UNIVERSIDADE DE LISBOA

FACULDADE DE CIÊNCIAS

DEPARTAMENTO DE INFORMÁTICA

Desenvolvimento de uma Aplicação em Orientação a Objetos

Pedro Miguel Ferreira Tavares Carrega

Mestrado em Engenharia Informática

Trabalho de Projeto orientado por:
Professora Maria Isabel Alves Batalha Reis da Gama Nunes

2022

Agradecimentos

Quero primeiro agradecer à minha orientadora, Prof. Maria Isabel Nunes, por todo o apoio e ajuda que me deu ao longo do ano. Sem ele a tese não seria possível.

Quero agradecer a todos os membros da equipa ARTSOFT pelo ambiente de equipa e por estarem disponíveis para me ajudar com qualquer problema ou dúvida que surgisse. Em particular quero agradecer ao meu coorientador João Catalão por todos os conselhos e apoio, ao Pedro Almeida por me fazer sentir que já fazia parte da equipa desde o primeiro dia, e ao João Costa por tirar tempo do seu trabalho para me esclarecer qualquer dúvida que tivesse.

Aos meus pais e irmão por todo o apoio e suporte que me deram ao longo do ano.

E por fim aos meus amigos por todos os momentos de distração e convívio.

Para o meu avô.

Resumo

O ser humano é uma espécie sempre em desenvolvimento. Para acompanhar esse mesmo desenvolvimento existe uma constante evolução tecnológica, que se aplica a todas as facetas da nossa sociedade, quer seja social, política ou económica. Contudo esta evolução tem uma potencial consequência: o aumento do número e da complexidade de recursos que podem ser tratados. A necessidade de gestão de todos estes recursos levou à criação de aplicações de *Enterprise Resource Planning* (ERP), sendo este tipo de aplicação a desenvolvida na empresa ARTSOFT. ERP ARTSOFT é uma aplicação de gestão empresarial que é estruturada em vários módulos, como Gestão Comercial, Contabilidade, Gestão Financeira e Recursos Humanos, permitindo adaptar a solução às necessidades do utilizador.

Este trabalho explora o módulo de Recursos Humanos, com o desenvolvimento de uma interface que integra os *web services* disponibilizados pela Segurança Social, para comunicar o vínculo de contrato ou o cessar de contrato de um trabalhador. Esta interface irá permitir ao utilizador pré-visualizar e editar os dados a serem comunicados pelo *web service*, propagando possíveis alterações efetuadas para a base de dados.

Palavras-chave: ERP ARTSOFT, OOP, Serviços Web, Segurança Social.

Abstract

The human species is in a state of permanent development. To match this development it exists a constant technological development to all facets of our society, whether it is social, political or economic. However this evolution comes with a potential cost: the increase in the number and complexity of the resources to be processed. That need for the management of all those resources led to the creation of Enterprise Resource Planning (ERP) applications, being this the type of solution developed at the company ARTSOFT. ERP ARTSOFT is an integrated management business application that is divided in multiple modules, like Commercial Management, Accounting, Financial Management and Human Resources, allowing the user to create a solution according to his needs.

This thesis project will revolve around the Human Resources modules, in particular the development of an interface that integrates web services made available by Segurança Social, to communicate the contract agreement or end of the contract agreement of a worker. This interface will allow the user to preview and edit the contract info to be communicated to the web service, saving possible changes in the database.

Keywords: ERP ARTSOFT, OOP, Segurança Social, Web Services

Conteúdo

Lista de Figuras

Glossário

API	Application Program Interface
Combo Boxes	Elemento de interface gráfica - Caixa de listagem que apresenta diversas opções quando o objeto é selecionado
ERP	Enterprise Resource Planning
HTTP	Hypertext Transfer Protocol
Host	Computador onde é executado um serviço *web*
MFC	Microsoft Foundation Class
SOAP	Simple Object Access Protocol
Socket	Objeto virtual pelo qual são enviados e recebidos dados
Stakeholders	Individual ou grupo afetado pelo *software* a desenvolver
SSD	System Sequence Diagram
WSDL	Web Service Description Language - Permite descrever um *web service* especificando a localização do serviço e os métodos que disponibiliza
XML	Extensible Markup Language

Capítulo 1

Introdução

O desenvolvimento tecnológico é o responsável por grande parte do avanço científico observável na nossa sociedade, sendo que este desenvolvimento é visível em todas as facetas da sociedade, quer seja social, político ou económico. De um ponto de vista empresarial este desenvolvimento trouxe várias vantagens oferecendo às empresas mais dados e informação sobre o mercado e o produto em que atua, permitindo uma gestão e análise mais detalhada e precisa do seu negócio. Porém, estas vantagens trazem possíveis consequências às empresas: um aumento da quantidade e complexidade dos recursos a processar, levando a uma gestão e análise mais lenta e menos rigorosa do seu negócio. Isto levou à criação de aplicações ERP, *software* capaz de gerir as atividades financeiras de uma empresa, as cadeias de fornecimentos, comércio, recursos humanos e outros.

ARTSOFT é uma empresa de *software*, com mais de 30 anos de experiência na área, que desenvolve soluções de gestão empresarial. A empresa produz atualmente somente um produto ERP ARTSOFT, uma aplicação de gestão constituída por vários módulos, permitindo a criação de uma solução que melhor se adapte às necessidades do cliente. Estes módulos cobrem todas as necessidades que uma empresa possa ter como, por exemplo, análise contabilística ou gestão de recursos humanos.

No âmbito da disciplina de Projeto do Mestrado de Engenharia Informática da Faculdade de Ciências da Universidade de Lisboa, a empresa ARTSOFT propôs um projeto, focado no módulo de Recursos Humanos, cujo propósito é a integração de vários serviços criados e disponibilizados pela Segurança Social. O aluno foi inserido na equipa de desenvolvimento da versão de *desktop* da aplicação ERP ARTSOFT e, sob a orientação de um orientador atribuído pela empresa, seguiu o procedimento regular de um desenvolvimento efetuado pela equipa, desde a análise do problema, até a implementação da solução.

1.1 Motivação

Qualquer ação que envolve um contrato entre empresa e trabalhador, como por exemplo o assinar ou terminar de um contrato, tem de ser reportado à Segurança Social. Anteriormente só podia ser efetuado de duas diferentes maneiras: a empresa utiliza o seu acesso na plataforma *online* da Segurança Social e aí comunica essa mudança, ou pode se deslocar diretamente a uma delegação da Segurança Social; pode ser bastante ineficiente.

Um grande número de empresas utilizam aplicações ERP para suportar o seu negócio e grande parte dessas aplicações disponibilizam funcionalidades de base de dados que guardam toda a informação relevante referente aos seus trabalhadores. Desta maneira, uma empresa que efetue uma nova contratação tem de criar um registo no ERP e depois preencher o registo de contrato na plataforma da Segurança Social com esses mesmos dados, o que, num contexto empresarial, causa um gasto de tempo desnecessariamente dispendioso.

Para suportar as empresas, a Segurança Social anunciou dois *web services*: o Consultar Trabalhadores e o Registar Vínculo Trabalhador; ambos estes serviços encontram-se integrados na aplicação ERP ARTSOFT. No fim do ano 2021 a Segurança Social anunciou a criação de diversos novos *web services* com o objetivo de suportar ainda mais as empresas. Em particular anunciou os dois serviços a serem integrados, na aplicação ERP ARTSOFT, no âmbito deste projeto: o Comunicar Vínculo do Contrato de Trabalhador, o sucessor do serviço Registar Vínculo Trabalhador que irá ser descontinuado, e o Cessar Vínculo de Trabalhador. Todos estes serviços serão integrados gradualmente na aplicação ERP ARTSOFT de forma a conseguir responder a todas as possíveis necessidades do cliente.

1.2 Objetivos

O objetivo deste projeto é a integração no ERP ARTSOFT de dois diferentes *web services* disponibilizados pela Segurança Social: o comunicar do vínculo de trabalhador e o cessar do vínculo de trabalhador. De forma a obter uma solução de qualidade, os seguintes sub-objetivos devem ser alcançados:

1. Estudo da aplicação ERP ARTSOFT e dos *web services* a integrar: A aplicação ERP ARTSOFT é composta por vários módulos em que alguns dependem de outros. Senda esta uma aplicação orientada a objetos, desenhada de forma a ser extensível, este projeto deve acomodar os novos serviços sem comprometer o funcionamento de qualquer funcionalidade já existente na aplicação ERP ARTSOFT.

2. Identificação dos requisitos funcionais: A partir do estudo realizado à aplicação ERP ARTSOFT e dos *web services* a serem integrados, são definidos os requisitos funcionais a serem implementados tendo em conta os objetivos do desenvolvimento.
3. Redação de especificações de requisitos: Tendo em conta os resultados do objetivo anterior, é redigida uma especificação de requisitos para cada *web service* a ser integrado na aplicação.
4. Desenho da solução: Dados os resultados da fase de análise do projeto são definidos os novos conceitos a implementar e tomadas decisões sobre a implementação.
5. Implementação da solução: Nesta fase do projeto são implementadas as funcionalidades identificadas e definidas na especificação de requisitos, com consideração para a arquitetura da aplicação e dos módulos afetados por este desenvolvimento.
6. Realização de testes: Uma vez implementada a solução, é efetuada uma bateria de testes funcionais para garantir que todos os requisitos funcionais identificados são respeitados. De notar que, embora o programador responsável efetue testes iniciais, os testes mencionados são realizados pela equipa de testes, por isso esta parte do desenvolvimento não vai ser analisado neste relatório.

1.3 Estrutura do documento

Este relatório é composto por sete capítulos. Além do presente capítulo, são apresentados no capítulo 2 alguns conceitos básicos necessários para auxiliar a compreensão dos temas e conceitos discutidos neste relatório. No capítulo 3 será apresentada uma análise em detalhe do projeto e as suas conclusões. Sobre esta análise iremos efetuar um desenho da solução. De seguida no capítulo 5 é abordada a concretização da implementação da solução do projeto, com uma apresentação dos resultados do desenvolvimento no capítulo 6. Finalmente no capítulo 7, as principais conclusões e trabalho futuro.

Capítulo 2

Background

Neste capítulo são apresentados e discutidos vários conceitos base e trabalhos relacionados com este projeto para inspiração e auxiliar a compreensão de algumas decisões efetuadas durante o desenvolvimento.

2.1 Microsoft Foundation Class

MFC é uma biblioteca introduzida em C++ 7.0 que oferece várias classes que servem como interface para a API do Windows. No contexto deste projeto, desta biblioteca foram somente utilizados dois tipos de objetos: janelas, que têm o seu comportamento definido na classe CDialog, e controladores, objetos responsáveis pela apresentação de informação ao utilizador e por receber os pedidos do utilizador. Estes controladores podem ter diferentes formatos dependendo da necessidade do programa, desde controladores *static* (simples mostradores de texto), até algo mais complexos como *grids* (tabelas totalmente customizáveis).

