Faturação de IVA – Parte 1

Algoritmos e Estruturas de Dados
MIEIC – 2015/16

Andreia Rodrigues – up201404691 – u@fe.up.ptp201404691
Diogo Cepa – up201403367 – up201403367@fe.up.pt
Eduardo Leite – gei12068 - gei12068@fe.up.pt

INDICE

Tema de Trabalho

Neste trabalho realizou-se um programa para gerir dados relativos a faturações de IVA. Este sistema é baseado em contribuintes, contribuintes estes que são identificados por um NIF (número de identificação fiscal).

O NIF é um numero de 9 dígitos onde os 8 primeiros são sequenciais e o ultimo é um digito de controlo formado da seguinte forma:

- Multiplicando o 1º digito por 9, o 2º por 8, o 3º por 7, o 4º por 6, o 5º por 5, o 6º por 4, o 7º por 3 e, por fim, o 8º por 2.
- Somando os resultados obtidos anteriormente.
- Calculando o resto da divisão inteira do resultado da soma anterior por 11.
- Se o resto for 0 ou 1, então o dígito de controlo é 0.
- Se for outro algarismo (X), o dígito de controlo irá ser 11 – X.

O contribuinte irá ter faturas associadas ao seu NIF, dependendo se este é emissor ou consumidor, se for emissor o contribuinte tem de emitir declarações todos os meses. Estas declarações podem ser recusadas caso o valor nelas indicado não corresponda ao valor total das faturas emitidas pelo contribuinte em causa.

Um contribuinte pode ser simultaneamente consumidor e emissor, e um emissor pode ainda ser ou não passivo de IVA.

SOLUÇÃO IMPLEMENTADA

Recorremos à linguagem de programação C++ para implementar o programa. Ao contrário da linguagem C, esta permite-nos usar classes, o que nos vem oferecer uma melhor e mais simples gestão e tratamento de dados devido às suas novas estruturas de dados, que vêm tornar mais fácil a alocação e o acesso a estes, usando formas mais intuitivas e eficientes para quem programa.

Após se considerar a melhor opção, decidimos devidir as estruturas e utilizar 5 classes, a Contribuinte, a PassivoIVA que é derivada da Contribuinte, a Date, a DeclIVA e a Fatura.

Para a classe Contribuinte usamos:

- membros-dados (protected):
  - string NIF – para guardar o NIF do contribuinte.
  - string nome – para guardar o nome do contribuinte.
  - string email – para guardar o email do contribuinte.
  - string morada_fiscal – para guardar a morada do contribuinte.
- Membros-função (public):
  - Contribuinte() - construtor da classe.
  - getNIF() - usado para aceder ao NIF do contribuinte.
  - getNome() - usado para aceder ao nome do contribuinte.
  - getEmail() - usado para aceder ao email do contribuinte.
  - getMF() - usado para aceder a morada do contribuinte.
  - setNIF() - usado para alterar ao NIF do contribuinte.
  - setNome() - usado para alterar ao nome do contribuinte.
  - setEmail() - usado para alterar ao email do contribuinte.
  - Print() - imprime a informação do contribuinte.
  - IsValid() - verifica se um NIF é valido.

Para a classe PassivoIVA : public Contribuinte usamos:

- membros-dado:
  - int CAE – para guarda o codigo de ativdade economica.
  - vector<DeclIVA> declaracoes – para guardar as declarações do contribuinte passivo de IVA.
- membros-função:
  - PassivoIVA() - construtor da classe.
  - getCAE() - acede ao CAE.
  - getDeclaracoes() - acede ao vetor declaracoes.
  - setCAE() - altera o CAE.
  - setDeclaracoes() - altera o vetor declaracoes para um novo vetor.
  - print() - imprime a informação do contruibuinte passivo de IVA.

Para a classe Date usamos:

- membros-dado:
  - int ano – guardar ano
  - int mês – guardar mês
  - int dia – guardar dia
- membros-função:
  - Date() - Construtor(es) da classe
  - getData() - retorna a propria data
  - getAno() - retorna o ano
  - getMes() - retorn o mês
  - getDia() - retorna o dia
  - setData() - altera toda a data
  - setAno() - altera o ano
  - setMes() - altera o mês
  - setDia() - altera o dia
  - print() - imprime a data

Para a classe Declaraçao de IVA usamos:

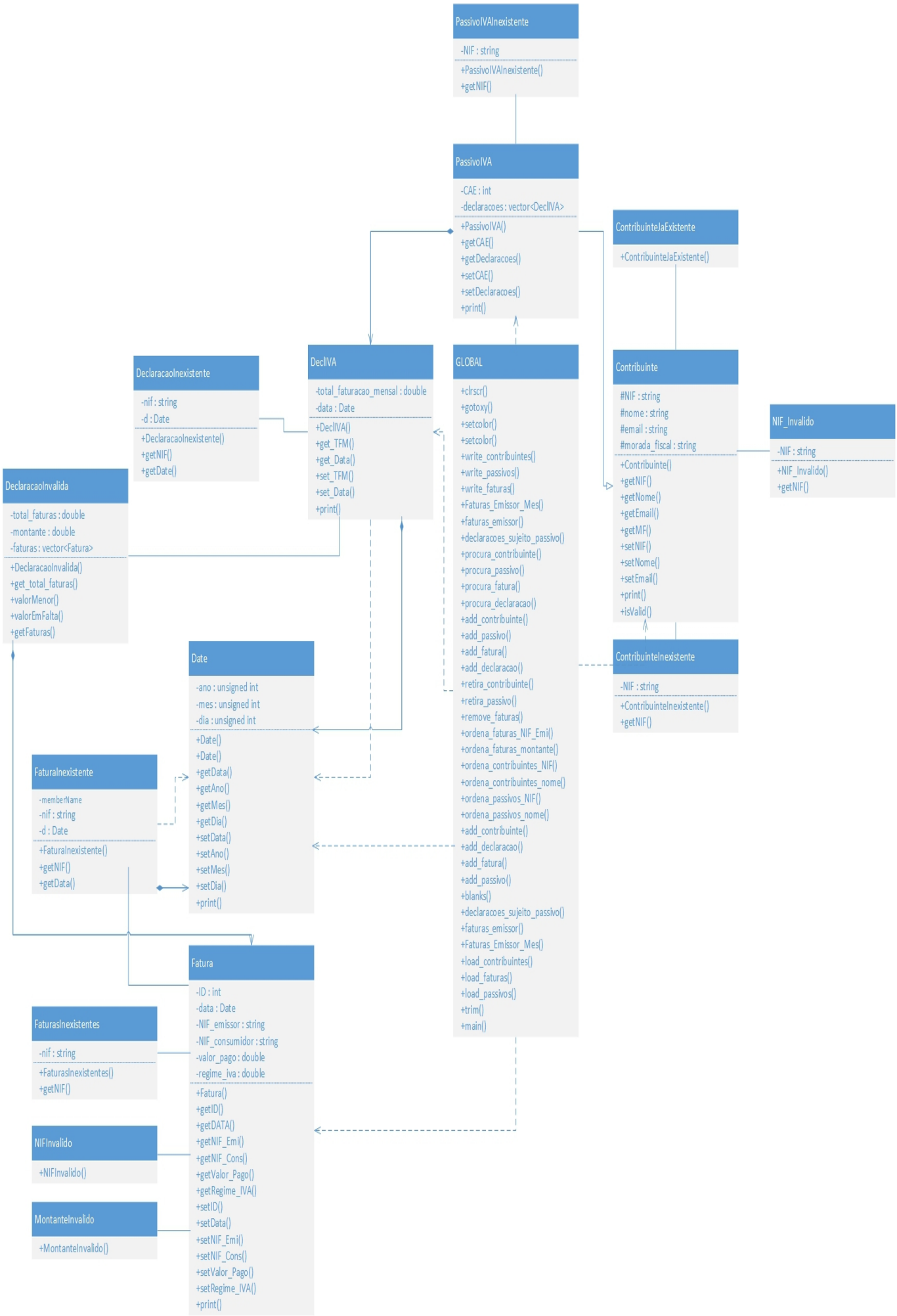
- membros-dados (protected):
  - double total_faturacao_mensal - para guardar o valor total da faturação mensal a declarar.
  - Date data – para guardar a data de emissão da declaração.
- Membros-função (public):
  - DeclIVA () - construtor da classe.
  - get_TFM () - usado para aceder ao total da faturação mensal declarado.
  - get_data () - usado para aceder á data de emissão da declaração.
  - set_TFM() - usado para alterar o valor total da faturação mensal.
  - set_data () - usado para alterar a data de emissao.
  - print () - imprime a informação da declaracão.
  - IsValid() - verifica se um NIF é valido.

