FRANCESCO PRESOTTO

Università degli Studi di Padova

LAUREA TRIENNALE IN INGEGNERIA INFORMATICA

Sviluppo di un'applicazione in VBA per la creazione di offerte commerciali complesse

RELATORE: PROF. SERGIO CONGIU

RELAZIONE DEL TIROCINIO

Giugno 2013

Francesco Presotto: *Sviluppo di un' applicazione in VBA per la creazione di offerte commerciali complesse.,*
Relazione del tirocinio, © Giugno 2013

E-MAIL: francesco.presotto@studenti.unipd.it

Questo documento presenta la relazione del tirocinio da me svolto presso Promoservice SRL, con inizio in data 18/7/2011 e termine in data /12/2011, e durata prevista di 500 ore.

Sommario

1 | INTRODUZIONE

In questo capitolo verrà descritta l'azienda e il contesto in cui verrà utilizzata l'applicazione in questione ed il suo obiettivo.

1.1 OBIETTIVI

L'obiettivo dell'esperienza è stato la realizzazione di una macro per la creazione di offerte commerciali complesse, che permettesse tramite l'uso di un' interfaccia grafica di selezionare le voci di interesse così da personalizzare l'offerta in funzione del cliente.
A carico dell'applicazione vi è anche la gestione del riepilogo economico strutturato in tre tipologie: configurazione ottima, media e minima, così da poter differenziare all'interno della stessa offerta tre soluzioni differenti in base alle caratteristiche che si vogliono includere o meno.

1.2 CAMPO APPLICATIVO

L'applicazione è nata per soddisfare l'esigenza dei commerciali dell'azienda, dando loro un sistema che raccogliesse tutti i servizi proposti e che permettesse di creare rapidamente le offerte commerciali tramite un'interfaccia grafica intuitiva, evitando di doverle comporre manualmente, con un notevole risparmio di tempo.

1.3 PROMOSERVICE SRL

L'Azienda *Promoservice s.r.l.* è una società informatica nata nel 1998 con sede a Chiarano, in Provincia di Treviso.
Si occupa di:

- *sviluppo di siti internet*, occupandosi anche di visibilità sui motori di ricerca e di brand reputation curando l'immagine del committente sui social network;

- *sviluppo di piattaforme di e-commerce personalizzate*, per permettere la vendita dei propri prodotti o servizi;

- *sviluppo software* per la gestione ospedaliera, per la gestione di e-commerce alberghiero, per la gestione dei fornitori.

2 | ANALISI DEL PROGETTO

Verrà qui effettuata un'analisi dei requisiti del sistema da realizzare, che offrirà in particolare una panoramica delle funzionalità che l'applicazione dovrà essere in grado di offrire.

2.1 APPROCCIO AL PROGETTO

Il linguaggio di programmazione con cui è stata implementata l'applicazione è Visual Basic for Application (VBA) su specifica richiesta di Promoservice srl in quanto già integrato nella suite Microsoft Office.
Aspetto fondamentale tenuto in considerazione in fase di sviluppo è stata l'implementazione con approccio modulare per esigenze interne dell'azienda, in quanto si è voluto rendere l'applicazione facilmente modificabile anche da chi non ha propriamente competenze nella programmazione, sia per la rimozione di voci obsolete dall'offerta commerciale, che per l'aggiunta di nuove.
A tal proposito al termine dello sviluppo è stata redatta una guida che spiega passo-passo le operazioni da eseguire per la rimozione/inserimento.

2.2 PANORAMICA

Lo sviluppo dell'applicazione è stato suddiviso in due blocchi principali, il modulo per la creazione delle offerte, e il modulo per la gestione del riepilogo economico basato su un'offerta già creata.

2.2.1 Gestione Offerte

Lo sviluppo dell'applicazione è stato preceduto dall'analisi della composizione delle offerte, strutturandone dapprima una versione completa in un unico file di testo con tutte le funzionalità proposte da Promoservice srl, e successivamente suddividendo i capitoli in singoli documenti utilizzando un template di base, così da rendere uniforme la formattazione di ciascun file attraverso stili dedicati e l'impiego di segnalibri per marcare le informazioni da estrapolare successivamente.

2.2.2 Gestione riepilogo economico

Per la tariffazione dei servizi sono state isolate in fase di analisi due tipologie distinte:

- *costo del servizio:* riferito allo sviluppo del servizio/funzionalità;
- *canone annuo:* riferito al mantenimento del servizio/funzionalità.

All'interno del template utilizzato per la redazione dei singoli capitoli sono stati inseriti due segnalibri in corrispondenza dello spazio dedicato a queste due voci, così da renderle accessibili durante l'elaborazione del documento finale.

3 | AMBIENTE DI PROGRAMMAZIONE

In questo capitolo verrà presentato il linguaggio e l'ambiente di programmazione utilizzato per la realizzazione del progetto.

3.1 VBA

Visual Basic for Applications (VBA), come suggerito dal nome stesso, è un'implementazione di Visual Basic sviluppata da Microsoft e nativa di alcune sue applicazioni quali la suite Microsoft Office.
Proprio la sua integrazione all'interno del pacchetto Office ha reso questo linguaggio interessante per lo sviluppo del progetto, in quanto permette di controllare ogni aspetto dell'applicazione espandendo le funzionalità dei precedenti linguaggi utilizzati per la registrazione di macro, e consente l'interoperabilità tra i vari programmi della suite.
Va però fatto notare che VBA necessita sempre di un'applicazione che faccia da host (come Excel o Word) per poter funzionare, eccezione fatta per programmi ben più complessi creati con Visual Studio, pertanto per poter eseguire delle macro sviluppate in VBA per Microsoft Word è necessario che tale programma sia attivo.
Subroutine e funzioni sono i principali oggetti di questo linguaggio.
La subroutine, chiamata anche procedura o macro, esegue automaticamente un insieme di operazioni, nel documento, nella cartella, foglio e/o cella selezionate al momento in cui viene richiamata,
La funzione è un altro tipo di procedura e si differenzia dalla subroutine in quanto ritorna sempre un valore e quasi sempre richiede uno o più parametri; anche le subroutine possono ricevere parametri, ma non restituiscono mai valori.
Analizziamo ora i tipi di progetto realizzabili in VBA:

- Progetto di documento
- Progetto di modello
- Progetto di componente aggiuntivo

Nei "Progetti di documento" tutte le componenti del progetto sono legate al documento di lavoro che si utilizza al momento, senza ripercussioni

sull'applicazione o sui modelli generali, quindi l'utilizzo delle subroutine rimane circoscritto al file in cui vengono salvate ed è la tipologia che è stata utilizzata per sviluppare quanto richiesto.
Nei "Progetti di modello" invece il codice generato è associato al modello generale dell'applicazione che si sta utilizzando, e sarà quindi disponibile anche in tutti i nuovi documenti che andremo a creare.
I componenti aggiuntivi sono strumenti che permettono di ampliare le funzionalità dell'applicazione in uso, come ad esempio un comando personalizzato richiamabile dalla barra degli strumenti o da una voce di menù ed anche questa tipologia di progetto si lega al modello generale dell'applicazione.

3.2 AMBIENTE DI SVILUPPO VBA

L'ambiente di sviluppo per VBA è facilmente accessibile sia in Microsoft Office 2007 che 2010; una volta avviata l'esecuzione del programma per cui si vuole sviluppare una macro è sufficiente utilizzare la combinazione di tasti `Alt+F11` per far apparire la finestra seguente:

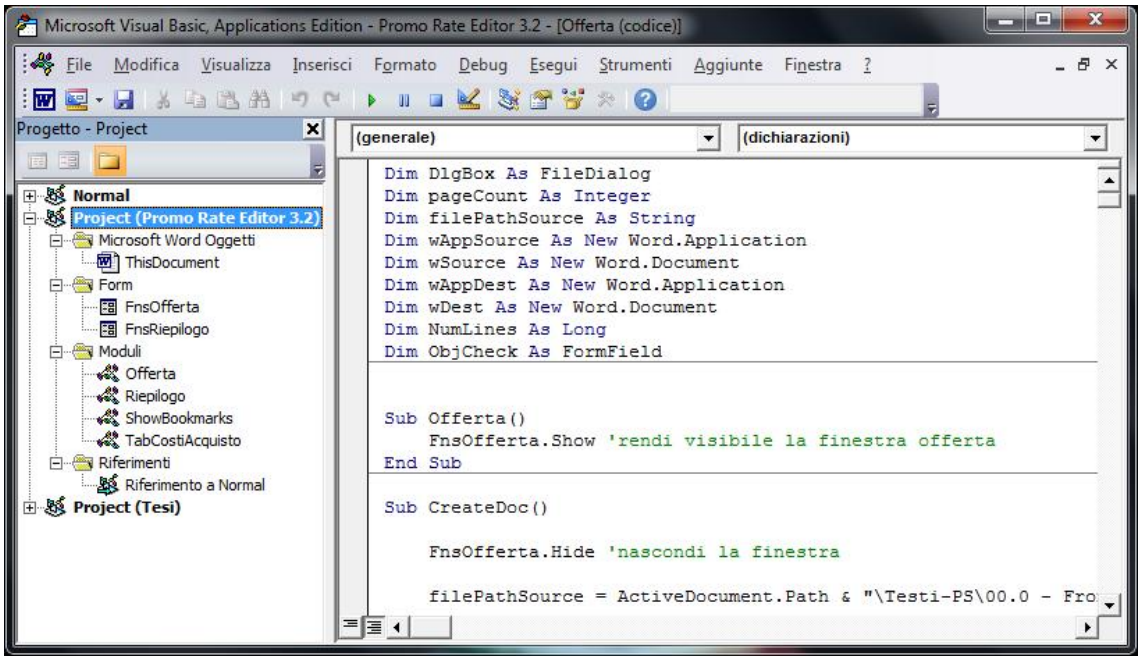

Figura 1: Ambiente di programmazione

Dalla struttura ad albero contenuta nel "Project Explorer" in sinistra è possibile decidere se sviluppare macro riutilizzabili in tutti i nuovi documenti, intervenendo all'interno della cartella "Normal", o macro utilizzabili solo dal documento corrente, come in questo caso.

All’interno della cartella “Form” vengono salvate tutte le finestre di dialogo con relativo code-behind, mentre all’interno della cartella “Moduli” vi sono le subroutine e le funzioni implementate, suddivise in file per accorpare tra di loro quelle riguardanti la stessa finestra di dialogo.
Sulla destra invece la classica “Code Window”, l’area di lavoro.

4 | REALIZZAZIONE

In questo capitolo verrà descritto l'effettivo sviluppo dell'applicazione, sulla base dell'analisi affrontata nel capitolo 2.

La prima attività è stata la gestione e l'archiviazione dei singoli file di testo prodotti, successivamente si è proceduto con l'effettivo sviluppo dell'applicazione, dapprima con la gestione delle offerte, e poi con il riepilogo economico dell'offerta generata.

4.1 GESTIONE DEI SINGOLI CAPITOLI

Per l'archiviazione dei file di testo si è scelto di utilizzare una gestione tramite cartelle e sottocartelle, in modo da rendere i singoli file facilmente raggiungibili e modificabili in base alle esigenze; la cartella di archiviazione dei file di testo va mantenuta allo stesso livello gerarchico del file contenente la macro, in quanto i riferimenti e i percorsi sono relativi alla posizione di quest'ultimo.

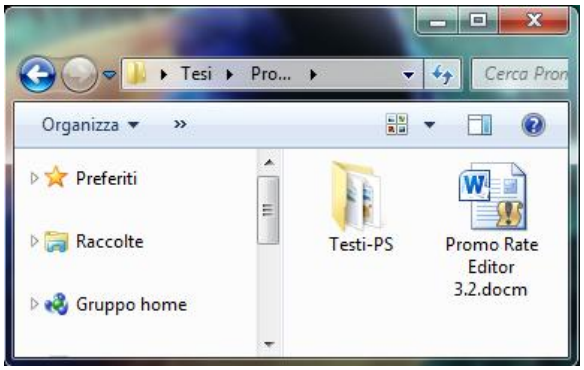

Figura 2: Cartella progetto

Le varie sottocartelle invece corrispondono alle sezioni di cui si compone l'offerta, nominate seguendo la regola: ##.# - [NOME DELLA SEZIONE].
L'uso dei numeri decimali serve per poter inserire una nuova sezione tra due già esistenti senza per questo dover rinominare tutte le successive per mantenere l'ordine alfabetico.

Figura 3: Struttura ad albero delle cartelle con i documenti originali

All'interno di ciascuna cartella è stato replicato lo stesso schema per la nomenclatura dei file, in modo da poterne inserire di nuovi senza dover andare a modificare quelli già presenti.
Caso a parte sono la prima cartella, *0.00 - Frontespizio*, e l'ultima, *ZZ – Condizioni Generali* contenenti rispettivamente la carta intestata dell'azienda e le condizioni generali di fornitura ed erogazione dei servizi offerti, mentre per quanto riguarda le altre, il primo documento presente in ciascuna di esse contiene solamente il titolo della sezione, formattato con il proprio stile, per renderlo indipendente da ciascun capitolo.

4.2 FORM OFFERTA

4.2.1 Interfaccia utente

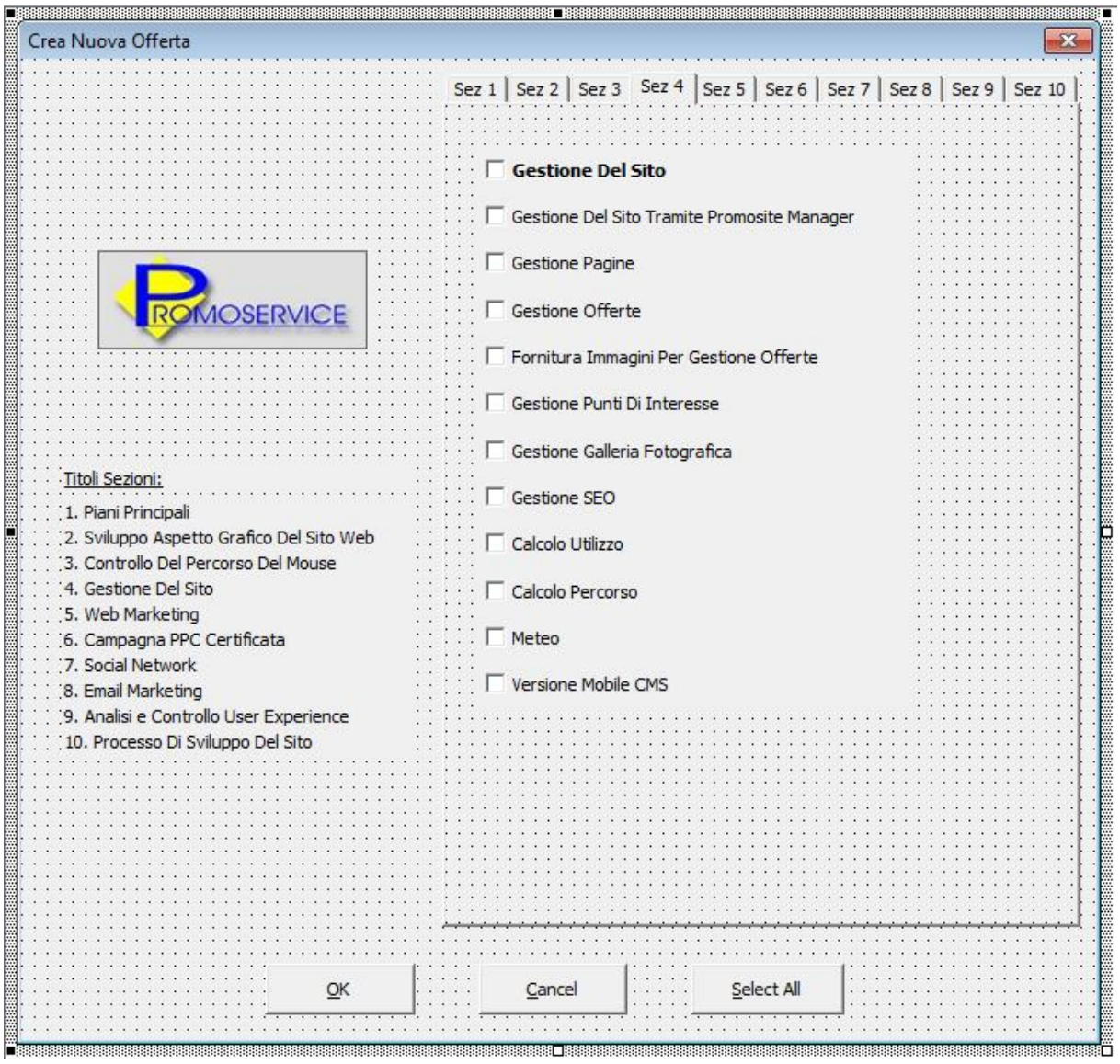

Figura 4: Interfaccia utente Form Offerta

Per l'implementazione dell'interfaccia grafica è stato utilizzato il tool di sviluppo integrato, che dispone di una casella strumenti essenziale, ma sufficiente per le finalità del progetto, il form infatti si compone perlopiù di tab e di checkbox; a ogni tab corrisponde una sezione, di cui è sempre visibile un indice di riepilogo per potersi orientare, mentre a ogni checkbox è associato un capitolo; se la checkbox è selezionata il contenuto del rispettivo file sarà inserito all'interno del documento finale.