2.2 Middleware

Middlewares são criados com o propósito de permitir a comunicação entre diferentes sistemas independentemente da implementação ou *hardware* que utilizem. Contudo, com o aumento da complexidade do *middlewares*, o problema que levou à sua criação voltou a surgir, criando uma necessidade de *"middleware for middleware"*. Os *web services* tentam resolver este problema mas, ao focarem as implementações no uso de SOAP API, os programadores destas soluções estão a ficar aquém do potencial de *web services*. SOAP API não segue um padrão universal, ou seja, diferentes aplicações têm diferentes implementações da mesma. Este é o problema apresentado no artigo de Vinoski, S., que expõe e explica em detalhe um dos problemas que os *web services* enfrentam. O autor desenvolve o tema apresentando várias soluções, em particular uma solução em desenvolvimento por uma equipa de engenheiros da Sun Microsystems com o nome Web Services Invocation Framework (WSIF). Esta solução abstrai o protocolo de comunicação, permitindo as aplicações seguirem só um padrão de comunicação, independente do protocolo utilizado pelo *web service*. Segundo o autor, esta solução resolve vários dos problemas atuais de *web services* mas é longe de ser perfeita sendo exclusiva a soluções que utilizem Java.

2.3 Web Services

Web Services é um tema já bastante explorado na área de engenharia de *software* devido à sua constante evolução, existindo vários artigos que propõe e exploram diferentes metodologias e técnicas de implementação. No contexto desta tese, um *web service* é uma interface que permite comunicação entre um dispositivo e um servidor de base de dados através de pedidos *web* respondendo com um documento *web* (HTML).

No artigo de Lee *et al.* é defendido que as atuais metodologias utilizadas para o desenvolvimento de aplicações que integram *web services* no seu funcionamento não se revelam suficientes. As principais dificuldades destacadas são:

- A dificuldade em determinar todos os requisitos da aplicação dado que os requisitos não provêm de uma só fonte e
- A implementação do consumo e comunicação do serviço, pois diferentes sistemas utilizam diferentes *interfaces* e métodos de interação.

O autor do artigo propõe então uma integração das melhores práticas da metodologia *Agile* no desenvolvimento de *web services* seguindo o seguinte fluxo:

1. Definir os requerimentos do projeto a partir da análise do problema e das necessidades dos *stakeholders*;
2. Efetuar uma análise do sistema e dos casos de uso;
3. Desenhar os casos de uso e os componentes a implementar, e definir a arquitetura do sistema;
4. Implementar a solução desenhada;
5. Definir testes e executá-los sobre o serviço;
6. Disponibilização do *web service*

2.4 Base64

É uma codificação de binário para texto representada por sequências de 24 bits. Regularmente utilizada em meios que só suportam comunicação por texto, como por exemplo a internet, convertendo data (binário) para texto. A privacidade e segurança dos dados codificados não é garantida utilizando Base64 sendo que a mesma só esconde os dados enviados. Para comunicar com os serviços da Segurança Social é necessário codificar os dados de acesso em Base64.

2.5 Polimorfismo

Uma das bases da programação orientada a objetos, o polimorfismo tenta responder à necessidade da utilização de várias classes ou objetos semelhantes a partir de uma única interface que serve como base para várias classes. Estas modificações são comumente realizadas com *overloading* de métodos e a partir de herança de classes e *overriding* de métodos.

2.5.1 Overloading

A definição de várias funções e/ou construtores com o mesmo nome, mas com parâmetros diferentes, permitindo diferentes implementações da mesma função.

2.6 Herança

Um mecanismo de estruturação e reutilização de código. Normalmente associado com o princípio de polimorfismo, permite a evolução do *software* de forma natural sem alterar implementações já existentes. Uma classe criada a partir de herança mantém todas as propriedades da super classe com exceção de construtores e destrutores. Através da redefinição de funções é possível definir um comportamento diferente de funções já implementadas. Isto vai permitir o nosso programa utilizar a mesma super classe como base do objeto que representa a resposta do serviço, mas permitindo implementar diferentes processamentos da resposta consoante o serviço utilizado pelo utilizador.

2.7 Padrões de Desenho

Padrões de desenho são soluções reutilizáveis para problemas comuns representando as melhores práticas de programação. No desenvolvimento de sistemas orientado a objetos, como é o caso do ERP ARTSOFT, estes padrões focam-se normalmente nas relações e interações entre classes ou objetos. A existência de vários diferentes padrões não implica a necessidade de os aplicar a todos, sendo necessário uma análise para determinar quais fazem mais sentido aplicar na solução a desenvolver. Os padrões aqui referidos são observáveis na solução desenvolvida.

2.7.1 Low Coupling

Coupling define o nível de dependência entre classes e objetos. Uma classe com nível alto de dependência de outras classes e/ou objetos menos reutilizável e mais difícil de modificar e compreender.

2.7.2 Alta Coesão

Outro conceito básico em programação orientada a objetos, a coesão é uma medida para determinar quão fortemente relacionados estão diferentes classes e/ou objetos. Classes com um alto nível de coesão são mais fáceis de compreender, reutilizar e de manter. Este padrão costuma estar diretamente relacionado com o padrão de *low coupling*.

2.7.3 Strategy

Este padrão determina que cada algoritmo deve ter a sua própria classe, com uma super classe como base, decidindo em tempo de execução qual algoritmo aplicar. Por exemplo, se tivermos uma lista de *web services* é criada uma classe para cada serviço, contudo, a classe que implementa a comunicação entre a aplicação e o serviço é partilhada entre todos, sendo decidida a classe a utilizar consoante as necessidades do utilizador.

2.7.4 Observer

Este padrão é relacionado com o tratamento de eventos; um objeto mantém uma lista de observadores que notifica automaticamente caso ocorra alguma mudança de estado, ou seja, cria um evento, chamando um método associado. A aplicação deste padrão é visível nas interfaces implementadas, dado que cada uma tem um objeto *callback* associado. Este objeto é notificado sempre que o utilizador tenta alterar um valor de um campo da interface ou tenta submeter os valores presentes, que permite ao programa validar os valores de campo sem ter de fechar a interface.

Capítulo 3

Análise do Problema

Neste capítulo é realizada uma análise do problema e do módulo no qual vai ser realizado o desenvolvimento deste projeto com o objetivo de responder aos objetivos 2 e 3: determinar os requisitos funcionais e a redação da especificação de requisitos. Para melhor responder a essas questões, este capítulo vai seguir a seguinte estrutura: primeiro vai ser efetuada uma análise do problema; de seguida será feito um estudo da aplicação ERP ARTSOFT, em particular do módulo relevante para o projeto; segue-se a apresentação e contextualização da estrutura da especificação de requisitos; conclui-se com o estudo da documentação relevante e redação da especificação de requisitos para cada desenvolvimento associado a este projeto - a comunicação do vínculo de contrato do trabalhador e o cessar do contrato de trabalhador.

3.1 Objetivo

Como dito anteriormente, o objetivo deste projeto é a integração dos *web services* da Segurança Social, comunicação do vínculo de trabalho e comunicar a cessação do vínculo, com a aplicação ERP ARTSOFT. Ambos os serviços a serem integrados no âmbito deste projeto são bastante semelhantes, sendo iguais no método de comunicação entre o cliente e servidor. Ambos os serviços utilizam *Hypertext Transfer Protocol Secure* (HTTPS) para a comunicação entre a cliente e o servidor, utilizando o formato SOAP XML para os seus pedidos, sendo este um formato bastante comum dada a sua versatilidade. Para autenticar o utilizador é utilizado um cabeçalho *http Basic Auth*, onde é concatenado o nome do utilizador com sua palavra-passe codificando ambos em Base64, sendo que o corpo do pedido tem toda a informação que é relevante comunicar ao serviço. O fluxo de ambos os serviços é bastante simples, como é possível observar na seguinte imagem:

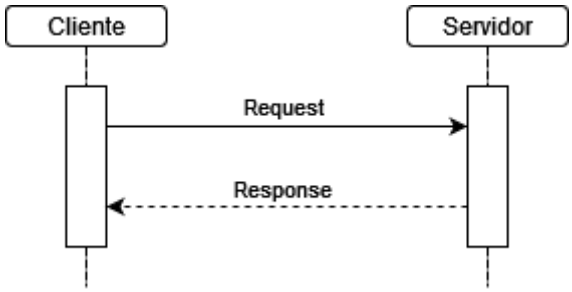

Figura 3.1 - Fluxo dos web services da Segurança Social

3.2 ERP ARTSOFT

Devido a desenvolvimentos anteriores, a aplicação ERP ARTSOFT já tem classes preparadas para efetuar codificações em Base64 e pedidos *http* que vão ser utilizadas e estendidas neste projeto. A partir da classe SSConnect, que estende a classe WSConnect servindo como *wrapper* da mesma, é possível definir o cabeçalho e o corpo da mensagem a enviar aos serviços da Segurança Social. Adicionalmente, para efetuar um pedido é necessário fornecer à classe SSConnect dois objetos que representem o pedido e a resposta do serviço. Para o pedido do serviço iremos utilizar a classe SSRequest e a classe SSResponse para representar a resposta do serviço; estas classes implementam as classes abstratas WSRequest e WSResponse respetivamente. É possível visualizar esta estrutura no seguinte diagrama de classes:

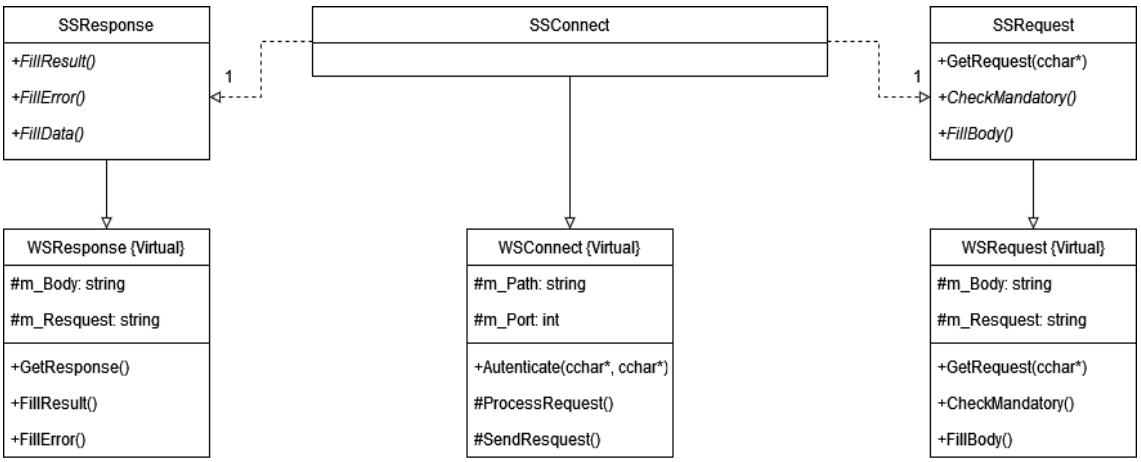

Figura 3.2 - Diagrama de Classes UML (Web Services)

