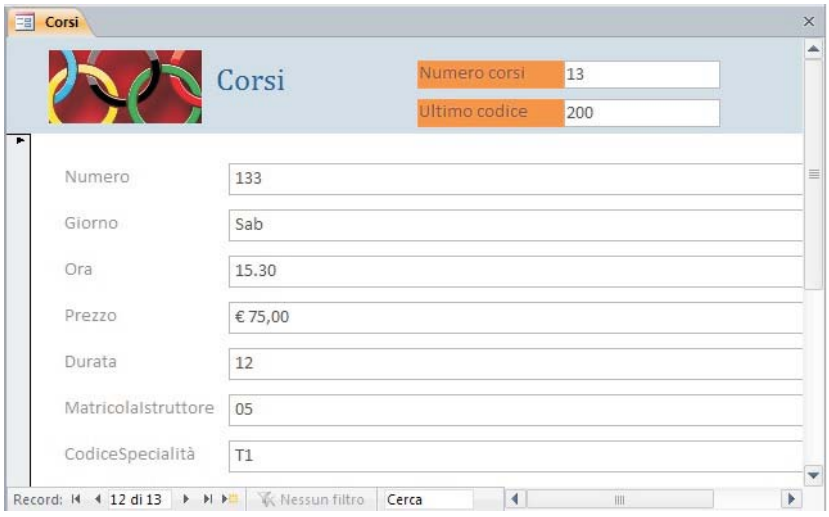
Controllo sulle attività dell'utente

L'uso del linguaggio Visual Basic permette di combinare le funzionalità del programma Access con la flessibilità del codice che consente di introdurre, all'interno delle applicazioni software, controlli e validazioni sulle attività dell'utente finale, oppure elementi che facilitano l'interazione dell'utente con il database.

PROGETTO - Nella maschera dei Corsi, visualizzare il numero di corsi già inseriti nel database e l'ultimo codice utilizzato come numero di corso. Controllare anche che non venga assegnato a un nuovo corso un numero di codice che è già presente nella tabella dei Corsi.

Inseriamo nella sezione *Intestazione maschera* della maschera *Corsi* due caselle di testo, con gli stessi valori per le proprietà **Etichetta** e **Nome elemento**: *NumeroCorsi* per visualizzare il numero di record presenti nella tabella *Corsi* e *UltimoCodice* per visualizzare l'ultimo numero utilizzato come codice del corso.



Apriamo la finestra del codice per la maschera *Corsi*, facendo clic sul pulsante **Visualizza codice,** nel gruppo **Strumenti** della scheda **Progettazione**: dalla casella in alto a sinistra (*Oggetto*) scegliamo **Form** e dalla casella a destra (*Routine*) scegliamo **Load** (caricamento della maschera).
Scriviamo poi il seguente codice che descrive le operazioni da eseguire quando la maschera viene aperta:

```
Private Sub Form_Load()

Dim Totale, Ultimo As Integer

Totale = DCount("*", "Corsi")
Ultimo = DMax("Numero", "Corsi")
Me!NumeroCorsi = Totale
Me!UltimoCodice = Ultimo
Me!Numero.SetFocus

End Sub
```

La funzione **DCount** restituisce il numero dei record presenti nella tabella *Corsi*, mentre la funzione **DMax** restituisce il massimo tra i valori assegnati al campo *Numero* della tabella. Questi valori, dopo essere stati assegnati a due variabili di tipo *Integer*, vengono scritti nelle caselle di testo predisposte nell'intestazione della maschera.
L'oggetto **Me** (me stessa) indica la maschera stessa *Corsi*.
Il metodo **SetFocus**, applicato a un controllo dell'interfaccia grafica, lo rende attivo: in questo caso il cursore viene posizionato nella casella *Numero* della maschera.

La seconda parte del problema riguarda il controllo sul valore assegnato al codice di un nuovo corso: il programma non deve accettare valori già esistenti perché il numero del corso è una chiave.
La validazione del numero di corso viene associata all'evento **BeforeUpdate** (Prima di aggiornare) sul controllo *Numero* (casella di testo) della maschera *Corsi*.
Nella finestra del codice associato alla maschera *Corsi*, scegliamo *Numero* dalla casella in alto a sinistra (*Oggetto*) e **BeforeUpdate** dalla casella *Routine* in alto a destra.
Scriviamo poi nella finestra il seguente codice per il controllo:

```
Private Sub Numero_BeforeUpdate(Cancel As Integer)
On Error GoTo Err_ConvalidaNumero
If Not IsNull(DLookup("[Numero]", "[Corsi]", "[Numero] =
   Form.[Numero]")) Then
   ' visualizza un messaggio se il numero corso esiste già
   MsgBox "Inserire un nuovo valore", vbCritical,
          "Codice corso duplicato"
   ' ritorna alla casella Numero
   DoCmd.CancelEvent
End If

Exit_ConvalidaNumero:
    Exit Sub

Err_ConvalidaNumero:
    MsgBox Err.Description
    Resume Exit_ConvalidaNumero
End Sub
```