Dei tre pulsanti presenti, il primo, “OK”, avvia la creazione del documento sulla base delle checkbox selezionate, il secondo, “Cancel” chiude la finestra di dialogo e il terzo, “Select All” permette di selezionare tutte le checkbox nel caso in cui l’offerta da creare sia particolarmente complessa e sia più rapido deselezionare le checkbox indesiderate che selezionare quelle volute.

4.2.2 Code Behind – Form Offerta

Il codice integrato nel form “Offerta” serve solo ad associare ai pulsanti presenti le relative funzioni, implementate all’interno del modulo offerta:

```
Sub cmdOK_Click()
    Call Offerta.CreateDoc
End Sub

Sub cmdCancel_Click()
   End
End Sub

Sub cmdAll_Click()
    Call Offerta.SelectAll
End Sub
```

Il click sul tasto “OK” richiama la subroutine `CreateDoc`, il click sul tasto “Cancel” semplicemente chiude la finestra di dialogo mentre il click sul tasto “Select All” richiama l’omonima subroutine `SelectAll`.

4.3 MODULO OFFERTA

Il modulo “Offerta” contiene tutte le subroutine inerenti la creazione del documento, per prima cosa vengono dichiarate le variabili che verranno utilizzate all’interno della subroutine:

```
Dim DlgBox As FileDialog
Dim pageCount As Integer
Dim filePathSource As String
Dim wAppSource As New Word.Application
Dim wSource As New Word.Document
Dim wAppDest As New Word.Application
Dim wDest As New Word.Document
Dim NumLines As Long
```

`DlgBox`: variabile utilizzata per instanziare la finestra di dialogo che si apre alla fine della creazione del documento e ne permette il salvataggio;

`pageCount`: variabile numerica utilizzata per memorizzare il numero totale delle pagine di cui è composto il documento creato;

`filePathSource`: variabile contenente di volta in volta il percorso del file da utilizzare come sorgente per essere copiato e incollato nel documento in costruzione;

`wAppSource`: variabile utilizzata come istanza dell’applicazione Microsoft Word destinata ad ospitare di volta in volta il documento sorgente, avvia l’applicazione senza documenti aperti;

`wSource`: variabile utilizzata come istanza per ospitare di volta in volta il documento sorgente;

`wAppDest`: variabile utilizzata come istanza dell’applicazione Microsoft Word destinata ad ospitare il documento in fase di creazione, avvia l’applicazione senza documenti aperti;

`wDest`: variabile utilizzata come istanza per ospitare il documento di destinazione, di cui al termine verrà cheisto il salvataggio.

Seguono poi le subroutine implementate:

```
Sub Offerta()

Sub CreateDoc()

Sub InsertIntoDoc(ByRef fileSource As String)

Sub SelectAll()
```

4.3.1 Subroutine Offerta( )

La subroutine `Offerta()` è associata al click del pulsante “Crea Offerta” aggiunto nella barra di accesso rapido:

Figura 5: Icona di accesso rapido al Form Offerta

La sua funzione è quella di rendere visibile la finestra di dialogo:

```
Sub Offerta()
    FnsOfferta.Show 'rendi visibile la finestra offerta
End Sub
```

4.3.2 Subroutine CreateDoc( )

La subroutine `CreateDoc()` viene invece, richiamata dal tasto “Ok” e come prima azione nasconde la finestra chiamante per poi costruire il percorso del primo file da aprire.

```
Sub CreateDoc()

    FnsOfferta.Hide 'nascondi la finestra

    filePathSource = ActiveDocument.Path &
    'percorso file origine
    "\Testi-PS\00.0 - Frontespizio\FRONTESPIZIO-PS.docx"
```

La costruzione dei percorsi si basa sulla posizione del documento attivo (`ActiveDocument.Path`), a cui poi viene aggiunto staticamente l'ultima parte dell'indirizzo, motivo per cui è necessario mantenere il file .docm e la cartella contenente i testi allo stesso livello gerarchico all'interno della medesima cartella per non invalidare il puntatore.

Nel passaggio successivo viene nascosta l'istanza di applicazione che ospiterà il documento di destinazione, così da impedire all'utente l'intervento accidentale in fase di elaborazione e viene inizializzata la variabile `wDest` associandola al documento FRONTESPIZIO-PS.docx, utilizzato come template di partenza in cui poi vengono inseriti i vari capitoli, che pur avendo la proprietà `ReadOnly` impostata a false, non viene mai modificato in quanto al termine della creazione dell'offerta viene forzato il salvataggio con nome.

```
'apri il file FRONTESPIZIO a cui poi verranno accodati gli altri file
wAppDest.Visible = False
Set wDest = wAppDest.Documents.Open(
    fileName:=filePathSource, ReadOnly:=False, Visible:=True
)
```

Una volta aperto il documento non rimane che posizionare il cursore per permettere il corretto inserimento dei successivi capitoli.

```
'sposta il cursore in coda al documento
wAppDest.Selection.EndKey Unit:=wdStory
'inserisci un'interruzione di pagina
wAppDest.Selection.InsertBreak wdPageBreak
```

L'inserimento di sezioni e capitoli è gestito tramite un blocco di codice modulare che si ripete uguale per ogni sezione, verificando il valore di ogni singola checkbox.
Le motivazioni di tale scelta sono state trattate nel capitolo 2, pertanto procediamo con l'analizzare nel dettaglio un singolo blocco di esempio:

```
'Sezione 1
'Piani Principali
    'aggiorna la barra di stato
    Application.StatusBar = "Inserimento sezione: 'Piani principali'"
    'imposta la cartella coi file di origine
    filePathSource = ActiveDocument.Path &
        "\Testi-PS\01.0 - Piani Principali\"
```

```
'se la checkbox è selezionata inserisci il file corrispondente
If FnsOfferta.CheckBox1 Then
    Call InsertIntoDoc(filePathSource & "00.0_PIANI_PRINCIPALI.docx")
End If
```

Per prima cosa viene aggiornata la barra di stato del file .docm indicando la sezione che si sta analizzando, così da fornire un riscontro visivo dell'attività in corso all'utente, successivamente all'interno della variabile `filePathSource` viene salvato il percorso relativo alla cartella della sezione in esame con la modalità già descritta; si esegue quindi un controllo sullo stato della checkbox, se è stata selezionata viene chiamata la subroutine `InsertIntoDoc([`*percorso*`])`, analizzata in seguito, che si occupa dell'inserimento del file all'interno del documento finale.
Una volta processato anche il contenuto dell'ultima sezione, presente in ogni documento, l'istanza di applicazione che ospita il documento di destinazione viene resa visibile così da poter far comparire la finestra di dialogo per il salvataggio del file appena creato.

```
'Sezione Finale (condizioni generali di fornitura)
'Note Generali
    'aggiorna la barra di stato
    Application.StatusBar = "Inserimento sezione: 'Note Generali'"
    filePathSource = ActiveDocument.Path & "\Testi-PS\ZZ.Condizioni Generali\"

        'questa sezione c'è in ogni offerta, pertanto non servono checkbox
        Call InsertIntoDoc(filePathSource & "CONDIZIONI_GENERALI.docx")

        'elimina l'ultima pagina aggiunta da InsertIntoDoc
        wAppDest.Selection.TypeBackspace
        wAppDest.Selection.TypeBackspace
        wAppDest.Selection.TypeBackspace

    'apri la finestra salva con nome
    Application.StatusBar = "Salva Con Nome" 'aggiorna la barra di stato
    wAppDest.Visible = True
    Set DlgBox = wAppDest.FileDialog(msoFileDialogSaveAs)
    With DlgBox
        .title = "Salva File Con Nome"
        .AllowMultiSelect = False
        .ButtonName = "Salva"
        .InitialView = msoFileDialogViewDetails
        If .Show = -1 Then
            .Execute
        Else
            Exit Sub
        End If
    End With
```

In seguito al click sul tasto “Salva”, prima di procedere con il vero e proprio salvataggio, tutto il testo formattato con lo stile “Capitolo” viene selezionato e riformattato così da assicurare la corretta numerazione progressiva dei capitoli inseriti (elenco puntato), a seguire si aggiornano i campi dinamici del footer: nome del documento e numero di pagine.

```
'aggiorna i numeri dei capitoli
wAppDest.Selection.Find.ClearFormatting
wAppDest.Selection.Find.Style = wDest.Styles("Capitolo")
wAppDest.Selection.Find.Replacement.ClearFormatting
wAppDest.Selection.Find.Replacement.Style = wDest.Styles("Capitolo")
With wAppDest.Selection.Find
    .Text = ""
    .Replacement.Text = ""
    .Forward = True
    .Wrap = wdFindContinue
    .Format = True
    .MatchCase = False
    .MatchWholeWord = True
    .MatchWildcards = False
    .MatchSoundsLike = False
    .MatchAllWordForms = False
End With
wAppDest.Selection.Find.Execute Replace:=wdReplaceAll

'aggiorna il footer
pageCount = wDest.BuiltInDocumentProperties(14)
If pageCount >= 1 Then
    wDest.Sections(ActiveDocument.Sections.Count)_
        .Footers(1).Range.Fields.Update
End If
Application.ScreenUpdating = True
```

Al termine di queste istruzioni si prosegue con la procedura di salvataggio e chiusura, rilasciando anche lo spazio utilizzato dalle variabili non più necessarie;

```
'salva e chiudi il file di destinazione
wDest.Save
wDest.Close
wAppDest.Quit
Set wDest = Nothing
Set wAppDest = Nothing

'chiudi il file di origine
wAppSource.Quit
Set wSource = Nothing
Set wAppSource = Nothing
```

```
    Unload FnsOfferta   'chiudi la finestra
    Application.StatusBar = ""  'aggiorna la barra di stato
    Beep    'segnale acustico
    MsgBox "Fatto!", vbExclamation

End Sub
```

Il completamento di tutte le operazioni viene notificato da un ultima finestra di dialogo.