3.3 Estrutura da Especificação de Requisitos

A nossa especificação de requisitos vai seguir o seguinte formato:

1. Introdução
2. Sumário
3. Requisitos
4. Informação Adicional
5. Documentos de referência e glossário
6. Testes

Na introdução são apresentadas as necessidades do cliente de forma a contextualizar o problema, são indicados os módulos que vão ser afetados pelo desenvolvimento, e por fim, é apresentado o coordenador e programador do projeto, os membros da equipa de teste, o custo estimado e uma estimativa do tempo que irá demorar a desenvolver a solução e a sua margem de erro. No sumário são resumidos em termos técnicos as necessidades do desenvolvimento e as funcionalidades chave a implementar. Na secção dos requisitos são descritos em detalhe, individualmente, os requisitos do desenvolvimento. A estes requisitos são também atribuídos um grau de importância e o esboço das interfaces relacionadas - por exemplo, se um requisito criar a necessidade de implementação de uma nova interface, isso é explicado em detalhe sendo apresentado um esboço da interface a ser desenvolvida e uma descrição das funcionalidades de todos os botões e campos interativos. No contexto da aplicação ERP ARTSOFT, esta secção contém mais um campo referente a possíveis esboços de relatórios novos a integrar na aplicação. No segmento da informação adicional, como o nome indica, são indicados quaisquer pressupostos e informação adicional que seja relevante para o desenvolvimento. De seguida são apresentados todos os documentos de referência e o glossário, concluindo o documento com a lista de testes aos quais o desenvolvimento vai ser submetido.

3.4 Web Service

Depois de a aplicação ter submetido o pedido ao *web service*, o mesmo pode receber quatro diferentes respostas do serviço, cada uma com a sua própria assinatura *WSDL*; o valor ‘0’, que indica que ocorreu um erro interno no servidor; o valor ‘1’, indicando que a operação foi realizada com sucesso; o valor ‘2’ representa que houve falta de parâmetros obrigatórios; a falha de validação dos dados enviados é indicado pelo valor ‘3’; ambos os resultados ‘2’ e ‘3’ vêm acompanhados por uma mensagem de erro descritiva para indicar ao utilizador a causa do erro. Contudo, de notar que só é indicado um erro de cada vez, logo, para uma experiência de utilizador mais suave e eficiente, o ideal será a aplicação validar os campos enviados antes de efetuar o pedido à Segurança Social.

3.5 Comunicar Vínculo de Contrato do Trabalhador

3.5.1 Documentação

O primeiro serviço a ser analisado é o de comunicar o vínculo de contrato do trabalhador à Segurança Social. O corpo do pedido contém toda a informação a ser comunicada, sendo necessária uma análise detalhada do mesmo:

- NISS do trabalhador
- Data de nascimento
- Data de início de contrato
- Data de fim de contrato
- Prestação de contrato
- Código de profissão
- Local de trabalho
- Modalidade de contrato
- Motivo de contrato
- Remuneração base mensal ilíquida
- Diuturnidades
- Número de horas de trabalho
- Número de dias de trabalho
- Percentagem de trabalho
- NISS do trabalhador a substituir

Destes elementos, vários são de comunicação obrigatória ao serviço: NISS do trabalhador, data de nascimento, data de início de contrato, código de profissão, local de trabalho, modalidade de contrato e remuneração base. Em alguns casos de uso, alguns dos outros elementos passam a ser de comunicação obrigatória, dependendo dos valores comunicados em certos elementos. É também necessário ter em conta o formato de cada elemento a ser comunicado: o número de identificação da Segurança Social tem de ser um número com onze dígitos e tem de ser um número válido e registado na Segurança Social; as datas de nascimento, início de contrato e fim de contrato tem de obedecer ao formato AAAA-MM-DD; além disso, a data de nascimento do trabalhador tem de corresponder à data registada na Segurança Social; a data de fim de contrato tem de ser igual ou posterior à data de início de contrato sendo que a mesma não pode ser doze meses anterior à data atual nem sete dias depois da data de utilização do serviço; a prestação de contrato é um campo opcional representado por um só caracter ‘P’ ou ‘T’, (“Presencial” e “Teletrabalho” respetivamente) este campo é único no aspeto que é o único em que a Segurança Social assume o valor “P – Presencial” caso não seja comunicado; o código de profissão é representado por um *array* de seis caracteres, onde os valores aceites correspondem a valores tabelados pela Classificação Portuguesa das Profissões; o local

de trabalho é um número até quatro dígitos, único para cada estabelecimento, fornecido pela Segurança Social; a modalidade e motivo de contrato do trabalhador são dois campos que aceitam uma *string* até quatro caracteres; os possíveis valores encontram-se indicados em duas diferentes tabelas presentes na documentação do *web service* fornecidas pela Segurança Social; a remuneração base mensal e diuturnidades são representadas por doze dígitos, aceitando duas casas decimais; o número de horas de trabalho e o número de dias de trabalho são representados por seis dígitos com duas casas decimais; a percentagem de trabalho aceita até cinco dígitos com duas casas decimais; e por fim, o número de identificação da Segurança Social do trabalhador a ser substituído é representado por um número de onze dígitos que corresponda a um número de identificação da Segurança Social válido e presente no sistema da Segurança Social.

3.5.2 Escrita da Especificação

Como indicado na secção 1.1, no período em que este desenvolvimento foi iniciado a Segurança Social anunciou o descontinuar do serviço Registar Vínculo tendo-o substituído por um serviço mais completo e robusto. Uma limitação da implementação atual da funcionalidade presente na aplicação ERP ARTSOFT é que só permite ao utilizador pré-visualizar a informação a ser enviada. Um utilizador que pretenda alterar ou corrigir algum dos dados tem de sair do ecrã existente e ir alterar diretamente na ficha do trabalhador. A adição da capacidade de edição dos dados a serem comunicados levanta a possibilidade da comunicação de informação inválida e ,embora estes erros sejam identificados pelo serviço da Segurança Social, foi decidido, com o objetivo de promover uma melhor experiência de utilizador, efetuar a validação dos campos a serem comunicados só permitindo ao utilizador efetuar o pedido ao serviço caso todos os campos se encontrem válidos. Depois de efetuado o pedido, é necessário apresentar a resposta do mesmo ao utilizador; em caso de sucesso, dado que os dados enviados podem ser diferentes dos presentes na ficha do trabalhador na base de dados, é necessário atualizar a mesma com essa nova informação. Por fim, para auxiliar a deteção e tratamento de erros depois de a funcionalidade ser lançada para o mercado, foi decidido implementar a escrita de um ficheiro log que contenha toda a informação comunicada no serviço. Este ficheiro será escrito antes de a aplicação comunicar com o serviço.

Dados o contexto, o problema e objetivos descritos, foram identificadas as seguintes funcionalidades chave:

a) Recolher os dados a enviar e apresentação dos mesmos no novo ecrã;
b) Permitir a alteração do valor dos campos existentes por parte do utilizador;
c) Atualizar a base de dados em caso de sucesso;
d) Apresentar um pedido de confirmação de envio;

e) Comunicar o vínculo de contrato utilizando o *web service* disponibilizado pela Segurança Social;
f) Apresentar o resultado da operação.

E os seguintes requisitos funcionais com os seus graus de importância:

a) Pré-visualização dos campos a enviar – Crítica;
b) Validação dos campos – Alta;
c) Escrita de um ficheiro log – Baixa;
d) Comunicação com o *web service* – Crítica;
e) Propagar campos alterados para a base de dados – Alta.

Para a pré-visualização dos campos a serem comunicados, e respetivos valores, foi decidido utilizar uma nova interface em que o foco é a apresentação e edição de dados. Na imagem seguinte encontra-se um esboço da interface a implementar:

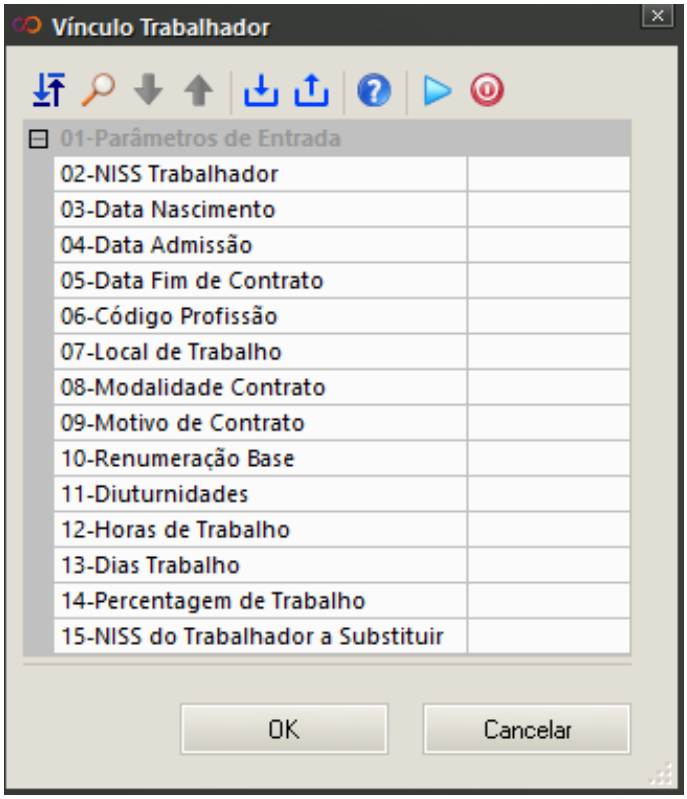

Figura 3.3 - Esboço Interface Vínculo Trabalhador

Como é possível observar, a interface irá apresentar ao utilizador todos os campos que podem ser comunicados. Alguns destes campos irão ter restrições nos valores que aceitam, dadas as restrições descritas na documentação; alguns campos também se encontrarão bloqueados dadas as opções selecionadas pelo utilizador. Depois de o utilizador pré-visualizar e, possivelmente, editar os dados, deve clicar no botão "OK"; de seguida é apresentada uma mensagem de confirmação. Após o utilizador confirmar, o pedido ao *web service* é efetuado. A resposta recebida é apresentada e, em caso de sucesso, guardam-se na base de dados quaisquer alterações efetuadas no ecrã.