Para a classe Fatura usamos:

- membros-dado:
  - int ID – guardar id
  - Date data – guardar data
  - string NIF_emissor – guardar nif do emissor
  - string NIF_consumidor – guardar nif do consumidor
  - double valor_pago – guardar o valor pago da fatura
  - double regime_iva – guardar regime IVA
- membros-função:
  - Fatura() - construtor
  - getID() - aceder ao id
  - getDATA() - aceder a data
  - getNIF_Emi() - aceder ao nif do emissor
  - getNIF_Cons- aceder ao nif do consumidor
  - getValor_Pago – aceder ao valor pago
  - getRegime_IVA – aceder ao regime IVA

Funções auxiliares:

- write_contribuintes() - guarda os contribuintes num ficheiro .txt
- write_passivos() - guarda os contribuintes passivos de IVA num ficheiro .txt
- write_faturas() - guarda as faturas num ficheiro .txt
- Faturas_Emissor_Mes() – retorna todas as faturas de um emissor especifico num determinado mes
- faturas_emissor() - retorna todas as faturas de um emissor especifico
- declaracoes_sujeito_passivo() – retorna todas as declarações de um sujeito passivo
- procura_contribuinte() – procura um determinado contribuinte no vetor Contribuintes
- procura_passivo() - procura um determinado passivo no vetor Passivos
- procura_fatura() - procura uma determinada fatura no vetor Faturas
- procura_declaracao() - procura uma determinada declaracao de um sujeito passivo especifico
- add_contribuinte() – adiciona um contribuinte ao vetor Contribuintes
- add_passivo()– adiciona um passivo aos vetores Contribuintes e Passivos
- add_fatura()– adiciona uma fatura a um passivo
- add_declaracao()– adiciona uma declaração a um passivo
- retira_contribuinte() – remove um contribuinte
- retira_passivo() – remove um sujeito passivo
- remove_faturas() – remove as faturas emitidas por um sujeito passivo
- ordena_faturas_NIF_Emi() – ordena as faturas de um sujeito passivo pelo NIF do emissor
- ordena_faturas_montante()– ordena as faturas de um sujeito passivo pelo montante
- ordena_contribuintes_NIF() – ordena contribuintes pelo NIF
- ordena_contribuintes_nome()– ordena contribuintes pelo nome
- ordena_passivos_NIF()– ordena passivos pelo NIF
- ordena_passivos_nome() – ordena passivos pelo nome
- blanks() – insere x espaços brancos a uma string – para escrita nos ficheiros
- declaracoes_sujeito_passivo() – retorna todas as declarações de um sujeito passivo especifico
- load_contribuintes() – faz load da informação guardada em contribuintes.txt
- load_faturas()– faz load da informação guardada em faturas.txt
- load_passivos()– faz load da informação guardada em passivos.txt
- trim() - remove x espaços brancos a uma string – para load dos ficheiros


PassivoIVAInexistente
-NIF : string
+PassivoIVAInexistente()
+getNIF()
PassivoIVA
-CAE : int
-declaracoes : vector<DeclIVA>
+PassivoIVA()
+getCAE()
+getDeclaracoes()
+setCAE()
+setDeclaracoes()
+print()
ContribuinteJaExistente
+ContribuinteJaExistente()
DeclaracaoInexistente
-nif : string
-d : Date
+DeclaracaoInexistente()
+getNIF()
+getDate()
DeclIVA
-total_faturacao_mensal : double
-data : Date
+DeclIVA()
+get_TFM()
+get_Data()
+set_TFM()
+set_Data()
+print()
GLOBAL
+clrscr()
+gotoxy()
+setcolor()
+setcolor()
+write_contribuintes()
+write_passivos()
+write_faturas()
+Faturas_Emissor_Mes()
+faturas_emissor()
+declaracoes_sujeito_passivo()
+procura_contribuinte()
+procura_passivo()
+procura_fatura()
+procura_declaracao()
+add_contribuinte()
+add_passivo()
+add_fatura()
+add_declaracao()
+retira_contribuinte()
+retira_passivo()
+remove_faturas()
+ordena_faturas_NIF_Emi()
+ordena_faturas_montante()
+ordena_contribuintes_NIF()
+ordena_contribuintes_nome()
+ordena_passivos_NIF()
+ordena_passivos_nome()
+add_contribuinte()
+add_declaracao()
+add_fatura()
+add_passivo()
+blanks()
+declaracoes_sujeito_passivo()
+faturas_emissor()
+Faturas_Emissor_Mes()
+load_contribuintes()
+load_faturas()
+load_passivos()
+trim()
+main()
Contribuinte
#NIF : string
#nome : string
#email : string
#morada_fiscal : string
+Contribuinte()
+getNIF()
+getNome()
+getEmail()
+getMF()
+setNIF()
+setNome()
+setEmail()
+print()
+isValid()
NIF_Invalido
-NIF : string
+NIF_Invalido()
+getNIF()
DeclaracaoInvalida
-total_faturas : double
-montante : double
-faturas : vector<Fatura>
+DeclaracaoInvalida()
+get_total_faturas()
+valorMenor()
+valorEmFalta()
+getFaturas()
ContribuinteInexistente
-NIF : string
+ContribuinteInexistente()
+getNIF()
Date
-ano : unsigned int
-mes : unsigned int
-dia : unsigned int
+Date()
+Date()
+getData()
+getAno()
+getMes()
+getDia()
+setData()
+setAno()
+setMes()
+setDia()
+print()
FaturaInexistente
-memberName
-nif : string
-d : Date
+FaturaInexistente()
+getNIF()
+getData()
Fatura
-ID : int
-data : Date
-NIF_emissor : string
-NIF_consumidor : string
-valor_pago : double
-regime_iva : double
+Fatura()
+getID()
+getDATA()
+getNIF_Emi()
+getNIF_Cons()
+getValor_Pago()
+getRegime_IVA()
+setID()
+setData()
+setNIF_Emi()
+setNIF_Cons()
+setValor_Pago()
+setRegime_IVA()
+print()
FaturasInexistentes
-nif : string
+FaturasInexistentes()
+getNIF()
NIFInvalido
+NIFInvalido()
MontanteInvalido
+MontanteInvalido()