4.3.3 Subroutine InsertIntoDoc(…)

La subroutine `InsertIntoDoc()` dopo aver aperto il file sorgente, il cui percorso viene passato per parametro, ne copia l'intero contenuto per poi inserirlo all'interno del file di destinazione; l'istanza che ospita il file di origine (`wSource`) viene inizializzata e chiusa ogni volta che la subroutine viene chiamata, mentre l'istanza di applicazione che ospita di volta in volta il documento sorgente (`wAppSpource`) viene chiusa solo alla conclusione della subroutine `CreateDoc()`, così da caricare in memoria una sola volta il processo winword.exe dedicato ai file sorgente.
In sostanza questo corrisponde a caricare una sola volta il programma Microsoft Word e limitarsi ad aprire e chiudere le schede associate ai documenti, scelta che in termini di efficienza comporta una notevole riduzione nei tempi di esecuzione.

```
'inserisci nel documento di destinazione il contenuto del file
'ricevuto come argomento (percorso file)
Sub InsertIntoDoc(ByRef fileSource As String)

    'setup puntatore
    wAppSource.Visible = False
    Set wSource = wAppSource.Documents.Open(
        fileName:=fileSource, ReadOnly:=False, Visible:=True
    )

    'seleziona ,copia e incolla
    wAppSource.Selection.WholeStory
    wAppSource.Selection.Copy
    wAppDest.Selection.PasteAndFormat (wdUseDestinationStylesRecovery)

    'se il file di origine ha più di 12 righe inserisci interruzione di pagina
    NumLines = wSource.ComputeStatistics(wdStatisticLines)
    If NumLines > 12 Then
        wAppDest.Selection.InsertBreak wdPageBreak
    End If

    'chiudi il file di origine
    wSource.Save
    wSource.Close

End Sub
```

4.4 FORM RIEPILOGO ECONOMICO

Per poter utilizzare la macro che gestisce il riepilogo economico è necessario aver prima creato un documento contenente un'offerta commerciale, un volta richiamata infatti la macro richiederà di fornire un file valido in caso contrario non verrà neppure visualizzata l'interfaccia utente.

4.4.1 Interfaccia Utente

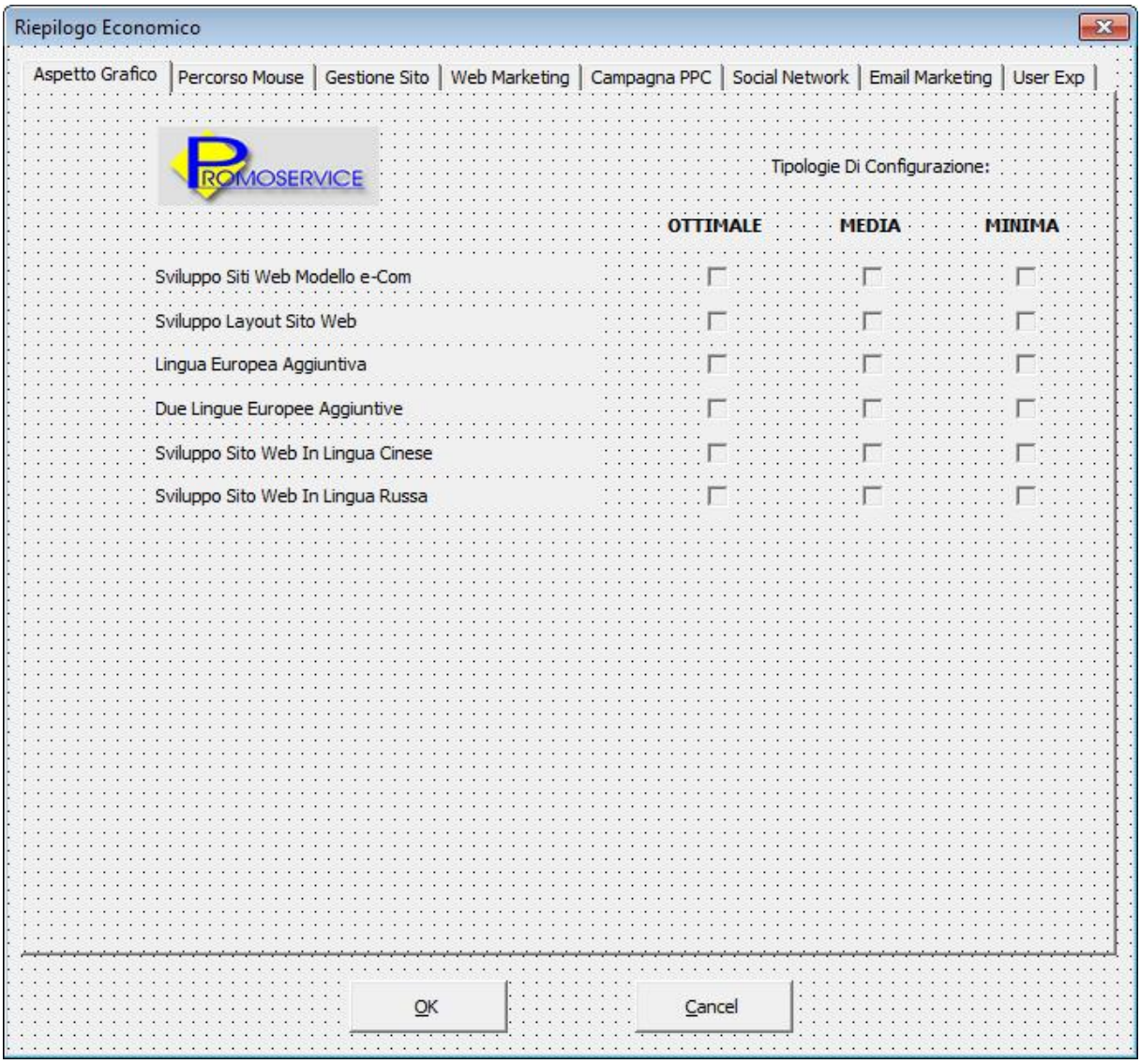

Figura 6: Interfaccia utente Form Riepilogo Economico

Da richieste iniziali, per ciascuna offerta dev'essere possibile strutturare tre diversi tipi di configurazione: ottima, media e minima, così da delineare tre fasce di prezzo in base alle funzionalità o servizi aggiunti e per fare questo nel form "Riepilogo

Economico", implementato sempre utilizzando il tool di sviluppo integrato, ad ogni capitolo sono associate tre checkbox suddivise in tre colonne che corrispondono ai tre diversi tipi di configurazioni possibili, per quanto riguarda le sezioni invece, come nel form "Offerta" ad ognuna è associato un tab.

4.4.2 Code Behind – Form Riepilogo Economico

Anche in questo caso l'implementazione del codice vero e proprio è contenuta nel relativo modulo, mentre il codice integrato associato al form "Riepilogo Economico" serve solo ad associare le funzioni ai pulsanti presenti:

```
Sub cmdCANC_Click()
    End
End Sub

Sub cmdOK_Click()
    Call Riepilogo.InsRiepilogo
End Sub
```

Cliccando sul tasto "OK" viene richiamata la subroutine `InsRiepilogo`, mentre cliccando su "Canc" viene chiusa la finestra.

4.5 MODULO RIEPILOGO ECONOMICO

Il modulo “Riepilogo Economico” contiene tutte le subroutine necessarie per gestire le tre tipologie di configurazione a partire dalle variabili che verranno utilizzate all’interno della subroutine, prima dell’implementazione però l’indice di base per gli array viene impostato a 1 e non a 0:

```
' l'indice degli array parte da 1 e non da 0
Option Base 1
```

Scelta fatta, così come richiesto in fase di progettazione, nell’ottica di eventuali modifiche dell’applicazione, anche da parte di chi non ha propriamente competenze nella programmazione, e che quindi non ha familiarità con gli array zero-based.

