# CHAPTER IV - INSTRUCTORS' HOME PAGE

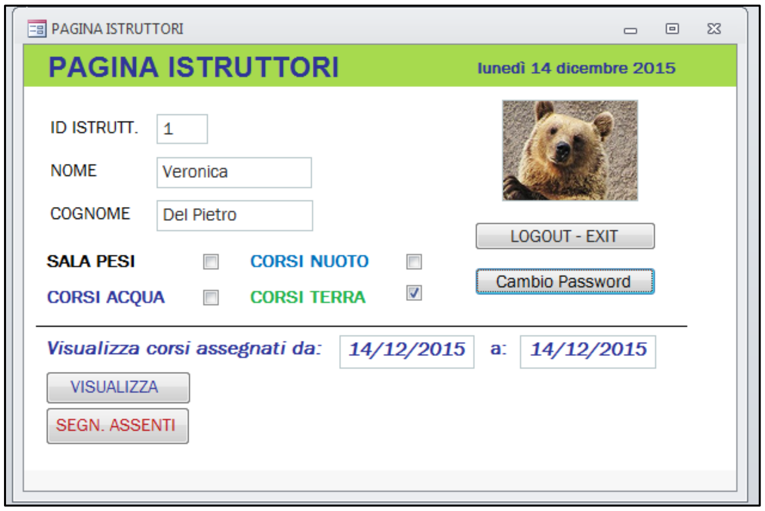
## 1. Introduction - The instructors' home page

Instructors have a personal page that, as shown in Figure 4.1, reports some basic information: name, surname and kind/typology of courses that can be assigned to them.

The personal page also allows instructors to:

- Change password;
- Check and filter (by date) the list of classes (i.e., courses) assigned to them;
- Print the list of people who registered for a specific class;
- Signal people that did not show off at a course (i.e., block absent people).

Please note that, the lasts three functions are not available for those instructors who make assistance in the weight rooms, but do not teach in any course.



ID ISTRUTT. 1
NOME Veronica
COGNOME Del Pietro
SALA PESI ☐ CORSI NUOTO ☐
CORSI ACQUA ☐ CORSI TERRA ☑
Visualizza corsi assegnati da: 14/12/2015 a: 14/12/2015

**Fig. 4.1.** *The Personal Page of the instructors*

### *2.1 Form Load*

The Personal Page is an "un-linked" form fed with data collected (from the Data Base) as soon as the form is loaded. Clearly, since displayed data depend on the logged instructor, data are filtered depending on the ID of the instructor. As usual, the value of the ID is passed to the form using the OpenArg argument, of the OpenForm method. The full code is shown below.

```
Private Sub Form_Load()
Dim Rs As DAO.Recordset
Dim Db As Database
' WRoom is a Boolean variable (with visibility limited to the current Form)
' It is True if the instructor gives assistance in the weight room,
WRoom = False
LoggedUser = Me.OpenArgs
Set Db = CurrentDb
Set Rs = Db.OpenRecordset("INSTRUCTORS", dbOpenSnapshot, dbReadOnly)
    Rs.FindFirst "[ID]=" & LoggedUser
 ' The recordset is used to collect data and to fill the form’s Text boxes and Check Boxes
    Me.TxtName = Rs!Name
    Me.TxtSurname = Rs!Surname
    Me.TxtID = Rs![ID]
    Me.ChBSwim = Rs![Swim Courses] ' Check Boxes
    Me.ChBEarth = Rs![Earth Courses]
    Me.ChBWeight = Rs![Weight Room]
    Me.ChBAqua = Rs![Water Courses]
    Me.Immagine = Rs![Picture]
   Rs.Close
  ' If the only duty is to make assistance in the weight rooms, then some options are not available
  If Me.ChBSwim = False And Me. ChBEarth = False And Me. ChBAqua = False Then
    Me.LblTimeTable.Caption = "Show Time Table"
    Me.CmdAbsents.Visible = False
    Me.Lbl2.Visible = False
    Me.TxtDataStart.Visible = False
    Me.TxtDataEnd.Visible = False
    WRoom = True
  Else    ' If the instructor has some courses, all the options are available
    Me.CmdAssignedClassed.Visible = True
    Me.CmdAbsents.Visible = True
    Me.LblTimeTable.Caption = "Show Assigned Classes from:"
    Me.Lbl2.Visible = True
   ' These are the filtering dates
    Me.TxtDataStart.Visible = True
    Me.TxtDataEnd.Visible = True
  End If
Set Rs = Nothing
Set Db = Nothing
' When the form opens, the filtering fields are pre-filled to show only classes scheduled within 7 days from now
Me.TxtDataStart.Value = Date
Me.TxtDataEnd.Value = DateAdd("d", 7, Date)
End Sub
```