3.6 Cessar Vínculo do Trabalhador

3.6.1 Documentação

Neste serviço, tal como no serviço previamente analisado, o corpo do pedido contém toda a informação a ser comunicada:

- NISS do trabalhador
- Data de fim de contrato
- Motivo de contrato
- Comunicação de desemprego
- Fundamentação

Com exceção do motivo de contrato e da fundamentação, todos os campos são obrigatórios. Quanto ao relacionamento entre campos, o campo fundamentação passa a ser obrigatório com certos valores dos campos motivo de contrato e comunicação de desemprego. Tal como no serviço da comunicação do vínculo de contrato, também é aqui importante ter em conta o formato de cada campo: o número de identificação de Segurança Social tem de ter 11 dígitos e ser um número registado no sistema; a data fim de contrato obedece ao formato AAAA-MM-DD e tem de ser igual ou inferior, até 60 meses, à data de utilização do serviço; o motivo de fim de contrato é representado por um array de até 4 caracteres, sendo que os valores válidos encontram-se tabelados na documentação fornecida; o campo comunicação de desemprego é binário, só aceitando os valores '0' e '1' significando "Não" e "Sim", respetivamente; por último o campo fundamentação é um conjunto de até 500 caracteres.

3.6.2 Escrita da Especificação

Como mencionado previamente, este serviço é bastante semelhante ao anteriormente descrito, por isso foi decidido implementar o mesmo ecrã para pré-visualização dos dados a serem comunicados, validando os mesmos para que o utilizador não faça um pedido com informação inválida. Depois de enviado o pedido à Segurança Social, também é

apresentada uma mensagem com a resposta do serviço. É igualmente efetuada a criação de um ficheiro *log* para um mais fácil tratamento de possíveis futuros erros.

Com o seguinte contexto, problema e objetivos em mente foram identificadas as seguintes funcionalidades chave:

a) Recolher os dados a enviar e apresentação dos mesmos no novo ecrã;
b) Permitir a alteração do valor dos campos existentes por parte do utilizador;
c) Atualizar a base de dados em caso de sucesso;
d) Apresentar um pedido de confirmação de envio;
e) Comunicar o vínculo de contrato utilizando o web service disponibilizado pela Segurança Social;
f) Apresentar o resultado da operação.

E os seguintes requisitos funcionais com os seus graus de importância:

a) Pré-visualização dos campos a enviar – Crítica;
b) Validação dos campos – Alta;
c) Escrita de um ficheiro log – Baixa;
d) Comunicação com o *web service* – Crítica;
e) Propagar campos alterados para a base de dados – Alta.

Como mencionado previamente, a interface utilizada será a mesma do serviço Comunicação de Vínculo do Trabalhador, sendo possível visualizar o esboço da mesma na seguinte imagem:

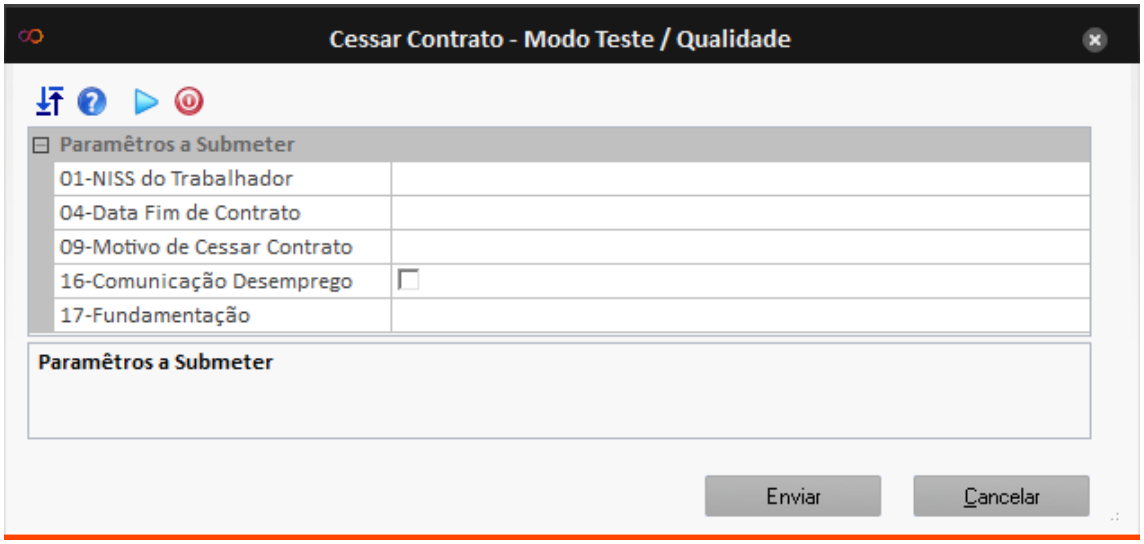

Figura 3.4 - Esboço Interface Cessar Contrato

Todas as liberdades e limitações descritas para o serviço Comunicação de Vínculo do Trabalhador aplicam-se a esta interface: a interface apresenta e permite a edição de todos os campos passíveis de comunicação pelo serviço; o *input* é limitado de acordo com

os valores esperados para cada campo; o envio do pedido para o serviço só será possível depois de todos os campos passarem as validações definidas pelo serviço. Depois do utilizador clicar no botão ”OK”, é-lhe apresentada uma mensagem para confirmar o seu desejo de enviar o pedido ao serviço. Uma vez enviado o pedido, o utilizador é informado do resultado do mesmo. Por fim, quaisquer alterações que o utilizador tenha efetuado na interface serão propagadas para a base de dados da aplicação ARTSOFT.

3.7 System Sequence Diagram (SSD)

Para este desenvolvimento iremos apresentar somente um SSD visto que o fluxo do mesmo se aplica para ambos os serviços. O utilizador irá informar a aplicação que pretende comunicar ou cessar o vínculo de trabalho; serão recolhidos os dados presentes na base de dados e apresentados ao utilizador; o utilizador terá a oportunidade de pré-visualizar e editar os dados a serem comunicados. A aplicação irá processar e converter estes dados para o formato documentado; será depois efetuado o pedido ao serviço e processada a resposta; essa resposta será apresentada ao utilizador e em caso de sucesso qualquer alteração que o utilizador tenha efetuado na interface será guardada na base de dados. Um cenário onde o pedido é efetuado com sucesso pode ser representado pelo seguinte diagrama:

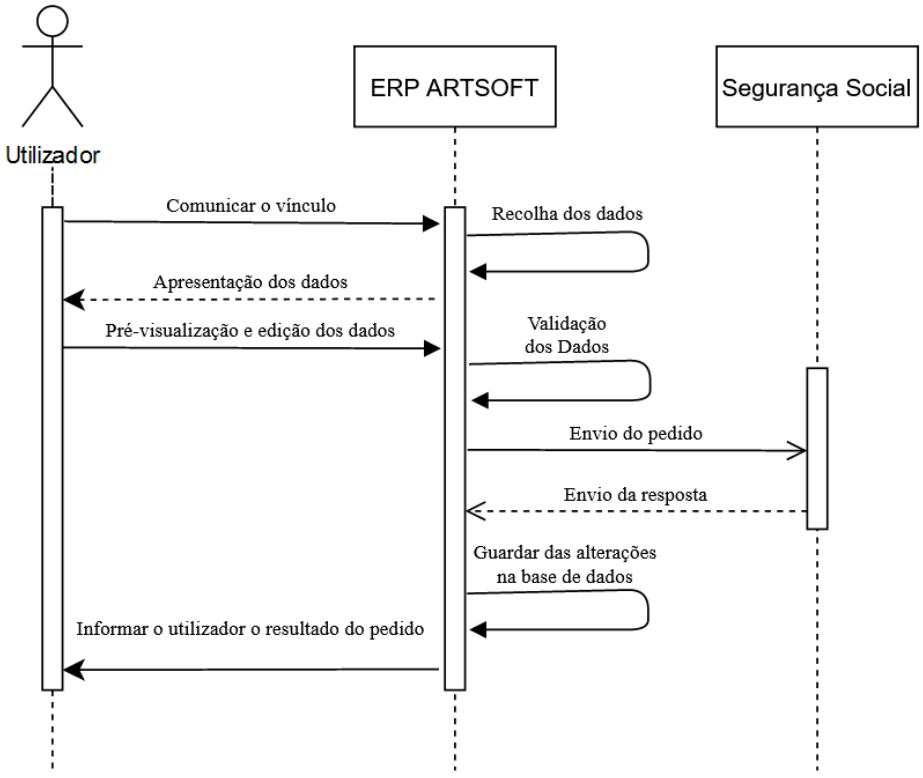

Figura 3.5 - System Sequence Diagram (SSD)

Capítulo 4

Desenho da Solução

Passaremos agora a descrever algumas das decisões de desenho da solução, começando pelos dados a comunicar ao serviço.

4.1 Tabelas

Dos dados envolvidos nos serviços em questão, há alguns que não se encontram presentes nas bases de dados do ERP ARTSOFT. Será necessário então primeiro identificar quais os campos a serem comunicados que não são guardados na base de dados de ERP ARTSOFT e desses mesmos campos quais fazem sentido guardar na base de dados para possível utilização futura. Relativamente ao 1º serviço, da informação pedida, somente dois dos campos não são atualmente guardados na base de dados: o motivo de contrato e o NISS do trabalhador a substituir; destes dois campos o NISS do trabalhador a substituir não é informação que vá ser utilizada noutras funcionalidades da aplicação, significando que não se justifica a criação de uma entrada na base de dados para guardar esta informação. O motivo de contrato já é um valor que vamos querer guardar na base de dados dado a sua potencial futura utilização. O serviço da comunicação da cessação do vínculo de contrato tem três campos que não são guardados na base de dados: o motivo de cessar contrato, comunicação para desemprego e fundamentação; destes três campos dois deles são exclusivos ao serviço, não tendo outra utilização fora do *web service*. O motivo de cessar contrato é um campo que vamos querer passar a guardar na base de dados devido à sua potencial futura utilização noutras funcionalidades.

