WEB CONSORTIUM. **W3C:** leading the web to its full potential. [S.l.], 1994. Disponível em: <http://www.w3.org>. Acesso em: 27 out. 2008.

APÊNDICE A – Modelo de XML da NF-E

O formato da NF-E é representado por um XML elaborado pela SEFAZ conforme o Quadro 12 que apresenta um modelo XML preenchido com dados fictícios.

```
<?xml version="1.0" encoding="utf-8"?>
<NFe xmlns="http://www.portalfiscal.inf.br/nfe">
  <infNFe Id="NFe35080599999090910270550010000000015180051273"
versao="1.10">
    <ide>
      <cUF>35</cUF>
      <cNF>518005127</cNF>
      <natOp>Venda a vista</natOp>
      <indPag>0</indPag>
      <mod>55</mod>
      <serie>1</serie>
      <nNF>1</nNF>
      <dEmi>2008-05-06</dEmi>
      <dSaiEnt>2008-05-06</dSaiEnt>
      <tpNF>0</tpNF>
      <cMunFG>3550308</cMunFG>
      <tpImp>1</tpImp>
      <tpEmis>1</tpEmis>
      <cDV>3</cDV>
      <tpAmb>2</tpAmb>
      <finNFe>1</finNFe>
      <procEmi>0</procEmi>
      <verProc>NF-eletronica.com</verProc>
    </ide>
    <emit>
      <CNPJ>99999090910270</CNPJ>
      <xNome>NF-e Associacao NF-e</xNome>
      <xFant>NF-e</xFant>
      <enderEmit>
        <xLgr>Rua Central</xLgr>
        <nro>100</nro>
        <xCpl>Fundos</xCpl>
        <xBairro>Distrito Industrial</xBairro>
        <cMun>3502200</cMun>
        <xMun>Angatuba</xMun>
        <UF>SP</UF>
        <CEP>17100171</CEP>
        <cPais>1058</cPais>
        <xPais>Brasil</xPais>
        <fone>1733021717</fone>
      </enderEmit>
      <IE>123456789012</IE>
    </emit>
    <dest>
      <CNPJ>00000000000191</CNPJ>
      <xNome>DISTRIBUIDORA DE AGUAS MINERAIS</xNome>
      <enderDest>
        <xLgr>AV DAS FONTES</xLgr>
        <nro>1777</nro>
        <xCpl>10 ANDAR</xCpl>
        <xBairro>PARQUE FONTES</xBairro>
```

Quadro 12 - Modelo de XML da NF-E versão 1.10 – Parte 1

```
        <cMun>5030801</cMun>
        <xMun>Sao Paulo</xMun>
        <UF>SP</UF>
        <CEP>13950000</CEP>
        <cPais>1058</cPais>
        <xPais>BRASIL</xPais>
        <fone>1932011234</fone>
      </enderDest>
      <IE></IE>
    </dest>
    <retirada>
      <CNPJ>99171171000194</CNPJ>
      <xLgr>AV PAULISTA</xLgr>
      <nro>12345</nro>
      <xCpl>TERREO</xCpl>
      <xBairro>CERQUEIRA CESAR</xBairro>
      <cMun>3550308</cMun>
      <xMun>SAO PAULO</xMun>
      <UF>SP</UF>
    </retirada>
    <entrega>
      <CNPJ>99299299000194</CNPJ>
      <xLgr>AV FARIA LIMA</xLgr>
      <nro>1500</nro>
      <xCpl>15 ANDAR</xCpl>
      <xBairro>PINHEIROS</xBairro>
      <cMun>3550308</cMun>
      <xMun>SAO PAULO</xMun>
      <UF>SP</UF>
    </entrega>
    <det nItem="1">
      <prod>
        <cProd>00001</cProd>
        <cEAN />
        <xProd>Agua Mineral</xProd>
        <CFOP>5101</CFOP>
        <uCom>dz</uCom>
        <qCom>1000000.0000</qCom>
        <vUnCom>1</vUnCom>
        <vProd>10000000.00</vProd>
        <cEANTrib />
        <uTrib>und</uTrib>
        <qTrib>12000000.0000</qTrib>
        <vUnTrib>1</vUnTrib>
      </prod>
      <imposto>
        <ICMS>
          <ICMS00>
            <orig>0</orig>
            <CST>00</CST>
            <modBC>0</modBC>
            <vBC>10000000.00</vBC>
            <pICMS>18.00</pICMS>
            <vICMS>1800000.00</vICMS>
          </ICMS00>
        </ICMS>
        <PIS>
          <PISAliq>
            <CST>01</CST>
```