As it can be seen, when the form is loaded, the filtering fields TxtDAtaStart and TxtDataEnd are pre-filled using the Date() and DataAdd() functions, respectively. In this way, when the assigned courses’ list is opened, classes are filtered and only those ones, scheduled within the next seven days are shown.

Indeed:

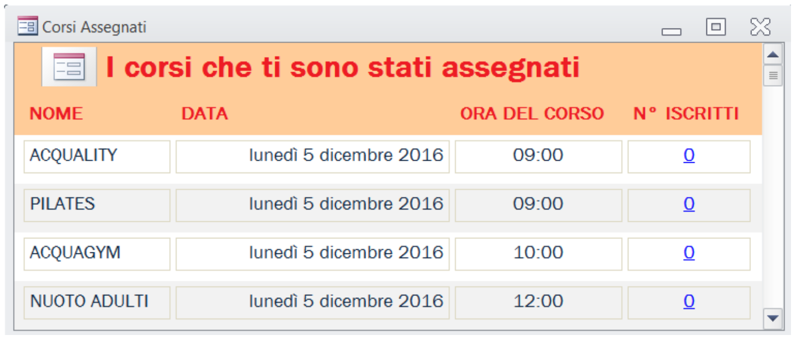
- *Me.TxtDataStart.Value = Date()* - This function is used to set the starting date to the current date;
- *Me.TxtDataEnd.Value = DateAdd("d", 7, Date)* - This is used to set the ending date 7 days from now.

Also note that the Show button (namely CmdShow) is never called inside the above-mentioned procedure. This button is always visible (both in case of instructors with or without assigned courses) but, as shown in the code below, the code triggered by the on click event changes, depending on the value of the Boolean variable WRoom (a variable that has a visibility limited to the current form and that is true if the logged instructor does assistance in the weights room). If this Boolean variable is true, the button opens the time table of the Weight rooms; conversely, it opens the list of all the classes assigned (within the selected dates) to the logged instructor.

The full code is shown below:

```
' This button shows either assigned curses or the Weight Room Time Table, depending on the value of the
' Boolean variable WRoom which has a local visibility
Private Sub CmdShow_Click()
Dim WhereCond As String
  If Not WRoom Then
' A Filtering condition is written; it will be used to filter data displayed in the Assigned Courses Form
    WhereCond = "InstructorID = " & Me.TxtID
    WhereCond = WhereCond & " AND [Class Date] >= " & Format(Me.TxtDataStart, "\#mm\/dd\/yyyy\#")
    WherCond = WhereCond & " AND [Class Date] <= " & Format(Me.TxtDataEnd, "\#mm\/dd\/yyyy\#")
    DoCmd.OpenForm "Assigned Courses", acNormal, , WhereCond, acFormReadOnly, acDialog, Me.TxtID
  Else
    DoCmd.OpenForm "Weight Room Time Table", acNormal
  End If
End Sub
```

As above mentioned, if WRoom is true, the button shows the weekly time table of the Weights rooms (a standard pivot query). If not so, the button opens the Assigned Courses form, shown in Figure 4.2:



I corsi che ti sono stati assegnati

| NOME | DATA | ORA DEL CORSO | N° ISCRITTI |
|---|---|---|---|
| ACQUALITY | lunedì 5 dicembre 2016 | 09:00 | 0 |
| PILATES | lunedì 5 dicembre 2016 | 09:00 | 0 |
| ACQUAGYM | lunedì 5 dicembre 2016 | 10:00 | 0 |
| NUOTO ADULTI | lunedì 5 dicembre 2016 | 12:00 | 0 |

**Fig. 4.2.** *The Assigned Courses Form*

This form is linked to the same query, namely “Time Table With Dates”, used to generate the “Available Classes” form, described in Chapter III. In this case, to show only the courses assigned to the logged instructor, a filtering condition is built, based on the instructor ID and on the selected beginning and ending dates.