Vengono poi dichiarate tutte le variabili utilizzate all’interno del modulo:

```
Dim percIVA As Integer
Dim DlgBox As FileDialog
Dim strTitoli(100) As String
Dim titoloCap As String
Dim nomeBm As String
Dim numCap As Integer
Dim iTab As Integer
Dim iRow As Integer
Dim numRows As Integer
Dim xlApp As New Excel.Application
Dim xlWB As New Excel.Workbook
```

- `percIVA`: costante numerica impostata al valore percentuale dell’IVA attuale;

- `DlgBox`: variabile utilizzata per instanziare la finestra di dialogo che si apre una volta cliccato “OK” sul form “Riepilogo Economico” e che permette di selezionare il file da aprire;

- `strTitoli`: Array di stringhe in cui vengono salvati i codici prodotto dei servizi o delle funzionalità presenti nell’offerta;

- `titoloCap`: variabile in cui viene salvato di volta in volta il titolo del capitolo in esame;

- `nomeBm`: variabile in cui viene salvato di volta in volta il codice prodotto del capitolo in esame;

- `numCap`: variabile in cui viene salvato di volta in volta il numero del capitolo in esame;

- `iTab`: variabile numerica utilizzata per tenere traccia del numero di tabelle presenti nel documento così da poter selezionare e modificare quella di interesse;

- `iRow`: variabile numerica utilizzata come puntatore per la riga, riferita alla tabella, da modificare;

- `numRows`: variabile numerica utilizzata per tenere traccia del totale di righe presenti nella tabella di volta in volta selezionata;

- `xlApp`: variabile utilizzata come istanza dell'applicazione Microsoft Excel destinata ad ospitare il foglio di calcolo utilizzato per il conteggio dei totali, avvia l'applicazione senza documenti aperti;

- `xlWB`: : variabile utilizzata come istanza per ospitare di volta in volta il foglio di calcolo in cui vengono eseguite le operazioni aritmetiche;

Segue quindi l'implementazione delle subroutine qui elencate:

```
Sub Riepilogo()
Sub InsRiepilogo()
Sub InsConfigOttimale()
Sub InsConfigMedia()
Sub InsConfigMinima()
Sub CreaTabRiepilogo()
Sub TrovaNumCap(ByVal bookmark As String, ByRef numCap As Integer)
Sub InserisciCostoServizio(ByVal bookmark As String, ByVal numCap As Integer,
                           ByVal title As String, ByVal bold As Boolean)
Sub InserisciCanoneAnnuo(ByVal bookmark As String, ByVal bold As Boolean)
Sub SommaTOT(ByVal numCol As Integer)
Sub InserisciSottoscrizione()
Sub EnableCheckBox()
```

4.5.1 Subroutine Riepilogo( )

La subroutine `Riepilogo()` è associata al click del pulsante "Riepilogo Economico" aggiunto nella barra di accesso rapido:

Figura 7: Icona di accesso rapido al Form per il Riepilogo Economico

Inizialmente rende visibile la finestra di dialogo che permette di selezionare il file da aprire, una volta selezionato richiama la subroutine `EnableCheckBox` e successivamente rende visibile la finestra di dialogo per la gestione delle tre configurazioni.

```
Sub Riepilogo()

    'apri la finestra Apri File
    Set DlgBox = Application.FileDialog(msoFileDialogOpen)
    With DlgBox
        .title = "Apri File"
        .AllowMultiSelect = False
        .ButtonName = "Apri"
        .Filters.Add "File Word", "*.docx", 1
        .InitialView = msoFileDialogViewDetails
        If .Show = -1 Then
            .Execute
        Else
            Exit Sub
        End If
    End With

    Call EnableCheckBox 'chiama la sub EnableCheckBox
    FnsRiepilogo.Show

End Sub
```

4.5.2 Subroutine InsRiepilogo( )

La subroutine `InsRiepilogo()` viene avviata al click sul tasto "OK" del form "Riepilogo Economico", nasconde la finestra di dialogo e richiama in sequenza tre

subroutine per inserire nel documento, gestendone la formattazione tramite il posizionamento del cursore, le configurazioni Ottima, Media e Minima.
Per ciascuna viene inserito il testo per la sottoscrizione del servizio da parte del cliente e al termine delle opearazioni la conclusione viene notificata da un segnale acustico e da una piccola MessageBox.

```
Sub InsRiepilogo()

    FnsRiepilogo.Hide 'nascondi la finestra FnsRiepilogo
    Application.StatusBar = "Please wait..."    'aggiorna la barra di stato

    Call InsConfigOttimale   'chiama la sub InsConfigOttimale
    Selection.MoveDown Unit:=wdLine, Count:=1   'sposta il cursore fuori dalla tabella
    Selection.TypeText vbCrLf   'vai a capo
    Call InserisciSottoscrizione    'chiama la sub InserisciSottoscrizione
    Selection.InsertBreak wdPageBreak   'inserisci un'interruzione di pagina

    Call InsConfigMedia
    Selection.MoveDown Unit:=wdLine, Count:=1
    Selection.TypeText vbCrLf
    Call InserisciSottoscrizione
    Selection.InsertBreak wdPageBreak

    Call InsConfigMinima
    Selection.MoveDown Unit:=wdLine, Count:=1
    Selection.TypeText vbCrLf
    Call InserisciSottoscrizione
    Selection.InsertBreak wdPageBreak
    Selection.TypeBackspace
    Selection.MoveDown wdLine

    Unload FnsRiepilogo     'chiudi la finestra
    Application.StatusBar = ""      'aggiorna la barra di stato
    Beep     'segnale acustico
    MsgBox "Fatto!", vbExclamation

End Sub
```

4.5.3 Subroutine InsConfigOttimale( )

La subroutine `InsConfigOttimale()` cicla tra tutte le checkbox associate alla configurazione ottimale e inserisce quelle attive (con relativi costi) in una tabella dedicata creata ad hoc dalla subroutine `CreaTabRiepilogo`() che verrà analizzata successivamente.
Anche in questo caso le prime operazioni posizionano correttamente il cursore, così da permettere una corretta formattazione

```
'INSERISCI CONFIGURAZIONE OTTIMALE
'inserisce nel documento la sezione relativa alla configurazione ottimale
Sub InsConfigOttimale()

    'vai al segnalibro NOTEGENERALI
    Selection.GoTo What:=wdGoToBookmark, Name:="NOTEGENERALI"
    'sposta il cursore indietro di un carattere
    Selection.MoveLeft Unit:=wdCharacter, Count:=1

    'inserisci il titolo
    Selection.Style = ActiveDocument.Styles("Capitolo")
    Selection.TypeText "Riepilogo economico - Configurazione OTTIMALE" & vbCrLf
    Selection.Style = ActiveDocument.Styles("Testo")
    Selection.TypeText vbCrLf

    numCap = 0
    Call CreaTabRiepilogo 'chiama la sub CreaTabellaRiepilogo
```

Successivamente viene eseguito il seguente blocco di codice per ciascun capitolo che ispeziona la checkbox associata alla configurazione ottimale, verifica se la rispettiva checkbox inerente la configurazione media è selezionata o meno e in caso affermativo richiama la subroutine `InserisciCostoServizio` con il parametro `bold=true` per aggiunge una nuova voce con numero e titolo del capitolo nella tabella di riepilogo formattata in grassetto, così da rendere evidenti le voci presenti nella configurazione ottimale che non sono invece presenti in quella media, in caso contrario viene inserita con carattere normale.

```
'Sviluppo Aspetto Grafico Del Sito Web
    'Sviluppo sito web Modello e-commerce
    If FnsRiepilogo.CheckBox1 Then
        nomeBm = "WEBSITOMOD"
        titoloCap = "Sviluppo sito web Modello e-commerce"
        Call TrovaNumCap(nomeBm, numCap)
        If FnsRiepilogo.CheckBox2 = False Then
            Call InserisciCostoServizio(nomeBm, numCap, titoloCap, true)
            Call InserisciCanoneAnnuo(nomeBm, true)
        Else
            Call InserisciCostoServizio(nomeBm, numCap, titoloCap ,false)
            Call InserisciCanoneAnnuo(nomeBm, false)
    End If
```

4.5.4 Subroutine InsConfigMedia( )

La subroutine `InsConfigMedia()` esegue le stesse operazione dell'ottimale, la verifica delle checkbox viene però effettuata su quelle appartenenti alla configurazione minima, così da evidenziare quelle non presenti su quest'ultima.