Quadro 13 - Modelo de XML da NF-E versão 1.10 – Parte 2

```
            <vBC>10000000.00</vBC>
            <pPIS>0.65</pPIS>
            <vPIS>65000</vPIS>
          </PISAliq>
        </PIS>
        <COFINS>
          <COFINSAliq>
            <CST>01</CST>
            <vBC>10000000.00</vBC>
            <pCOFINS>2.00</pCOFINS>
            <vCOFINS>200000.00</vCOFINS>
          </COFINSAliq>
        </COFINS>
      </imposto>
    </det>
    <det nItem="2">
      <prod>
        <cProd>00002</cProd>
        <cEAN />
        <xProd>Agua Mineral</xProd>
        <CFOP>5101</CFOP>
        <uCom>pack</uCom>
        <qCom>5000000.0000</qCom>
        <vUnCom>2</vUnCom>
        <vProd>10000000.00</vProd>
        <cEANTrib />
        <uTrib>und</uTrib>
        <qTrib>3000000.0000</qTrib>
        <vUnTrib>0.3333</vUnTrib>
      </prod>
      <imposto>
        <ICMS>
          <ICMS00>
            <orig>0</orig>
            <CST>00</CST>
            <modBC>0</modBC>
            <vBC>10000000.00</vBC>
            <pICMS>18.00</pICMS>
            <vICMS>1800000.00</vICMS>
          </ICMS00>
        </ICMS>
        <PIS>
          <PISAliq>
            <CST>01</CST>
            <vBC>10000000.00</vBC>
            <pPIS>0.65</pPIS>
            <vPIS>65000</vPIS>
          </PISAliq>
        </PIS>
        <COFINS>
          <COFINSAliq>
            <CST>01</CST>
            <vBC>10000000.00</vBC>
            <pCOFINS>2.00</pCOFINS>
            <vCOFINS>200000.00</vCOFINS>
          </COFINSAliq>
        </COFINS>
      </imposto>
    </det>
    <total>
```

Quadro 14 - Modelo de XML da NF-E versão 1.10 – Parte 3

```
      <ICMSTot>
        <vBC>20000000.00</vBC>
        <vICMS>18.00</vICMS>
        <vBCST>0</vBCST>
        <vST>0</vST>
        <vProd>20000000.00</vProd>
        <vFrete>0</vFrete>
        <vSeg>0</vSeg>
        <vDesc>0</vDesc>
        <vII>0</vII>
        <vIPI>0</vIPI>
        <vPIS>130000.00</vPIS>
        <vCOFINS>400000.00</vCOFINS>
        <vOutro>0</vOutro>
        <vNF>20000000.00</vNF>
      </ICMSTot>
    </total>
    <transp>
      <modFrete>0</modFrete>
      <transporta>
        <CNPJ>99171171000191</CNPJ>
        <xNome>Distribuidora de Bebidas Fazenda de SP Ltda.</xNome>
        <IE>171999999119</IE>
        <xEnder>Rua Central 100 - Fundos - Distrito Industrial</xEnder>
        <xMun>SAO PAULO</xMun>
        <UF>SP</UF>
      </transporta>
      <veicTransp>
        <placa>BXI1717</placa>
        <UF>SP</UF>
        <RNTC>123456789</RNTC>
      </veicTransp>
      <reboque>
        <placa>BXI1818</placa>
        <UF>SP</UF>
        <RNTC>123456789</RNTC>
      </reboque>
      <vol>
        <qVol>10000</qVol>
        <esp>CAIXA</esp>
        <marca>LINDOYA</marca>
        <nVol>500</nVol>
        <pesoL>1000000000.000</pesoL>
        <pesoB>1200000000.000</pesoB>
        <lacres>
          <nLacre>XYZ10231486</nLacre>
        </lacres>
      </vol>
    </transp>
    <infAdic>
      <infAdFisco>Nota Fiscal de exemplo NF-eletronica.com</infAdFisco>
    </infAdic>
  </infNFe>
</NFe>
```

Quadro 15 - Modelo de XML da NF-E versão 1.10 – Parte 4

ANEXO A – Tabela de códigos de retorno dos *webservices* da SEFAZ

Todo *webservice* disponibilizado pela SEFAZ fornece um retorno conforme o Quadro 16 para que o sistema consumidor possa tratar as exceções e fornecer ao usuário uma informação clara sobre o erro.