É então necessário adicionar dois campos à ficha do trabalhador e criar duas tabelas, uma que guarde o motivo de vínculo de contrato e outra para os motivos de cessar contrato. Estes campos na ficha do trabalhador irão guardar a chave primária da respetiva tabela no formato do tipo *ushort* que permite guardar um valor numérico ocupando o mínimo de espaço possível. Aqui surge a primeira decisão de desenho do nosso projeto: como concretizar as novas tabelas. O conteúdo de cada tabela é definido pela Segurança Social na documentação técnica dos serviços. Isto leva-nos a duas escolhas possíveis: implementar a tabela no código ou criar uma tabela na base de dados que guarde estes valores. A primeira opção tem várias vantagens: a tabela ser fixa levaria a uma melhor performance dado que é removido o *overhead* de acesso à base de dados e também impediria qualquer alteração em tempo de execução. Contudo, esta implementação implica que, caso a Segurança Social decida alterar o conteúdo desta tabela, é necessário

lançar uma atualização de produto. A segunda opção iria permitir a alteração do conteúdo da tabela, removendo a necessidade de lançar uma atualização de produto caso a Segurança Social atualizasse o conteúdo da tabela, porém poderia levar a que utilizadores inexperientes alterassem o conteúdo e levassem ao não funcionamento do serviço. Foi decidido que, devido aos custos da primeira opção não serem sustentáveis, seria criada uma tabela na base de dados para guardar os motivos de vínculo de contrato e outra para os motivos de cessar contrato. Quanto às regras que limitam esta interface, foi decidido impedir o utilizador de manipular qualquer entrada da tabela, permitindo somente alterar informação utilizando as funcionalidades de importar e exportar da interface dado que os ficheiros de importe são criados pela empresa ARTSOFT, garantindo que os mesmos se encontram corretos. As nossas tabelas têm a seguinte estrutura:

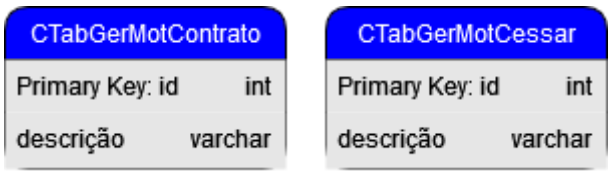

Figura 4.1 - Estrutura Tabelas

4.2 Interface

Relativamente à interface a utilizar para a pré-visualização dos dados, foi decidido utilizar a classe "GenEditHdr". Esta classe já se encontra presente em várias funcionalidades da aplicação ERP ARTSOFT e está desenhada para pré-visualização e edição de dados, oferecendo também um alto nível de diversidade quanto ao tipo de *input* que o campo aceita, desde simples campos de edição de texto até algo mais complexo como um *browser* de ficheiros, e a possibilidade de validação dos campos durante a edição com a classe de *callback* "GenEditCBack".

4.3 Estruturas de Dados

Para guardar os dados a serem utilizados pela interface e para serem comunicados pelos serviços da Segurança Social, foi decidido criar uma estrutura de dados para cada serviço. Cada estrutura vai conter uma variável que corresponde a cada campo a ser comunicado ao serviço.

4.4 Log

Foi decido implementar a escrita de um ficheiro *log*, depois do utilizador confirmar os dados a enviar, permitindo a mais fácil análise dos dados a serem comunicados. Isto vai permitir um programador identificar e corrigir potenciais erros mais rapidamente. Este

ficheiro será criado numa pasta cujo propósito é guardar ficheiros relacionados com a Segurança Social, com um nome que indique o serviço que o criou e com um *timestamp* da hora e dia que foi criado.

A escrita deste ficheiro será efetuada com a classe "CFileStream", uma classe que encapsula as funções de escrita base do C++, oferecendo ao programador uma biblioteca mais fácil de utilizar e de efetuar correção de erros. Este ficheiro vai ser escrito linha a linha, contendo cada uma variável da estrutura de dados, escrevendo todos os campos até os que se encontram vazios.

4.5 Diagrama de Classes

Já refletindo as decisões tomadas o seguinte, este é o nosso diagrama de classes para este projeto:

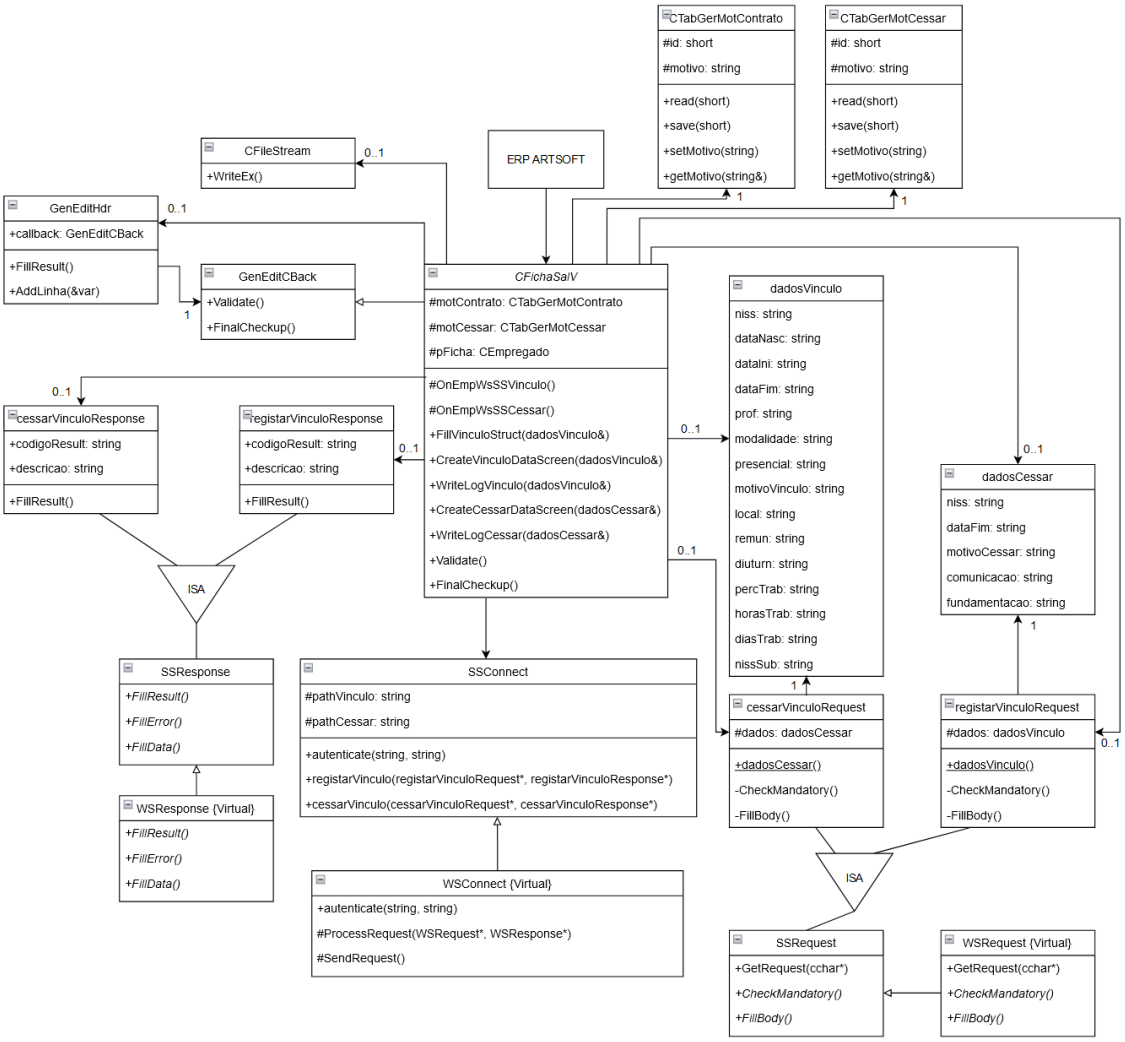

Figura 4.2 - Diagrama de Classes UML (Desenho)

Capítulo 5

Implementação

Dado que os serviços a implementar têm semelhanças, abordaremos aqui de modo a uniformizar tudo o que é comum, intercalando algumas partes de discussão individual nos aspetos únicos de cada desenvolvimento.

5.1 Tabelas

A implementação das duas tabelas envolveu não só a criação das classes que representam cada tabela como também os métodos e IDs para que a aplicação consiga ter acesso às tabelas onde seja necessário. Começando pelas classes que representam as nossas tabelas, foram implementadas "CTabGerMotContrato" e "CTabGerMotCessar"; ambas estas classes herdam da classe "CTabelasStd" que é a classe base de todas as tabelas no ERP ARTSOFT. A classe "CTabelasStd" já tem implementados todos os métodos que precisamos para comunicar com a base de dados sendo que utiliza um ID que está associado à tabela para distinguir entre diferentes tabelas. Dada a estrutura das tabelas, só precisamos de implementar métodos de *get* e *set* que depois chamam os métodos já implementados de leitura e escrita na base de dados de "CTabelasStd"; para completar as nossas tabelas, é necessário definir os seus métodos de importar e exportar o conteúdo das mesmas. Os métodos de importe e exporte de tabelas são globais, ambos recebendo somente um apontador de memória para a tabela a ser importada ou exportada e o caminho de onde se pretende importar ou exportar o ficheiro. Utilizando o ID associado à tabela e um *switch,* é identificado o método de importe ou exporte a utilizar.

Uma vez criadas as classes, é necessário criar a interface, e o acesso à mesma, que vai permitir ao utilizador modificar as tabelas. Vamos alterar o menu do ecrã principal para ter mais duas entradas, uma para cada tabela. Estes campos terão um ID associado que irá permitir ao programa identificar qual a tabela que o utilizador pretende consultar. Estes novos ecrãs serão gerados pela *framework* MFC que implementa a classe "CDialog", classe que por sua vez é herdada pela classe "CDlgCheckMultiselC". Para criar este ecrã vamos utilizar a classe "CEditTabelasBaseGer" que recebe um objeto base de dados, como por exemplo um das nossas novas tabelas, e o ID associado a essa mesma tabela; com isto ele cria uma ligação ao *handler* da tabela e um conjunto de parâmetros que define a tabela como, por exemplo, os dados a apresentar e um objeto de *callback* que neste caso será o próprio objeto CEditTabelasBaseGer. Isto implica a necessidade de implementar um método que recolha todas as entradas da tabela num formato definido

pela classe que irá criar o ecrã que o utilizador irá visualizar: CDlgCheckMultiselC. Estes são os métodos que implementamos previamente e são chamados pelo *callback*, ou seja, pela classe CEditTabelasBaseGer. Para completar esta parte do desenvolvimento, falta adicionar um campo no ecrã da ficha do colaborador para permitir ao utilizador alterar os novos campos que foram adicionados na ficha do colaborador. Primeiro temos de alterar o ecrã atual, adicionando duas novas *combo boxes*, uma para cada campo. Uma vez adicionadas as *combos boxes*, é necessário adicionar o controlador do objeto à classe responsável pela janela, sendo essa "CEmpregIdent"; ao iniciar a classe é necessário ligar o controlador à *combo box* utilizando o ID associado ao mesmo. De seguida temos de preencher a combo com o conteúdo da nossa tabela, percorrendo e, por cada campo da mesma, utilizar o método "AddStringAndData" do controlador. Este permite mostrar ao utilizador só a descrição do motivo, mas associando a *primary key*, para depois guardar na ficha do utilizador sem ter de recorrer outra vez à tabela.