Lastly, we note that, if there are not classes assigned to the instructor within the selected time period, then a message box saying “*you do not have classes in this period*” is displayed on the screen. This check is made on the On Load event of the Assigned Classes form. Specifically, as we have seen in the code above, the Assigned Courses form is open with a filtering condition:

```
    DoCmd.OpenForm "Assigned Courses", acNormal, , WhereCond, acFormReadOnly, acDialog, Me.TxtID
```

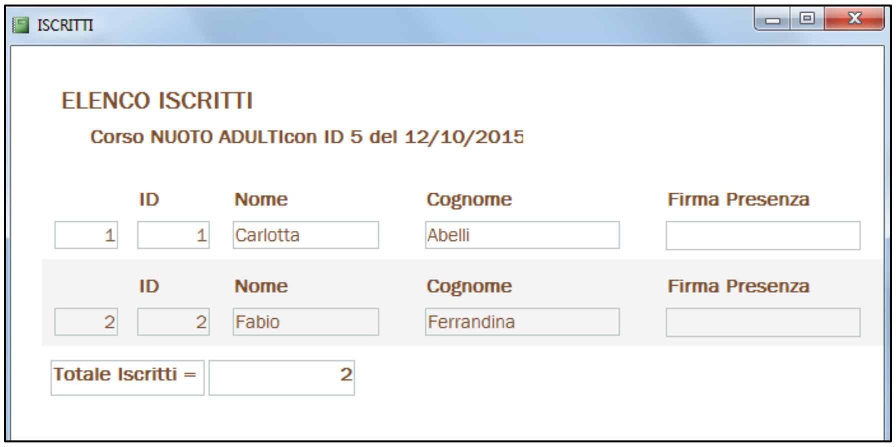
If the condition returns some records (i.e., there are courses assigned to the logged instructor) the RecordCount property of the recordset linked to the table will be greater than zero and it will be zero otherwise. So, when the form is loaded, we can check this condition to verify whether some courses have been assigned to an instructor. The full code is shown below.

```
Private Sub Form_Load()
Dim Rs As DAO.Recordset
  LoggedUser  = Me.OpenArgs
  Set Rs = Me.RecordsetClone ' A clone of the recordset linked to the form is made
  If Rs.RecordCount < 1 Then ' The recordset is empty!!! There are not classes
    MsgBox ("you do not have classes in this period ")
    Rs.Close
    Set Rs = Nothing
    DoCmd.Close
  Else
    Rs.Close
    Set Rs = Nothing
  End If
End Sub
```

Note that this check was made using the same Cloning technique described in Chapter III.

## 2. The list of people that registered to a class - Changing the Record Origin at run time

Once the Assigned Classes form has been opened, the instructor can click on the number of registrations of a specific class, in order to open a report with the names of all the people that registered to that class. An example of the Registrations form is shown in Figure 4.3.


ISCRITTI

ELENCO ISCRITTI

Corso NUOTO ADULTIcon ID 5 del 12/10/2015

| | ID | Nome | Cognome | Firma Presenza |
|---|---|---|---|---|
| 1 | 1 | Carlotta | Abelli | |
| 2 | 2 | Fabio | Ferrandina | |

Totale Iscritti = 2

**Fig. 4.3.** *The Registrations Form*

This report is particularly interesting as it shows how it is possible to change, dynamically, the Record Source (Origine Record) property of a report at run time (the same can be made for a form, too).

To this aim it is sufficient to:

- Write a valid SQL code (to collect all data needed to populate the report);
- Assign it to a MySQL string variable;
- Assign the string to the RecordSource Property of the Report.