4.5.5 Subroutine InsConfigMinima( )

La subroutine `InsConfigMinima()` oltre ad inserire le voci selezionate nella configurazione minima all'interno della tabella come le subroutine precedenti, ne gestisce anche le ultime fasi di elaborazione e formattazione.
Per prima cosa viene creatala riga adibita a contenere la somma totale, successivamente per il conteggio vero e proprio viene richiamata, mantenendola nascosta all'utente, l'applicazione Microsoft Excel che dispone di librerie che permettono di eseguire somme aritmetiche.

```
    'alla fine della tabella inserisci una riga per il Totale
    Selection.InsertRowsBelow 1
    Selection.Style = ActiveDocument.Styles("TabTesto")
    Selection.TypeText "Totale Sviluppo"
    Selection.SelectRow
    Selection.ParagraphFormat.Alignment = wdAlignParagraphRight

    xlApp.Visible = False    'apri Excel
    Call SommaTOT(2)
    Call SommaTOT(3)
    xlApp.Quit               'chiudi Excel
    Set xlApp = Nothing

    'ombreggia la prima riga
    Selection.Tables(1).Rows(1).Shading.Texture = wdTexture10Percent

End Sub
```

4.5.6 Subroutine CreaTabRiepilogo( )

La subroutine `CreaTabRiepilogo()` è adibita alla creazione delle tabelle che riassumono i costi per ciascuna delle tre configurazioni, impostandone dimensioni e intestazioni.

```
Sub CreaTabRiepilogo()

    'inserisci una tabella 3x1
    ActiveDocument.Tables.Add Range:=Selection.Range, numRows:=1, NumColumns:=3, _
        DefaultTableBehavior:=wdWord9TableBehavior
    iRow = 1
    iTab = ActiveDocument.Tables.Count

    'imposta la larghezza delle colonne
    Selection.Tables(1).Columns(1).PreferredWidth = CentimetersToPoints(11.2)
    Selection.Tables(1).Columns(2).PreferredWidth = CentimetersToPoints(3.5)
    Selection.Tables(1).Columns(3).PreferredWidth = CentimetersToPoints(3)

    'applica stile e allinea il testo della tabella al centro a sinistra
    Selection.SelectRow
    Selection.Style = ActiveDocument.Styles("TabTesto")
    Selection.ParagraphFormat.Alignment = wdAlignParagraphLeft
    Selection.Cells.VerticalAlignment = wdCellAlignVerticalCenter

    'inserisci l'intestazione delle colonne
    Selection.TypeText Text:="Descrizione servizio:"
    Selection.MoveRight Unit:=wdCell
    Selection.TypeText Text:="Costo Servizio" & vbCr & "Euro"
    Selection.MoveRight Unit:=wdCell
    Selection.TypeText Text:="Canone Annuo" & vbCr & "Euro"

End Sub
```

4.5.7 Subroutine EnableCheckBox()

La subroutine `EnableCheckBox()` scorre una prima volta il documento creato per verificare la presenza o meno dei segnalibri associati a ciascun capitolo; se il segnalibro è presente viene incrementata la variabile `i`, utilizzata come puntatore per la prima posizione libera all'interno dell'array, e vengono abilitate le checkBox delle configurazioni ottima, media e minima corrispondenti al capitolo, così da evitare di poter aggiungere nel riepilogo economico voci che non sono realmente presenti nell'offerta commerciale.

```
'ABILITA LE CHECKBOX
Sub EnableCheckBox()
  'abilitale se nel documento è presente come segnalibro
  'il codice prodotto relativo al capitolo
  'e inserisci nell'array strTitoli
  'il segnalibro di riferimento per ogni capitolo

    Dim i As Integer
    i = 1

'Sviluppo Aspetto Grafico Del Sito Web
    'Sviluppo Siti Web Modello e-Com
    If ActiveDocument.Bookmarks.Exists("WEBSITOMOD") = True Then
        strTitoli(i) = "WEBSITOMOD"
        i = i + 1
        FnsRiepilogo.CheckBox1.Enabled = True
        FnsRiepilogo.CheckBox2.Enabled = True
        FnsRiepilogo.CheckBox3.Enabled = True
    End If
```

Al termine della subroutine l'array `strTitoli` è strutturato in modo da contenere nella cella i-esima il segnalibro del capitolo i-esimo dell'offerta, va ricordato infatti che grazie al comando "Option Base 1" la numerazione delle celle degli array parte da 1.

4.5.8 Subroutine TrovaNumCapitolo(…)

Per poter inserire all'interno delle tabelle di riepilogo economico delle configurazioni ottima, media e minima il numero esatto del capitolo per ciascuna voce che compone l'offerta è stato necessario sviluppare una funzione ad hoc in quanto al momento in VBA non è possibile leggere direttamente la parte numerica di un elenco numerato, utilizzato per formattare i capitoli. Proprio per permettere di risalire a quest'informazione la subroutine `EnableCheckBox`, oltre a verificare quali checkBox abilitare, popola anche l'array di stringhe `strTitoli`, come visto precedentemente.
La subroutine `TrovaNumCapitolo (ByVal bookmark As String, ByRef numCap As Integer)` riceve come valore il codice di un segnalibro e il riferimento alla variabile in cui è memorizzata la posizione dell'ultimo match nell'array `strTitoli`; la ricerca del bookmark all'interno di questo array infatti riprende dalla posizione successiva a quella dell'ultimo risultato positivo, una volta trovato viene aggiornato il valore della variabile `numCap` e l'esecuzione della subroutine viene termina.

```
Sub TrovaNumCap(ByVal bookmark As String, ByRef numCap As Integer)
    'il numero associato a ogni capitolo è un elenco numerato,
    'pertanto non può essere selezionato come testo
    'questa sub risale al numero di ogni capitolo attraverso
    'la ricerca all'interno di un array
    'popolato precedentemente dalla sub EnableCheckBox

    Dim flag As Boolean
    Dim i As Integer
    flag = True
    numCap = numCap + 1
    ' la ricerca all'interno dell'array riprende dall'ultimo match

    Do While flag
        If strTitoli(numCap) = bookmark Then
            flag = False
        Else
            numCap = numCap + 1
        End If
    Loop

End Sub
```

4.5.9 Subroutine InserisciCostoServizio(…)

La subroutine `InserisciCostoServizio(ByVal Bookmark As String, ByVal numCap As Integer, ByVal title As String, ByVal bold As Boolean)` viene richiamata quando il controllo fatto su una checkbox dalle subroutine per l'inserimento delle configurazioni risulta essere positivo.

Riceve quattro parametri passati per valore:

- `bookmark`: il codice del segnalibro relativo alla voce dell'offerta;
- `numCap`: il numero del capitolo;
- `title`: la stringa del titolo del capitolo per esteso;
- `bold`: parametro booleano utilizzato per distinguere l'inserimento del testo in grassetto dal testo normale.

```
Sub InserisciCostoServizio(ByVal bookmark As String, ByVal numCap As Integer,
                           ByVal title As String, ByVal bold As Boolean)
    'inserisci una riga
    Selection.InsertRowsBelow 1
    'incrementa il contatore di righe, serve a excel
    iRow = iRow + 1
    'applica stile
    Selection.Style = ActiveDocument.Styles("TabTesto")
    'inserisci num e titolo capitolo ricevuti come argomento
    Selection.TypeText numCap & ") " & title

    If ActiveDocument.Bookmarks.Exists(bookmark & "PR") Then
        'seleziona e copia il costo servizio
        Selection.GoTo What:=wdGoToBookmark, Name:=bookmark & "PR"
        Selection.EndKey Unit:=wdLine, Extend:=wdExtend
        Selection.MoveLeft Unit:=wdCharacter, Count:=1, Extend:=wdExtend
      On Error GoTo HandlerPR:
        Selection.Copy
        'torna nella tabella
        ActiveDocument.Tables(iTab).Rows(iRow).Select
        'vai nella cella di destinezione e incolla
        Selection.MoveRight Unit:=wdCell, Count:=2
        Selection.Paste
        Selection.SelectCell
        'formatta il contenuto
        Selection.ParagraphFormat.Alignment = wdAlignParagraphRight
        Selection.SelectRow
        Selection.Font.Size = 10
        ' se flag = true formatta il testo in grassetto
        If bold Then
            Selection.Font.Bold = True
        End If
    End If
Exit Sub
```

Quando viene richiamata il cursore si trova già all'interno della tabella di riepilogo pertanto aggiunge una riga e scrive all'interno della prima cella numero e titolo del capitolo passati per valore; per poter inserire il costo del servizio invece viene fatta una ricerca per segnalibro all'interno del documento, in tutti i testi originali che vengono copiati ed incollati all'interno del documento di destinazione infatti è presente, appena prima dell'indicazione del costo, un segnalibro formattato con lo stesso codice del capitolo e il suffisso "PR".
Una volta costruita la stringa di ricerca con il parametro bookmark e individuato il segnalibro corrispondente, la cifra viene copiata ed incollata all'interno della tabella nella cella corrispondente.
In ultima battuta viene effettuato il controllo sul parametro booleano `bold`, se un voce dell'offerta infatti non compare nella configurazione inferiore allora l'intera riga viene formattata in grassetto, ad esempio se una voce è compresa nella

configurazione ottima ma non nella media, la riga corrispondente nella tabella di riepilogo per la ottima viene formattata in grassetto così da rendere più evidenti le caratteristiche aggiuntive.