Código	Descrição
100	Autorizado o uso da NF-e
101	Cancelamento de NF-e homologado
102	Inutilização de número homologado
103	Lote recebido com sucesso
104	Lote processado
105	Lote em processamento
106	Lote não localizado
107	Serviço em Operação
108	Serviço Paralisado Momentaneamente (curto prazo)
109	Serviço Paralisado sem Previsão
110	Uso Denegado
111	Consulta cadastro com uma ocorrência
112	Consulta cadastro com mais de uma ocorrência
201	Rejeição: O numero máximo de numeração de NF-e a inutilizar ultrapassou o limite
202	Rejeição: Falha no reconhecimento da autoria ou integridade do arquivo digital
203	Rejeição: Emissor não habilitado para emissão da NF-e
204	Rejeição: Duplicidade de NF-e
205	Rejeição: NF-e está denegada na base de dados da SEFAZ
206	Rejeição: NF-e já está inutilizada na Base de dados da SEFAZ
207	Rejeição: CNPJ do emitente inválido
208	Rejeição: CNPJ do destinatário inválido
209	Rejeição: IE do emitente inválida
210	Rejeição: IE do destinatário inválida
211	Rejeição: IE do substituto inválida
212	Rejeição: Data de emissão NF-e posterior a data de recebimento
213	Rejeição: CNPJ-Base do Emitente difere do CNPJ-Base do Certificado Digital
214	Rejeição: Tamanho da mensagem excedeu o limite estabelecido
215	Rejeição: Falha no schema XML
216	Rejeição: Chave de Acesso difere da cadastrada
217	Rejeição: NF-e não consta na base de dados da SEFAZ
218	Rejeição: NF-e já esta cancelada na base de dados da SEFAZ
219	Rejeição: Circulação da NF-e verificada
220	Rejeição: NF-e autorizada há mais de 7 dias (168 horas)
221	Rejeição: Confirmado o recebimento da NF-e pelo destinatário
222	Rejeição: Protocolo de Autorização de Uso difere do cadastrado

Fonte: adaptado de Encontro Nacional de Coordenadores Administradores Tributários Estaduais (2008, p.65).

Quadro 16 – Tabela de códigos e descrição de erros dos *webservices* da SEFAZ – Parte 1

Código	Descrição
223	Rejeição: CNPJ do transmissor do lote difere do CNPJ do transmissor da consulta
224	Rejeição: A faixa inicial é maior que a faixa final
225	Rejeição: Falha no Schema XML da NFe
226	Rejeição: Código da UF do Emitente diverge da UF autorizadora
227	Rejeição: Erro na Chave de Acesso - Campo ID
228	Rejeição: Data de Emissão muito atrasada
229	Rejeição: IE do emitente não informada
230	Rejeição: IE do emitente não cadastrada
231	Rejeição: IE do emitente não vinculada ao CNPJ
232	Rejeição: IE do destinatário não informada
233	Rejeição: IE do destinatário não cadastrada
234	Rejeição: IE do destinatário não vinculada ao CNPJ
235	Rejeição: Inscrição SUFRAMA inválida
236	Rejeição: Chave de Acesso com dígito verificador inválido
237	Rejeição: CPF do destinatário inválido
238	Rejeição: Cabeçalho - Versão do arquivo XML superior a Versão vigente
239	Rejeição: Cabeçalho - Versão do arquivo XML não suportada
240	Rejeição: Cancelamento/Inutilização - Irregularidade Fiscal do Emitente
241	Rejeição: Um número da faixa já foi utilizado
242	Rejeição: Cabeçalho - Falha no Schema XML
243	Rejeição: XML Mal Formado
244	Rejeição: CNPJ do Certificado Digital difere do CNPJ da Matriz e do CNPJ do Emitente
245	Rejeição: CNPJ Emitente não cadastrado
246	Rejeição: CNPJ Destinatário não cadastrado
247	Rejeição: Sigla da UF do Emitente diverge da UF autorizadora
248	Rejeição: UF do Recibo diverge da UF autorizadora
249	Rejeição: UF da Chave de Acesso diverge da UF autorizadora
250	Rejeição: UF diverge da UF autorizadora
251	Rejeição: UF/Município destinatário não pertence a SUFRAMA
252	Rejeição: Ambiente informado diverge do Ambiente de recebimento
253	Rejeição: Digito Verificador da chave de acesso composta inválida
254	Rejeição: NF-e referenciada não informada para NF-e complementar
255	Rejeição: Informada mais de uma NF-e referenciada para NF-e complementar
256	Rejeição: Uma NF-e da faixa já está inutilizada na Base de dados da SEFAZ
257	Rejeição: Solicitante não habilitado para emissão da NF-e
258	Rejeição: CNPJ da consulta inválido
259	Rejeição: CNPJ da consulta não cadastrado como contribuinte na UF
260	Rejeição: IE da consulta inválida
261	Rejeição: IE da consulta não cadastrada como contribuinte na UF
262	Rejeição: UF não fornece consulta por CPF
263	Rejeição: CPF da consulta inválido
264	Rejeição: CPF da consulta não cadastrado como contribuinte na UF
265	Rejeição: Sigla da UF da consulta difere da UF do Web Service