For instance, in the present case, the following instruction was used:

```
  Reports![Registrations].RecordSource = MySQL
```

Where Reports is the collection of all the reports of the information system and MySQL is the string containing the SELECT query (with the WHERE condition) used to collect and filter data. Specifically, since we want the list of people that signed to a class, the query performs a Join on the RESERVATIONS and on the CUSTOMERS tables and has a structure similar to the following one.

**SELECT** CustomerId, CustomerName, CustomerSurname

**FROM** CUSTOMERS **INNER JOIN** RESERVATIONS **ON** CUSTOMERS.ID = RESERVATIONS.[Customers_ID]

**WHERE** RESERVATIONS.ClassID = *CID* **AND** RESERVATIONS.[Class Date] = *CDate* **AND** _

RESERVATIONS.[Class Time] = *CT* AND RESERVATIONS.State = 1

Note that values written in italics are variables, whose value depends on the selection made by the instructor.

The full code is shown below.

```
Private Sub Reservations_DblClick(Cancel As Integer)
Dim CID As Integer
Dim CD As Date, CT As Date ' Date and time
Dim MySQL As String
  CID = DLookup("ID", "COURSES", "Name = '" & Me.[Class Name] & "'")
  CD = Me.Data
  CT = Me.[Class Time]
  MySQL = "SELECT CustomerId, CustomerName, CustomerSurname FROM CUSTOMERS INNER JOIN"
  MySQL = MySQL & " RESERVATIONS ON CUSTOMERS.ID = RESERVATIONS.[Customers_ID]"
  MySQL = MySQL & " WHERE RESERVATIONS.ClassID = CID"
  MySQL = MySQL & " AND RESERVATIONS.[Class Date] = " & Format(CD, "\#mm\/dd\/yyyy\#")
  MySQL = MySQL & " AND RESERVATIONS.[Class Time] = #" & CT & "#"
  MySQL = MySQL & " AND RESERVATIONS.State = 1"
  ' We need to open the report before modifying its values. We open it as acHidden, so it is not displayed
  DoCmd.OpenReport "Registrations", acViewDesign, , , acHidden
  ' THIS IS THE IMPORTANT PART OF THIS CODE
  ' The record source of the form and the Control Source of the controls of the form are set
With Reports![Registrations] do ' Alternatively we could have written Reports("Registrations")
          .RecordSource = MySQL
          .TxtID.ControlSource = "CustomerId"
          .TxtName.ControlSource = "CustomerName"
          .TxtSurname.ControlSource = "CustomerSurname"
          ' The description of the report
          .Lbl1.Caption = "Class " & Me.[Class Name] & " Class ID " & CID & " scheduled on " & CD
 End With
   ' Now we need to close the report and to re-open it once again (to make it visible)
   DoCmd.SetWarnings (False)
   DoCmd.CloseReport "Registrations"
   DoCmd.OpenReport "Registrations", acViewPreview
   DoCmd.SetWarnings (True)
End Sub
```

It is interesting to note that, before one can modify the Record Source property of a report, the report must be opened. Since we do not want to display the repot, until it is ready, we opened it in a "Hidden way", using the following instruction:

```
   DoCmd.OpenReport "Registrations", acViewDesign, , , acHidden
```

Also note that, as we changed the Record Source property of the form, we also needed to change the Control Source of its controls. That is:

```
  .TxtID.ControlSource = "CustomerId"
```

```
.TxtName.ControlSource = "CustomerName"

.TxtSurname.ControlSource = "CustomerSurname"
```

## 3. Signaling unjustified absents - Moving along records, and using support tables

Clicking on the Cmd_Absents button (segna assenti), the Absents Form (shown in Figure 4.4) is opened.