4.5.10 HandlerPR

Nel caso in cui la voce dell'offerta non preveda un costo per il servizio, il tentativo di selezione della cifra solleva un' eccezione gestita dal blocco di codice `HandlerPR` che sposta il cursore all'interno della tabella di riepilogo e inserisce la stringa 0,00.

```
HandlerPR:
    If Err = 4605 Then
        'torna nella tabella
        ActiveDocument.Tables(iTab).Rows(iRow).Select
        'vai nella cella di destinezione e incolla
        Selection.MoveRight Unit:=wdCell, Count:=2
        'inserisci costo zero
        Selection.TypeText "0,00"
        'formatta il contenuto
        Selection.SelectCell
        Selection.ParagraphFormat.Alignment = wdAlignParagraphRight
    End If
End Sub
```

La dichiarazione `HandlerPR`: funge da puntatore e rende le istruzioni che seguono rintracciabili, mentre la riga di codice `On Error Go To HandlerPR` indica a VBA da dove proseguire con l'esecuzione della subroutine nel caso in cui venga sollevata un'eccezione.

4.5.11 Subroutine InserisciCanoneAnnuo(...)

La subroutine `InserisciCanoneAnnuo(ByVal Bookmark As String, ByVal numCap As Integer, ByVal title As String, ByVal bold As Boolean)` viene richiamata quando la voce che si sta inserendo prevede anche un costo per il canone annuale di mantenimento.

Riceve due parametri passati per valore:

- `bookmark`: il codice del segnalibro relativo alla voce dell'offerta;

- `bold`: parametro booleano utilizzato per distinguere l'inserimento del testo in grassetto dal testo normale.

```
Sub InserisciCanoneAnnuo(ByVal bookmark As String, ByVal bold As Boolean)

    If ActiveDocument.Bookmarks.Exists(bookmark & "CA") Then
        'seleziona e copia il canone annuo
        Selection.GoTo What:=wdGoToBookmark, Name:=bookmark & "CA"
        Selection.EndKey Unit:=wdLine, Extend:=wdExtend
        Selection.MoveLeft Unit:=wdCharacter, Count:=1, Extend:=wdExtend
          On Error GoTo HandlerCA:
            Selection.Copy
            'torna nella tabella
            ActiveDocument.Tables(iTab).Rows(iRow).Select
            'vai nella cella di destinezione e incolla
            Selection.MoveRight Unit:=wdCell, Count:=3
            Selection.Paste
            Selection.SelectCell
            'formatta il contenuto
            Selection.ParagraphFormat.Alignment = wdAlignParagraphRight
            Selection.SelectRow
            Selection.Font.Size = 10
            If flag Then
                'testo in grassetto
                Selection.Font.Bold = True
            End If
        End If
Exit Sub
```

Nei testi originali delle voci che prevedono un canone annuale è presente, così come per i costi di attivazione, un segnalibro formattato con lo stesso codice del capitolo e il suffisso "CA" appena prima dell'indicazione del costo, pertanto quando la subroutine avvia la ricerca per segnalibro all'interno del documento, una volta individuato la cifra viene copiata ed incollata all'interno della tabella nella cella corrispondente.
Anche in questo caso viene effettuato il controllo sul parametro booleano `bold`, per formattare l'intera riga in grassetto nel caso in cui sia impostato a true.

4.5.12 HandlerCA

HandlerCA gestisce lo stesso tipo di eccezione di `HandlerPR` ma il blocco di istruzioni è specifico per la subroutine `InserisciCanoneAnnuo(…)`

```
HandlerCA:
    If Err = 4605 Then
        'torna nella tabella
        ActiveDocument.Tables(iTab).Rows(iRow).Select
        'vai nella cella di destinezione e incolla
        Selection.MoveRight Unit:=wdCell, Count:=3
        'inserisci stringa vuota
        Selection.TypeText ""
        'formatta il contenuto
        Selection.SelectCell
        Selection.ParagraphFormat.Alignment = wdAlignParagraphRight
        Selection.SelectRow
        Selection.Font.Size = 10
        If bold Then
            'testo in grassetto
            Selection.Font.Bold = True
        End If
    End If

End Sub
```

4.5.13 Subroutine SommaTOT(…)

Microsoft Word nativamente non dispone di funzioni che permettano di eseguire una somma algebrica, a tale scopo è stata implementata la subroutine `SommaTOT(ByVal numCol As Integer)` che sfrutta le funzioni implementate in Microsoft Excel.
L'unico parametro passato per valore serve per identificare di quale colonna della tabella di riepilogo selezionare le celle, 2 corrisponde alla seconda colonna, ovvero i costi di attivazione, 3 alla terza, ovvero i canoni annuali.
La prima parte della subroutine infatti conta il numero di righe dell'ultima tabella inserita (quella di interesse) e seleziona il contenuto delle cella della colonna indicata.

```
Sub SommaTOT(ByVal numCol As Integer)

    'contatore numero righe della tabella su cui fare la somma
    numRows = ActiveDocument.Tables(iTab).Rows.Count
    'contatore num tabelle nel doc
    '(l'ultima inserita è quella che interessa)
    iTab = ActiveDocument.Tables.Count
    'seleziona la seconda riga (la prima ha solo titoli)
    ActiveDocument.Tables(iTab).Rows(2).Select
    'sposta il cursore al numCol in argomento
    '(2 se costo servizio,3 se canone annuo)
```

```
    Selection.MoveRight Unit:=wdCell, Count:=numCol
    'seleziona tutte le righe di interesse,prima e ultima non servono
    '(titoli e totale), la prima è già selezionata, quindi numRows-3
    Selection.MoveDown Unit:=wdLine, Count:=numRows - 3, Extend:=wdExtend

    'se la selezione è vuota viene generato un errore
    On Error GoTo HandlerTOT:
        Selection.Copy
```

Nel caso in cui l'esecuzione del comando `Copy`, in particolare se la selezione è vuota, sollevi un'eccezione, questa viene intercettata dal gestore degli errori che rimanda all'esecuzione del codice alla riga `HandlerTOT`, in caso contrario successivamente viene effettuato un controllo del numero voci presenti nella tabella in quanto se ve ne è una sola non è necessario eseguire la somma.

```
    'usa excel solo se la tabella ha almeno due servizi
    If numRows - 3 > 0 Then

        'crea un nuovo workbook, incolla i prezzi
        Set xlWB = xlApp.Workbooks.Add
        xlWB.Worksheets(1).Paste
        xlApp.Selection.NumberFormat = "#,##0.00"
        'seleziona le celle e ne calcola il totale
        xlWB.Worksheets(1).Cells(numRows - 1, 1).FormulaR1C1 =
                                    "=SUM(R[-" & numRows - 2 & "]C:R[-1]C)"
        'incolla il risultato nella prima cella libera della prima colonna
        xlWB.Worksheets(1).Range("A" & numRows - 1).Select
        xlApp.Selection.Copy
        Selection.MoveDown Unit:=wdLine 'sposta il cursore nella cella
        Selection.Paste    'incolla il totale
        Selection.SelectCell
        Selection.Font.Size = 10
        xlWB.Close False ' close the workbook without saving
        Set xlWB = Nothing

    'altrimenti copia e incolla nella cella del totale l'unico prezzo
    Else
        Selection.MoveDown Unit:=wdLine
        Selection.Paste
    End If

Exit Sub
```

Se la tabella ha più di due voci viene inizializzata la variabile `xlWB` come nuovo foglio di calcolo e il contenuto copiato in precedenza negli appunti viene incollato in colonna a partire dalla prima cella del foglio 1.

L'indirizzamento avviene tramite coordinate del tipo (numero riga, numero colonna) riferite alla cella in cui è posizionato il cursore e ad eseguire il calcolo è l'istruzione `FormulaR1C1` che permette di richiamare le funzioni implementate in Microsoft Excel.
In questo caso viene eseguita la funzione somma(`=SUM`) utilizzando come addendi i singoli contenuti delle celle comprese nell'intervallo costruito dinamicamente.
Una volta completato il calcolo, il risultato viene riportato nella tabella di riepilogo economico, il foglio di calcolo chiuso senza salvare le modifiche e la subroutine può così terminare.