5.2 Web Services

O passo seguinte no desenvolvimento será implementar a interface com a qual o utilizador irá interagir de modo a pré-visualizar os dados a serem comunicados pelo serviço e a enviar o pedido para o *web service* da Segurança Social.

5.2.1 Request

Iremos primeiro criar a classe que representa o pedido ao serviço. Esta classe herda de "SSRequest" que implementa a classe abstrata "WSRequest". Vamos aproveitar esta classe para definir uma estrutura de dados que irá guardar toda a informação possível de ser comunicada. De seguida precisamos de implementar os dois métodos herdados da classe abstrata: "CheckMandatory", que irá simplesmente devolver 0 dado que todas as verificações serão tratadas pela interface, e "FillBody", que devolve o corpo da nossa mensagem que irá ser construído aos pedaços. Inicialmente a mensagem irá conter todos os campos obrigatórios, adicionando um a um os campos opcionais caso os mesmos se encontrem preenchidos na estrutura de dados. Esta estrutura vai ser inicialmente preenchida recolhendo informação das tabelas e depois utilizada pela interface a apresentar ao utilizador.

5.2.2 Interface

Tal como as janelas para visualizar as tabelas da base de dados, a interface para a pré-visualização e edição dos dados a comunicar vai ser criada a partir da *framework* MFC com a classe "CDialog". Vamos então precisar primeiro de definir os parâmetros da interface, contudo para criar a interface do esboço precisamos de utilizar um conjunto diferente de classes e métodos. Para adicionar campos à interface utilizamos o método

"AddLinha" da nossa classe "GenEditHdr" que tem duas assinaturas diferentes dependendo do tipo de campo que estamos a tentar adicionar. A principal diferença entre as assinaturas é se o método recebe uma referência de memória para uma variável; caso não receba, a interface interpreta como um nó de árvore que o utilizador pode abrir ou fechar, senão a interface interpreta como um campo que tem dados para apresentar ao utilizador. O facto de utilizar uma referência implica que qualquer alteração que o utilizador efetue na interface vai automaticamente refletir na variável fazendo as variáveis da estrutura de dados que criamos perfeitas para utilizar como parâmetro do método "AddLinha". Este método também permite definir um ID para cada campo; isto será particularmente útil para quando formos implementar as validações para, não só distinguir qual é o serviço que estamos a pré-visualizar, como também para ter acesso aos valores de outros campos da interface a partir do "FindElemByID"; por isso vamos definir um enumerado que representa o ID de cada campo a ser apresentado pela interface.

Por fim, vamos querer definir o *callback* da interface; isto vai permitir implementar validações quando o utilizador tira o foco de certos campos e quando o utilizador tenta submeter os seus dados. Primeiro vamos ter de alterar a classe "CFichaSalV" para herdar a classe de *callback* de "GenEditHdr", depois definir o próprio objeto "CFichaSalV" como o *callback* do objeto "GenEditHdr", e por fim implementar o comportamento dos métodos de validação da interface. Estes dois métodos são herdados da classe "GenEditCBack": "FinalCheckup" é chamado quando o utilizador tenta submeter os dados (em caso de erro é devolvido o ID de um dos campos que contém um erro); o "Validate" é chamado quando campos que contêm a *flag* "KillFocus_CB" têm o seu valor alterado. Uma vez definido o objeto "GenEditHdr" vamos utilizar o mesmo como parâmetro do método global "GenericEditTool" que processa estes parâmetros para um formato que a classe "CDialog" consiga utilizar.

Uma vez pré-visualizados e editados os dados, vamos iniciar uma transação contendo as alterações efetuadas pelo utilizador; aproveitando o facto da classe "GenEditHdr" conter um método que verifica se o campo foi alterado permitindo evitar alteração de valores desnecessários na base de dados. Vamos também aproveitar para converter os dados que apresentamos ao utilizador para o formato requisitado pela Segurança Social.

5.2.3 Response

Para completar a implementação necessária para podermos comunicar com os serviços da Segurança Social, falta implementar as classes que recebem a resposta do serviço. Herdando da classe "SSResponse", além de guardar a resposta do serviço, esta classe é responsável por indicar o caminho do *web service* e por implementar a função "FillResult" que recebe uma resposta em formato XML e processa a mesma. A resposta XML é convertida para uma árvore de nós, cada nó guardando um valor e um nome; por

exemplo, pegando no nó raiz podemos pesquisar o nó filho que contém o código da resposta correspondendo ao resultado do serviço.

5.2.4 Comunicação

Com todas as preparações feitas podemos finalmente comunicar com o serviço. Vamos começar por autenticar o utilizador concatenando os detalhes de *log in* da Segurança Social que foram inseridos previamente no menu da configuração da empresa. Esta concatenação obedecerá ao seguinte formato: “utilizador:palavra-passe”. Esta *string* depois será codificada em Base64 e adicionada como parâmetro de autenticação do *header* do pedido *http*; juntamos o corpo, que foi construído pela nossa implementação da classe do pedido do serviço, ao *header* e temos uma mensagem preparada para enviar ao serviço. De seguida conectamos ao servidor da Segurança Social que *host* ao serviço, efetuando testes antes de enviarmos o pedido para garantir que a ligação foi efetuada com sucesso, e enviamos a nossa mensagem pelo *socket*. Este mesmo *socket* recebe a mensagem com a resposta do serviço e, analisando o código *http* de resposta, determinamos se ocorreu um problema com o serviço. Contudo se o código corresponder a um dos documentados na documentação, a mensagem de resposta é processada pelo método “FillResult”. Dada a resposta do serviço, apresentamos uma mensagem de erro ou sucesso do pedido ao utilizador, sendo que, em caso de erro, fazemos *rollback* à transação SQL que tínhamos iniciado, para cancelar as alterações que tínhamos submetido na base de dados, em caso de sucesso, fazemos *commit* à transação. Com a possibilidade de os dados da ficha de trabalhador terem sido atualizados forçamos a interface a recarregar a informação do trabalhador.

5.3 Diagrama de Classes

Uma vez implementada a nossa solução é possível desenhar o seguinte diagrama de classes:

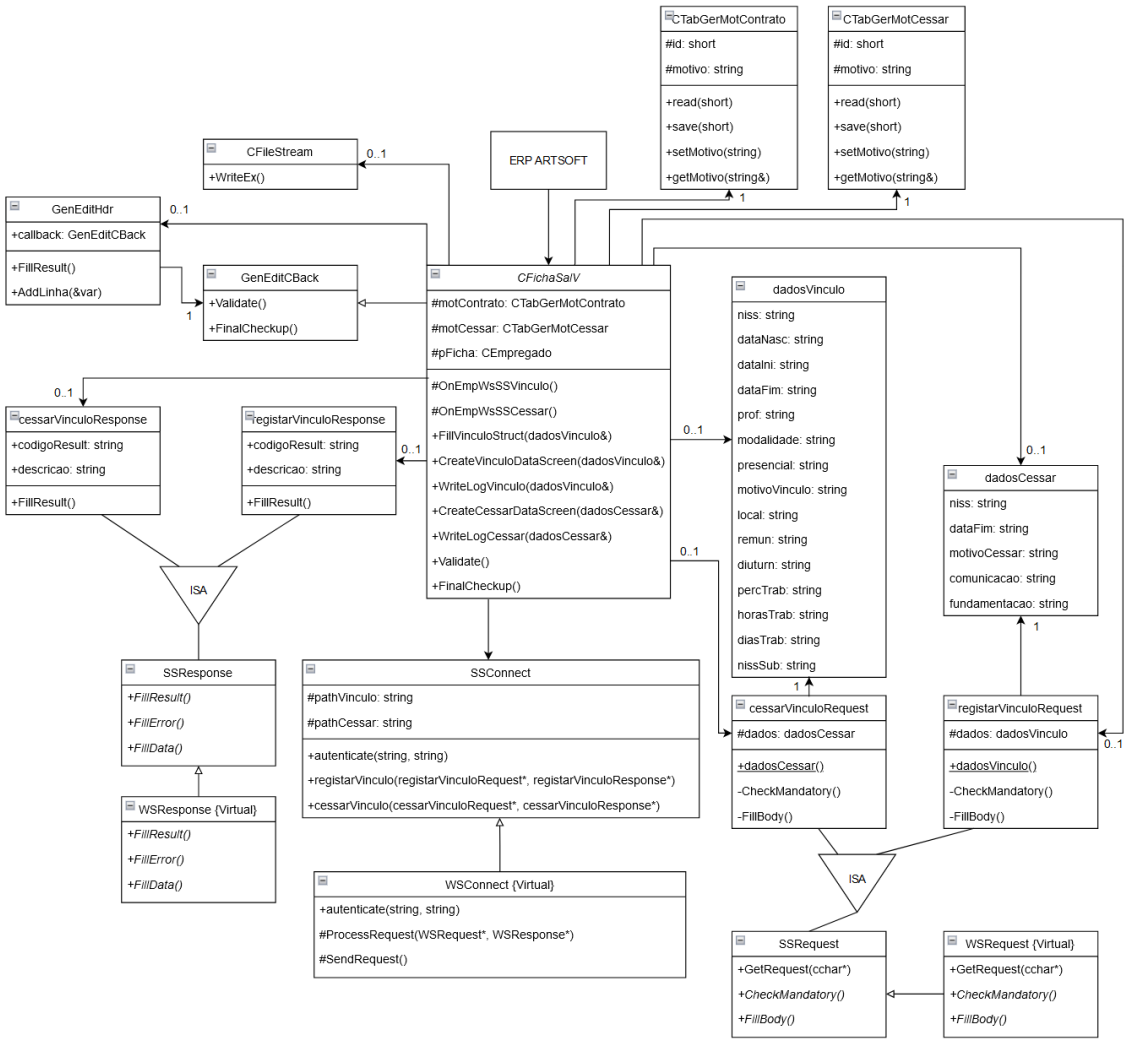

Figura 5.1 - Diagrama de Classes UML (Implementação)

Capítulo 6

Resultados

Neste capítulo vamos apresentar recortes do ecrã onde se pode ver o funcionamento do projeto na sua totalidade.