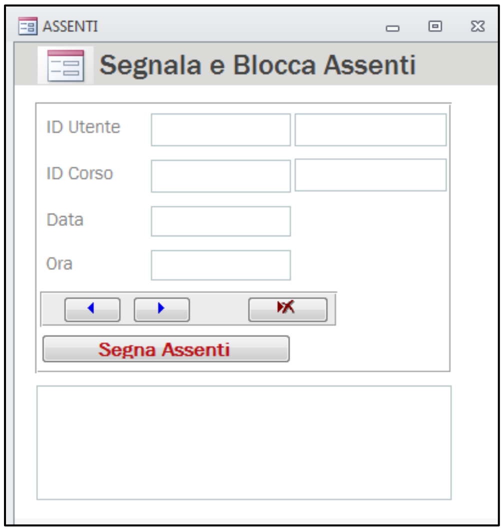


**Fig. 4.4** *The Absents Form*

This form allows instructors to signal unjustified customers that did not attend the class to which they registered to. The peculiarity of this form is that the instructor does not have to insert the absent customers one by one; instead a batch procedure can be followed: it is possible, in fact, to insert at the same time the list of all the customers that missed at least one of the classes held by the instructor (even in held in different days).

Specifically, to add a new record, the instructor must perform the following actions:

- Fill the first text box with the ID of the user that has to be blocked; note that, in this case, specifying the ID is not a problem, since the instructor must fill in this form using the report of the registered users (the same that he used to take attendance of the class and to register absents) where the user Id is also indicated. As soon as a valid ID is inserted, the name of the corresponding customer is automatically displayed in the text box at its immediate right. To this aim, the value of the text box is based on the following DLookUp function:

```
IIf(IsNull([UserID]);"";DLookUp("Surname";"CUSTOMERS";"ID = " & [UserID]))
```

- Fill the second text box with the ID of the course missed by the user. Again, the name of the course appears, automatically, in the adjacent text box.
- Fill the other two text boxes with the date and time of the class.

- Additional records can be added, using the "move to next" navigation button (located at the bottom of the form) and repeating the above mentioned procedure.
- By means of the navigation buttons, it is possible to move forward and backward (along the records of the list), to make modifications and to delete records.

By doing so, the instructor can insert, all at once, the whole list of the customers that must be blocked and/or correction can be made as well. Note that the navigation buttons perform a very simple action, triggered by the following code, used to move to the next and to the previous record (of a table), respectively:

```
DoCmd.GoToRecord , , acNext
DoCmd.GoToRecord , , acPrevious
```

Concerning the "cancel button" (i.e., the red one with the crossed arrow icon), its action is based on the following two lines of code that are needed to select a record (the current one) and to delete it, respectively:

```
DoCmd.RunCommand acCmdSelectRecord ' The current record is selected
DoCmd.RunCommand acCmdDeleteRecord ' The selected record is deleted
```

Lastly, to save data in the data base, the instructor has to click on the "Add Absents" button, which updates the data of the data base. However, in case of input errors, only correct records will be updated and removed from the list shown on the form. All the other ones (i.e., those with wrong values) remain in the list, so that they can be visualized and amended by the instructor.

In order to implement this peculiar functioning, we took advantage of a support table, called BLOCKED, that is not linked to any one of the other tables of the Data Base. This table has records made of five fields (ID, CustomerID, CourseID, Class Date and Class Time) and it is dynamically linked to the Absent Form (i.e., Record Source = Dynaset). So, any modification/amendment made operating on the form is immediately recorded in the BLOCKED support table of the data base. Nonetheless, none of the data stored in the other tables of the data base are affected in any ways, and so data integrity is totally assured. Only when the instructor clicks on the Add Absent button the data are verified, for correctness, and, in case of compliance, a query is triggered to update data of the CUSTOMERS table.

Specifically:

- Two recordsets are used. The first is a clone of the recordset of the form; the second one is opened on the RESERVATIONS table. This makes it possible to compare values inserted by the instructor with the values contained in the RESERVATIONS table (i.e., check if the customers signaled by the instructor registered to the class indicated by the instructor);

- For each record (i.e., customer to be blocked), the inserted date is checked first; it is thus evident that absent customers can be declared only for past classes;
    - A Second check is made to see if the class is one of those assigned to the instructor that is operating on the information system. This condition is evaluated with a Dlookup function that operates on the bridge table INSTRUCTORS-COURSES;
        - If both the date and the CourseID were correct, a last check is made to see if the customers (signaled by the instructor) registered to the class indicated by the instructor. To this aim, the second recordset is used. Specifically, this recordset is opened on a quite complex query that operates on the RESERVATIONS table
- If everything went fine, an update query is written and executed; this query sets to true the blocked field of the signaled customer. Next, the current record is deleted from the form (and from the support table). Conversely, in case of errors (i.e., at least one condition was not satisfied), the update query is not executed, and the current record is not deleted.