4.5.14 HandlerTOT

In caso di eccezione non viene eseguita nessuna somma, semplicemente il cursore viene posto all'ultima riga della tabella di riepilogo

```
HandlerTOT:
    If Err = 4605 Then
        'in caso di errore sposta il cursore all'ultima riga della tabella
        Selection.MoveDown Unit:=wdLine, Count:=numRows - 2
    End If

End Sub
```

4.5.15 Subroutine InserisciSottoscrizione()

La subroutine `InserisciSottoscrizione()` aggiunge staticamente il modulo del contratto da compilare e sottoscrivere per accettare l'offerta commerciale e viene richiamata dopo l'inserimento del riepilogo di ciascuna configurazione, ottimale, media e minima.

```
Sub InserisciSottoscrizione()

    ActiveDocument.Tables.Add Range:=Selection.Range, numRows:=1, NumColumns:=3, _
        DefaultTableBehavior:=wdWord9TableBehavior

    Selection.Tables(1).Columns(1).PreferredWidth = CentimetersToPoints(5.2)
    Selection.Tables(1).Columns(2).PreferredWidth = CentimetersToPoints(10.5)
    Selection.Tables(1).Columns(3).PreferredWidth = CentimetersToPoints(1.7)

    Selection.SelectRow
    Selection.Style = ActiveDocument.Styles("TabTesto")
    Selection.ParagraphFormat.Alignment = wdAlignParagraphLeft
    Selection.TypeText "Pagamenti"
    Selection.InsertRowsBelow 1
```

```
Selection.TypeText "Acconto all'ordine:" & vbCrLf & "1/3 servizi + canoni annui"
Selection.MoveRight Unit:=wdCharacter, Count:=1
Selection.TypeText _
    "Bonifico su Unicredit Spa, Chiarano - CC 000029380529" & vbCrLf & _
    "cod Cin J -ABI:02008-CAB:61600" & vbCrLf & "IBAN IT96J0200861600000029380529"
Selection.InsertRowsBelow 1
Selection.TypeText "Avanzamento alla pubblicazione versione beta:"
                                                & vbCrLf & "1/3 dei servizi"
Selection.MoveRight Unit:=wdCharacter, Count:=1
Selection.TypeText _
    "Bonifico su Unicredit Spa, Chiarano - CC 000029380529" & vbCrLf & _
    "cod Cin J -ABI:02008-CAB:61600" & vbCrLf & "IBAN IT96J0200861600000029380529"
Selection.InsertRowsBelow 1
Selection.TypeText "Saldo 30 giorni data pubblicazione:" & vbCrLf & "1/3 dei servizi"
Selection.MoveRight Unit:=wdCharacter, Count:=1
Selection.TypeText _
    "Bonifico su Unicredit Spa, Chiarano - CC 000029380529" & vbCrLf & _
    "cod Cin J -ABI:02008-CAB:61600" & vbCrLf & "IBAN IT96J0200861600000029380529"
Selection.InsertRowsBelow 1
Selection.TypeText "Canone dal secondo anno"
Selection.MoveRight Unit:=wdCharacter, Count:=1
Selection.TypeText "Ricevuta Bancaria"
Selection.InsertRowsBelow 1
Selection.TypeText "Budget da destinare a Google"
Selection.MoveRight Unit:=wdCharacter, Count:=1
Selection.TypeText "Pagamento diretto via BB o carta di credito a Google Italia"
Selection.Tables(1).Rows(1).Shading.Texture = wdTexture10Percent
Selection.MoveDown Unit:=wdLine, Count:=1
Selection.Style = ActiveDocument.Styles("Testo")
Selection.TypeText vbCrLf
Selection.ParagraphFormat.Alignment = wdAlignParagraphCenter
Selection.TypeText "PER CONFERMA INVIARE LA PRESENTE COMPILATA"
                  _"IN OGNI SUA PARTE TRAMITE FAX " & vbCrLf & "ALLO 0422 806952"
Selection.TypeText vbCrLf & vbCrLf
Selection.TypeText "DICHIARO DI AVER PRESO VISIONE DELLE CONDIZIONI "
             _"GENERALI DI FORNITURA E CONFERMO" & vbCrLf & "L'ORDINE SOPRA RIPORTATO"
Selection.TypeText vbCrLf & vbCrLf
Selection.ParagraphFormat.Alignment = wdAlignParagraphJustify

Selection.TypeText "RAGIONE SOCIALE:" & vbTab & vbTab &
          "_____" & vbTab & vbTab & "DATA___/___/___"
Selection.TypeText vbCrLf
Selection.Font.Size = 5
Selection.TypeText vbCrLf
Selection.Style = ActiveDocument.Styles("Testo")
Selection.TypeText "SEDE LEGALE:" & vbTab & vbTab & vbTab &
                              "_____" & vbTab & vbTab & _
    "TIMBRO E FIRMA"
Selection.TypeText vbCrLf
Selection.Font.Size = 5
Selection.TypeText vbCrLf
Selection.Style = ActiveDocument.Styles("Testo")
Selection.TypeText "COD.FISCALE E P.IVA" & vbTab & vbTab
                                        & "_____"
Selection.TypeText vbCrLf
Selection.Font.Size = 5
Selection.TypeText vbCrLf
Selection.Style = ActiveDocument.Styles("Testo")
Selection.TypeText "IBAN" & vbTab & vbTab & vbTab & vbTab
                                          & "_____"
```

```
    Selection.TypeText vbCrLf
    Selection.Font.Size = 5
    Selection.TypeText vbCrLf
    Selection.Style = ActiveDocument.Styles("Testo")
    Selection.TypeText Text:="BANCA ABI E CAB" & vbTab & vbTab
                                                & "_____"
    Selection.TypeText vbCrLf
    Selection.Font.Size = 5
    Selection.TypeText vbCrLf
    Selection.Style = ActiveDocument.Styles("Testo")
    Selection.TypeText Text:="TEL E FAX" & vbTab & vbTab & vbTab
                                                & "_____"
    Selection.TypeText vbCrLf
    Selection.Font.Size = 5
    Selection.TypeText vbCrLf
    Selection.Style = ActiveDocument.Styles("Testo")
    Selection.TypeText Text:="INDIRIZZO E-MAIL" & vbTab & vbTab
                                                & "_____"
    Selection.TypeText vbCrLf
    Selection.Font.Size = 5
    Selection.TypeText vbCrLf
    Selection.Style = ActiveDocument.Styles("Testo")
    Selection.Font.Size = 9
    Selection.TypeText "(questo indirizzo serve solamente per inviare le fatture; " & _
        "è richiesto anche il nome del responsabile per le fatturazioni)"
    Selection.TypeText vbCrLf
    Selection.Style = ActiveDocument.Styles("Testo")

End Sub
```

5 | CONCLUSIONI

5.1 Sviluppi Futuri

L'applicazione descritta, dopo una prima fase di test, è stata rilasciata a tutti i commerciali dell'azienda ed ha già subito diversi aggiornamenti dalla prima release per allineare le voci presenti con i servizi offerti da Promoservice srl e viene tutt'ora utilizzata.
Qualora venisse meno il requisito di poter essere modificabile anche da chi non possiede propriamente competenze nella programmazione si potrebbero certamente apportare delle migliorie, lavorando sulla qualità del codice, abbandonando l'approccio modulare seguito o anche passando ad un'applicazione stand-alone sviluppata in un ambiente di programmazione più evoluto.

5.2 Considerazioni sull'esperienza

È stata sicuramente un'esperienza positiva che mi ha dato modo di applicare alcuni dei concetti appresi nel corso di studi e di apprenderne di nuovi, la tipologia del progetto poi, mi ha permesso di svilupparlo in autonomia, a partire dall'apprendimento del linguaggio utilizzato avendo già a disposizione, come bagaglio acquisito durante il corso di studi, gli strumenti necessari per portarne avanti la realizzazione e le capacità per apprendere gli aspetti che non conoscevo.
Il tirocinio si è svolto con reciproca soddisfazione mia e del tutore e si è concluso la proposta di lavorare all'interno dell'azienda, dandomi l'occasione d' imparare i principali linguaggi utilizzati nello sviluppo web come ad esempio HTML, CSS, Javascript.

5.3 Ringraziamenti

Intendo rivolgere un sentito ringraziamento verso le seguenti persone:
il mio tutore e relatore, il professor Sergio Congiu per la disponibilità e pazienza; il mio tutore aziendale Luca Vescovi per avermi accolto e seguito nel corso del tirocinio e i colleghi di Promoservice, in particolar modo Sara e Nicola per i preziosi suggerimenti e consigli; la mia famiglia, per avermi offerto l'occasione di portare a termine questo corso di laurea e avermi dato sostegno in questi anni; i miei amici e compagni di studio della sede di Treviso, Luca, Alessandro, Alberto e Fabio;

Bibliografia:

1. Mastering VBA for Microsoft Office 2007
2. Mastering VBA for Microsoft Office 2010
3. Microsoft Office Developer Center
   <http://msdn.microsoft.com/en-us/library/office/>