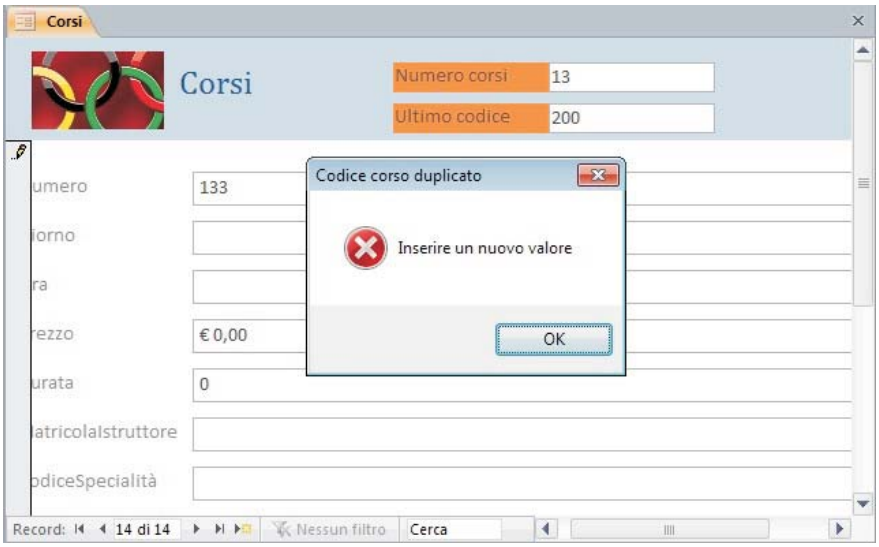
La funzione **DLookup** cerca all'interno dei valori di un campo di una tabella quelli che soddisfano a un criterio.
La sintassi generale è:

DLookup(*campo, tabella, criterio di ricerca*)

Se il valore restituito è nullo, significa che la ricerca ha prodotto esito negativo.
Il codice precedente controlla che il valore restituito da *DLookup* non sia nullo:

```
If Not IsNull(DLookup(. . .)) Then . . .
```

Infatti, se il valore di *DLookup* non è nullo, significa che esiste un valore del campo *Numero* che è uguale al valore scritto nella casella *Numero* della maschera (*Form.[Numero]*). In questo caso viene inviato un messaggio di errore all'utente e l'operazione di aggiornamento sul campo *Numero* viene annullata attraverso il comando **DoCmd.CancelEvent**: il cursore si riposiziona sulla casella *Numero* della maschera.



La routine precedente *BeforeUpdate* mostra anche un esempio di utilizzo della struttura **On Error** del linguaggio Visual Basic che controlla le situazioni di errore. La struttura generale è:

On Error GoTo *EtichettaErrore*
.....
.....

EtichettaUscita
Exit Sub
Etichetta Errore
MsgBox *MessaggioErrore*
Resume *EtichettaUscita*

In assenza di errori il sottoprogramma esegue normalmente le istruzioni scritte dopo **On Error** ... ed esce quando incontra l'istruzione **Exit Sub**. Se si verifica un errore, invece, il controllo passa alle istruzioni contrassegnate dall'etichetta indicata dopo **GoTo**: l'istruzione **MsgBox** visualizza il messaggio di errore del sistema e l'istruzione **Resume** riprende il controllo dell'esecuzione passando alle istruzioni contrassegnate con *EtichettaUscita*.

Al termine del Paragrafo 2 di questo capitolo è stato presentato un esempio di validazione, utilizzando una macro di dati associata all'evento **Prima della modifica** di una maschera, con la quale è stato implementato un **trigger.**
L'evento *Prima della modifica* è rappresentato nel linguaggio Visual Basic con l'evento **BeforeUpdate** illustrato nel progetto precedente.
In particolare il codice associato all'evento *BeforeUpdate* risulta particolarmente utile per attivare un *trigger* quando l'utente lascia vuote le caselle di una maschera corrispondenti a campi obbligatori (per esempio il campo *CodiceFiscale*) oppure inserisce valori non corretti per un campo (per esempio un numero di caratteri per il *CodiceFiscale* diverso da 16 caratteri).