At the end of the process, if there are undeleted records in the forms, a message error is shown, and the instructor is prompted back to the list of the signaled customers. The instructor can scroll the list and amend the errors, before saving it again.

The full code is shown below.

```
Private Sub SegnaAssenti_Click()
Dim Rs1 As DAO.Recordset, Rs2 As DAO.Recordset
Dim Db As Database
Dim Condition, Messagge, UpdSQL As String
On Error GoTo Err

DoCmd.GoToRecord , , acNext ' Before saving we move to the next record.

                           '  This avoid errors in case of missing records
Set Db = CurrentDb
Set Rs1 = Me.RecordsetClone ' We clone the recordset of the form
' A second record is opened on the RESERVATIONS table
Set Rs2 = Db.OpenRecordset("RESERVATIONS", dbOpenDynaset)
  DoCmd.SetWarnings (False)
  Rs1.MoveFirst
  Do While Not Rs1.EOF  ' We check all the records inserted in the form
       ' First condition to be checked: absents can be declared only for past classes
       If Rs1.Fields("Class Date") < Date()  Then
      ' Second condition: the class must be assigned to the instructor,
        Condition = "IstructorID = " & LoggedUser
        If Not IsNull(DLookup("CourseID", "[COURSES-INSTRUCTORS]", Condition)) Then
          Condition = "[CustomerID] = " & Rs1.Fields("CustomerID")
```

```
                Condition = Condition & " AND CourseID = " & Rs1.Fields("CourseID")
                Condition = Condition & " AND [Class Date] = "
                Conditon = Condition & Format(Rs1.Fields("Class Date"),"\#mm\/dd\/yyyy\#")
                Condition = Condition & " AND [Class Time] = #" & Rs1.Fields("Class Time") & "#"
                Condition = Condition & " AND State  = 1"
                Rs2.FindFirst Condition ' A record fulfilling the condition is searched in the RESERVATIONS table
                If Not Rs2.NoMatch Then ' If a record was found the corresponding user is blocked
                 ' An Update Query is written and executed
                 UpdSQL = "UPDATE CUSTOMERS SET Blocked = 1 WHERE ID = " & Rs1.Fields("CustmerId")
                 DoCmd.RunSQL (UpdSQL)
                 Rs1.Delete  ' And the record is deleted from the cloned recordset (and so also from the form)
' Also the RESERVATIONS table is updated setting to 1 the value of the absent field
' This is made operating on the second recordset, rather than using an update query
                Rs2.Edit
                 Rs2.Fields("Absent") = 1
                 Rs2.Update
              End If
           End If
        End If
        Rs1.MoveNext     '  Now it goes to the next record
    Loop
    If Rs1.RecordCount >= 1 Then ' If there are some records, something went wrong
         Messagge = "Records still visible in the form were not updated. Please check them again"
         DoCmd.GoToRecord , , acFirst  ' Focus goes back to the first (wrong) record
    Else
       Message = "Updating completed. All records were correctly processed"
    End If
    Rs1.Close
    Rs2.Close
    Set Rs1 = Nothing
    Set Rs2 = Nothing
    Set Db = Nothing
    DoCmd.SetWarnings (True)
    Me.TxtMessate.Value = Message
Err:
    If Err.Number <> 0 Then
          Resume Next
    End If
End Sub
```

Also note that, before executing the above-mentioned code, the support table BLOCKED must be empty. Thus, to be sure of it, a deleting query is executed whenever the form is opened.

The full code is shown below.

```
Private Sub Form_Load()
Dim MySQL As String
LoggedUser = Me.OpenArgs ' As usual we use the openarg to get the ID of the logged user
   MySQL = "DELETE * FROM BLOCKED ' Since there is no a WHERE condition, all records will be deleted
    DoCmd.SetWarnings (False)
    DoCmd.RunSQL (MySQL)
    DoCmd.SetWarnings (True)
    Me.Requery
End Sub
```