Fonte: adaptado de Encontro Nacional de Coordenadores Administradores Tributários Estaduais (2008, p.66).

Quadro 17 – Tabela de códigos e descrição de erros dos *webservices* da SEFAZ – Parte 2

Código	Descrição
266	Rejeição: Série utilizada não permitida no Web Service
267	Rejeição: NF Complementar referencia uma NF-e inexistente
268	Rejeição: NF Complementar referencia uma outra NF-e Complementar
269	Rejeição: CNPJ Emitente da NF Complementar difere do CNPJ da NF Referenciada
270	Rejeição: Código Município do Fato Gerador: dígito inválido
271	Rejeição: Código Município do Fato Gerador: difere da UF do emitente
272	Rejeição: Código Município do Emitente: dígito inválido
273	Rejeição: Código Município do Emitente: difere da UF do emitente
274	Rejeição: Código Município do Destinatário: dígito inválido
275	Rejeição: Código Município do Destinatário: difere da UF do Destinatário
276	Rejeição: Código Município do Local de Retirada: dígito inválido
277	Rejeição: Código Município do Local de Retirada: difere da UF do Local de Retirada
278	Rejeição: Código Município do Local de Entrega: dígito inválido
279	Rejeição: Código Município do Local de Entrega: difere da UF do Local de Entrega
280	Rejeição: Certificado Transmissor inválido
281	Rejeição: Certificado Transmissor Data Validade
282	Rejeição: Certificado Transmissor sem CNPJ
283	Rejeição: Certificado Transmissor - erro Cadeia de Certificação
284	Rejeição: Certificado Transmissor revogado
285	Rejeição: Certificado Transmissor difere ICP-Brasil
286	Rejeição: Certificado Transmissor erro no acesso a LCR
287	Rejeição: Código Município do FG - ISSQN: dígito inválido
288	Rejeição: Código Município do FG - Transporte: dígito inválido
289	Rejeição: Código da UF informada diverge da UF solicitada
290	Rejeição: Certificado Assinatura inválido
291	Rejeição: Certificado Assinatura Data Validade
292	Rejeição: Certificado Assinatura sem CNPJ
293	Rejeição: Certificado Assinatura - erro Cadeia de Certificação
294	Rejeição: Certificado Assinatura revogado
295	Rejeição: Certificado Assinatura difere ICP-Brasil
296	Rejeição: Certificado Assinatura erro no acesso a LCR
297	Rejeição: Assinatura difere do calculado
298	Rejeição: Assinatura difere do padrão do Projeto
299	Rejeição: XML da área de cabeçalho com codificação diferente de UTF-8
401	Rejeição: CPF do remetente inválido
402	Rejeição: XML da área de dados com codificação diferente de UTF-8
403	Rejeição: O grupo de informações da NF-e avulsa é de uso exclusivo do Fisco
404	Rejeição: Uso de prefixo de namespace não permitido
405	Rejeição: Código do país do emitente: dígito inválido
406	Rejeição: Código do país do destinatário: dígito inválido
407	Rejeição: O CPF só pode ser informado no campo emitente para a NF-e avulsa
999	Rejeição: Erro não catalogado (informar a mensagem de erro capturado no tratamento da exceção)

Fonte: adaptado de Encontro Nacional de Coordenadores Administradores Tributários Estaduais (2008, p.67).

Quadro 18 – Tabela de códigos e descrição de erros dos *webservices* da SEFAZ – Parte 3

Código	Descrição
301	Uso Denegado : Irregularidade fiscal do emitente
302	Uso Denegado : Irregularidade fiscal do destinatário

Fonte: adaptado de Encontro Nacional de Coordenadores Administradores Tributários Estaduais (2008, p.67).

Quadro 19 – Tabela de códigos e descrição de erros dos *webservices* da SEFAZ – Parte 4