6.1 Tabelas

A configuração das tabelas implementadas faz-se a partir do mesmo menu acedido pelo ecrã principal do ERP ARTSOFT, como é possível observar na seguinte imagem:

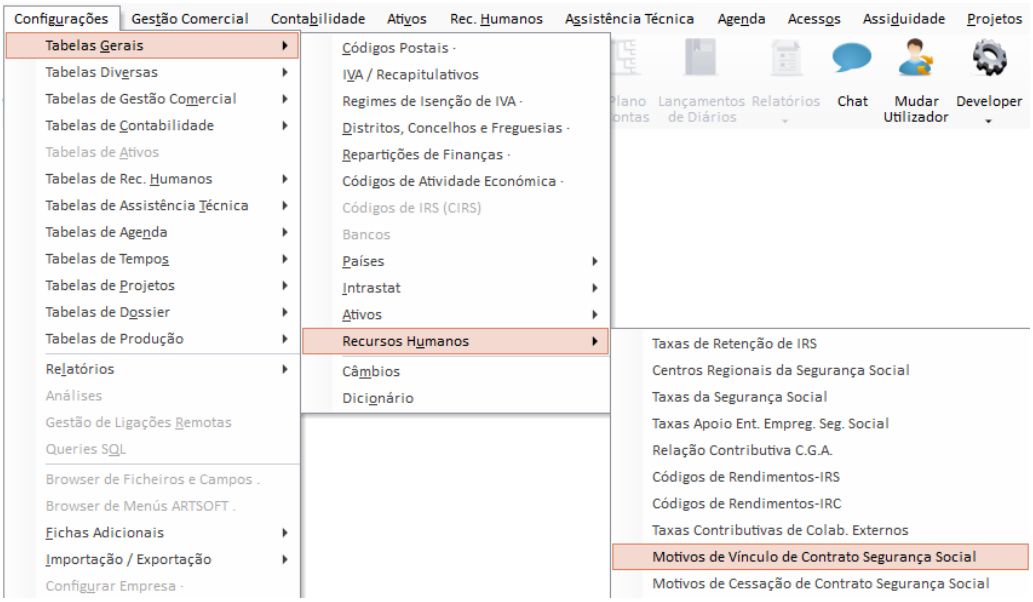

Figura 6.1 - Aceder às Tabelas

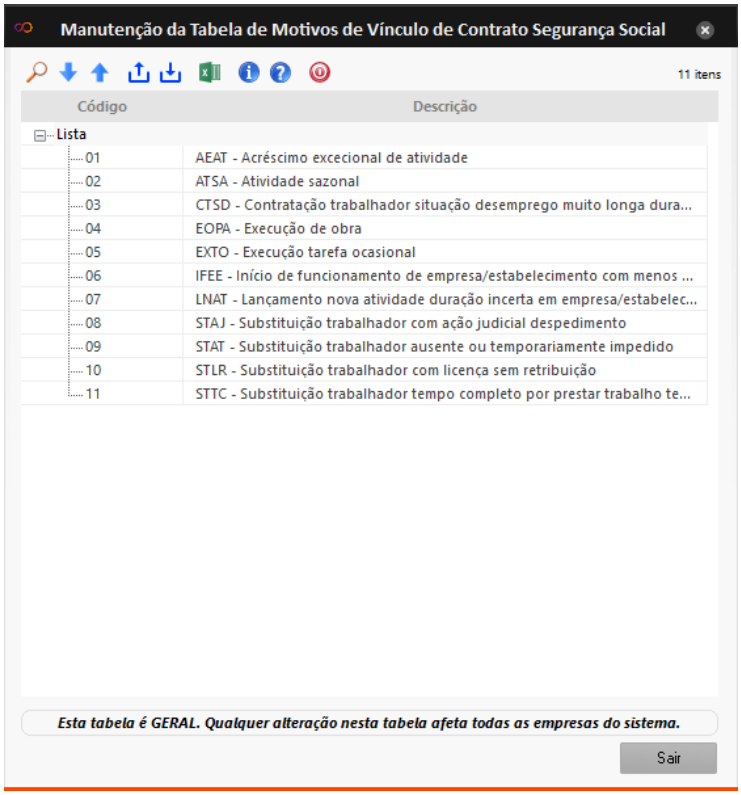

Figura 6.2 - Tabela de Motivos de Vínculo de Contrato

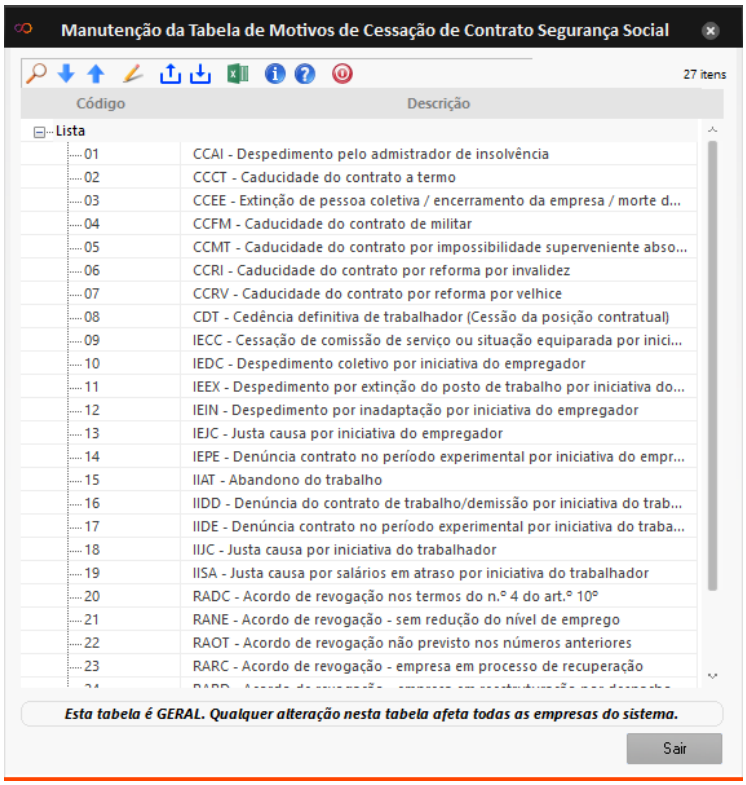

Figura 6.3 - Tabela de Motivos de Cessação de Contrato

Uma vez configuradas as tabelas, podemos utilizar os novos serviços:

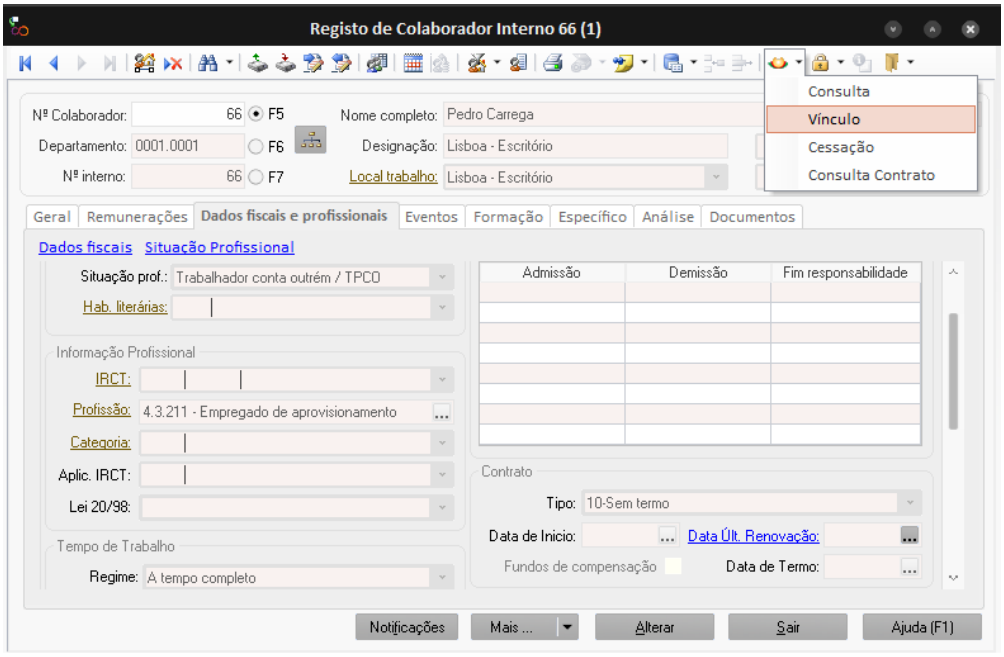

Figura 6.4 - Menu Serviços Segurança Social

Se selecionarmos o serviço de comunicar o vínculo:

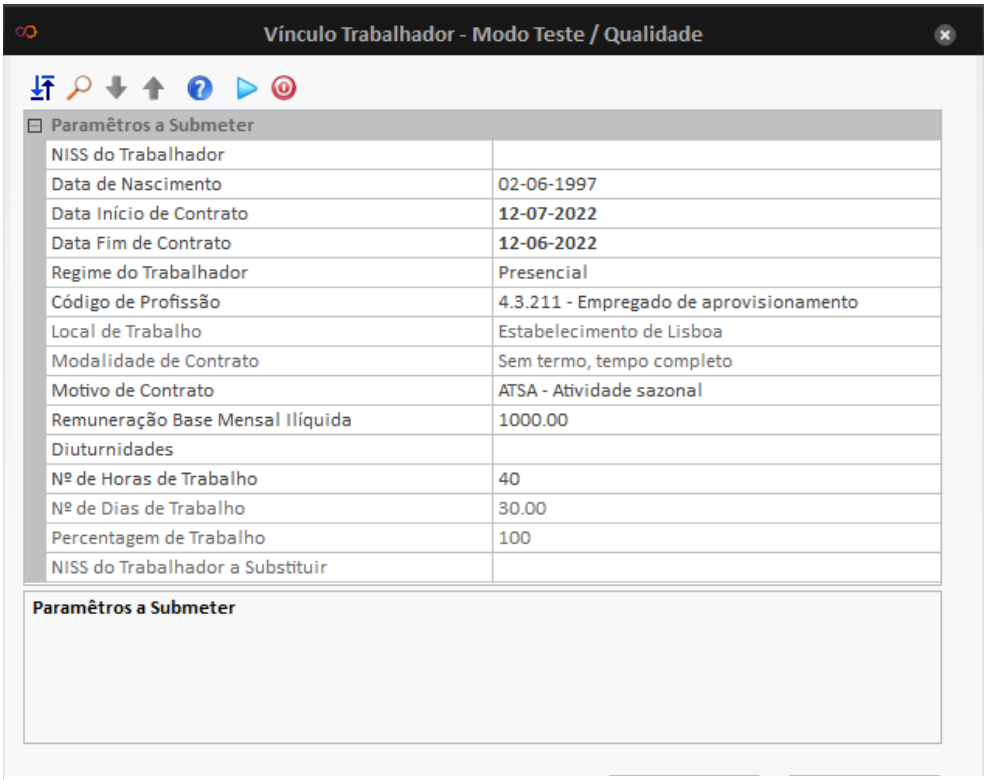

Figura 6.5 - Interface Vínculo Trabalhador

Caso o utilizador insira um valor inválido ou não insira um campo obrigatório aparece uma mensagem descritiva a indicar o erro:

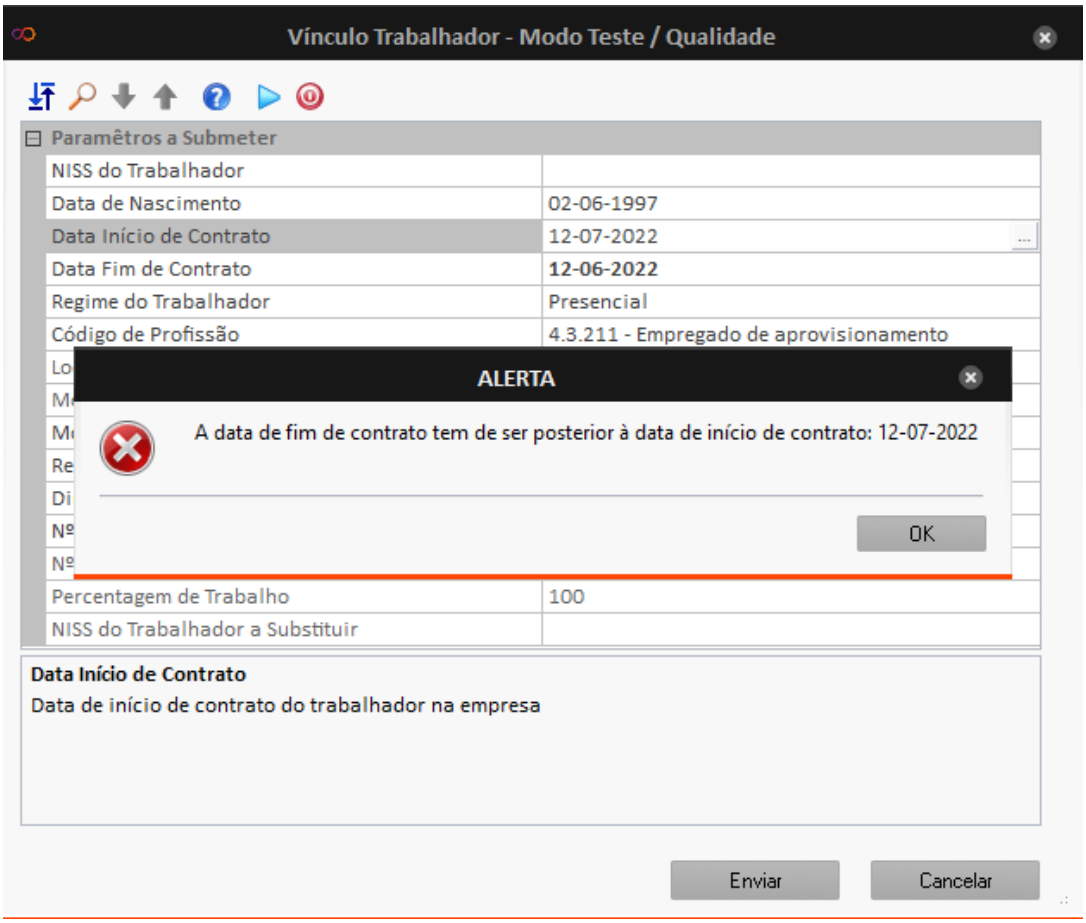

Figura 6.6 - Interface Erro na Validação

O serviço de cessação do vínculo de contrato tem a seguinte interface:

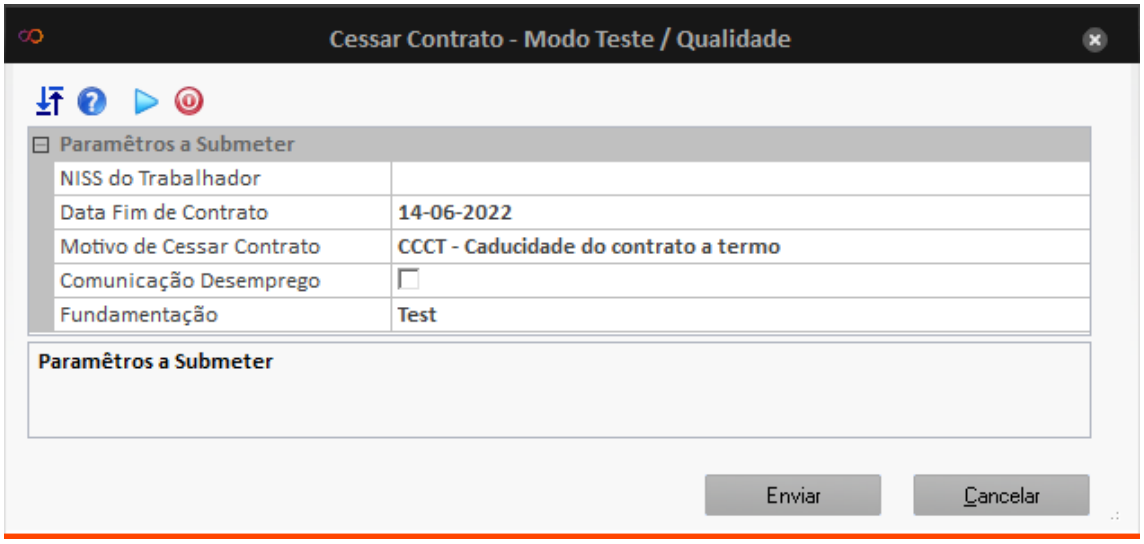

Figura 6.7 - Interface Cessar Contrato

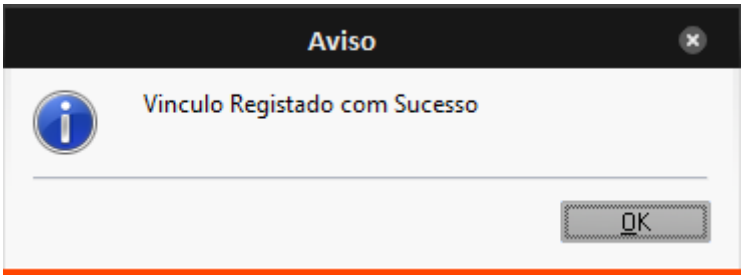

Figura 6.8 - Vínculo mensagem de sucesso

Uma vez apresentada uma mensagem de sucesso no serviço são guardadas quaisquer alterações que o utilizar tenha realizado na interface na base de dados.

Capítulo 7

Conclusão

Este projeto foi proposto pela empresa ARTSOFT à disciplina de Mestrado em Engenharia Informática da FCUL e teve como propósito o desenvolvimento de uma extensão de uma aplicação orientada a objetos. O trabalho deste projeto foi realizado durante um período de 9 meses na empresa ARTSOFT, na aplicação desenvolvida e comercializada internamente, ERP ARTSOFT, e teve como objetivo a integração de diferentes serviços desenvolvidos e disponibilizados pela Segurança Social. De modo a ir ao encontro desses objetivos, focou-se grande parte do tempo do projeto na realização de um estudo da aplicação e módulos relevantes para o desenvolvimento e da documentação técnica, disponibilizada pela Segurança Social, dos serviços a integrar na aplicação.

O módulo abrangido por este desenvolvimento foi o dos Recursos Humanos, que trata de todo o *front e backend* de todas as funcionalidades relacionadas com trabalhadores da empresa.

Para este projeto foi realizada a análise do problema e desenvolvida uma especificação de requisitos para cada serviço, apresentando toda a informação relevante para o desenvolvimento. Foi efetuado um estudo do problema, oferecendo contexto ao mesmo e indicando o comportamento desejado. Daí determinaram-se os principais objetivos do projeto, as funcionalidades chave e os requisitos funcionais a implementar no desenvolvimento, apresentando-se também esboços das interfaces a implementar.

Do estudo da documentação foi efetuado um desenho da solução para garantir que os objetivos fossem atingidos. Foi determinado que teria de ser adicionada informação à base de dados de ERP ARTSOFT para satisfazer todos os requisitos; foram adicionados novos campos à tabela da ficha do trabalhador e criadas duas tabelas para complementar estes novos campos. Foram implementadas as novas interfaces de utilizador esboçadas na especificação de requisitos, as quais permitem ao utilizador a pré-visualização e a edição dos dados a serem comunicados aos serviços e à aplicação identificar e informar o utilizador de erros no *input*.

A comunicação entre a aplicação e o servidor é efetuada a partir de um pedido *http* utilizando um formato SOAP XML. A autenticação é efetuada a partir do *header* "*autentication*", a partir da concatenação dos detalhes de acesso à Segurança Social codificando essa concatenação em base 64. A resposta ao serviço é efetuada no mesmo

formato, tendo sido implementado um *parse* seguindo a documentação disponibilizada; da resposta do serviço é apresentado ao utilizador sobre o resultado da operação.

7.1 Trabalho Futuro

No futuro irá ser efetuado trabalho de manutenção sobre os serviços integrados para garantir e melhorar a qualidade e funcionamento dos mesmos. Será também efetuado um estudo para analisar se existe interesse do mercado em integrar os outros serviços oferecidos pela Segurança Social e, caso esse interesse exista, esses serviços serão integrados na aplicação ERP ARTSOFT.

Bibliografia

Davis, Alan, Scott Overmyer, Kathleen Jordan, Joseph Caruso, Fatma Dandashi, Anhtuan Dinh, Gary Kincaid, et al. 1993. “Identifying and Measuring Quality in a Software Requirements Specification.” In *[1993] Proceedings First International Software Metrics Symposium*, 141–52. Ieee.

Larman, Craig. 2005. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*. Pearson Education India.

Benslimane, Djamal, Schahram Dustdar, and Amit Sheth. 2008. “Services Mashups: The New Generation of Web Applications.” IEEE Internet Computing 12 (5): 13–15.

Lee, Siew Poh, Lai Peng Chan, and Eng Wah Lee. 2006. “Web Services Implementation Methodology for SOA Application.” In *2006 4th IEEE International Conference on Industrial Informatics*, 335–40. https://doi.org/10.1109/INDIN.2006.275822.

Pilato, C. Michael, Ben Collins-Sussman, and Brian W. Fitzpatrick. 2008. *Version Control with Subversion - the Standard in Open Source Version Control*. O’Reilly. http://www.oreilly.de/catalog/9780596510336/index.html.

Vinoski, S. 2002. “Where Is Middleware.” *IEEE Internet Computing* 6 (2): 83–85.

Vinoski, S. 2003. “Integration with Web Services.” *IEEE Internet Computing* 7 (6): 75–77. https://doi.org/10.1109/MIC.2003.1250587.