Lista de casos de utilização

Quanto à interface desenvolvida para este programa, foi criado um menu inicial onde é apresentado o titulo do projeto e uma lista de comandos que permitem ao utilizador escolher que ação quer tomar: se quer adicionar algo, remover algo, imprimir a informação já adicionada por si, ordená-la segundo um determinado critério, procurar um certo contribuinte ou outro tipo de informação, salvar a informação introduzida ou encerrar o programa. Estes comandos estão enumerados e é pedido ao utilizador que introduza um destes na consola para seguir para o menu seguinte referente apenas á ação escolhida.

Depois do utilizador escolher, por exemplo, o comando nº 1, vai ser encaminhado para o menu seguinte no qual tem outra série de comandos possíveis enumerados, inclusive a opção de retornar ao menu anterior, caso queira voltar atrás. O comando é escolhido através da sua escrita na consola seguido da tecla ENTER.

Se o utilizador escolher o comando nº 3, vai ser encaminhado para o menu que tem os comandos que lhe permitem imprimir todos os consumidores, sujeitos passivos de iva, faturas, faturas de um sujeito passivo de iva específico, faturas de um sujeito passivo de iva especifico num determinado mês e imprimir declarações de um sujeito passivo de iva específico. Desta forma pode controlar melhor toda a informação que adicionou ao programa, de uma maneira organizada e onde pode aceder a todos os pormenores desta.

Outro exemplo, mais particular, é o do comando número 6, que lhe permite “salvar informação”, ou seja, toda a informação já registada pelo utilizador vai ser guardada recorrendo a ficheiros de texto, que vão ser automaticamente carregados ao iniciar o programa na próxima vez. Assim nenhuma informação é perdida entre utilizações. Os ficheiros de texto correspondentes são contribuintes.txt, que guarda toda a informação relativa aos contribuintes adicionados, passivos.txt, que guarda a informação relativa aos sujeitos passivos de iva, incluindo todas as faturas e todas as declarações por eles emitidas, e faturas.txt que guarda todas as faturas emitidas durante o correr do programa.

Devemos ainda referir que em todos os casos temos em conta todos os estados de erro possíveis de acontecer, através de exceções que são lançadas no decorrer do programa. São mostradas mensagens de erro na consola caso estes aconteçam. Também são mostradas mensagens de sucesso caso a adição ou remoção de algum tipo de informação for bem-sucedido.

Difculdades encontradas

O trabalho decorreu sem grandes dificuldades, acreditamos que as tarefas foram bem distribuídas e cada um desempenhou as tarefas com que se sentia mais à vontade.
A maior dificuldade encontrada, que foi no entanto superada, foi a tarefa de criar as funções que interagem com ficheiros visto que nenhum de nós tinha memória de como as implementar. Contudo apesar da dificuldade inicial, com algum estudo fomos capazes de as implementar com relativa facilidade.

Outra dificuldade foi a distância a que estávamos durante a semana da FEUP, contudo através das comunicações online esta dificuldade também foi facilmente superada.
Por último pode ser referido que uma dificuldade extra que sentimos por causa do tema, foi o facto de termos de pesquisar como são alguns aspetos das faturas e declarações de IVA.

A parte do código foi realizada pelo Eduardo Leite e pela Andreia Rodrigues, sendo que toda a interface, os sorts, as pesquisas e os adicionar foram realizadas pela Andreia. O Diogo implementou a função de validar, e a funçoes de load e write das classes Fatura e Declaracao. O restante codigo foi realizado pelo Eduardo Leite. A correcção foi efetuada posteriormente por todos.

O doxygen e o UML foram realizados pelo Diogo.

E o relatório foi realizado em conjunto.