

IPISTIAI MORETTO BRESCIA

Tesi di maturità

Anno scolastico 2003/04

Messaggistica Su Pannello Scorrevole

Classe 5[^]BZ

Studente:
Studente:

DILZAN SINGH RAI

Introduzione.....	3
Capitolo 1 - Tecnologie utilizzate	5
I DATABASE:.....	5
MICROSOFT ACCESS	9
Borland Delphi 5 Enterprise	10
TABELLA TIPO IMPEGNI:.....	17
TABELLA IMPEGNI:	17
TABELLA LUOGHI:.....	18
TABELLA DESTINATARI:.....	18
Cap. 3 - Programma di data-entry realizzato con Ms Access	18
Capitolo 4 - Programma di Messaggistica realizzato con Delphi 5.....	24
DATABASE IN ENGLISH :	35
Conclusione.....	36

Introduzione

L'obiettivo finale di questa tesina è di realizzare un programma software che, tramite diverse soluzioni (Pannello Scorrevole, Sms, Email), comunichi tutti gli impegni scolastici al personale della scuola interessata (Collaboratori esterni compresi).

I vantaggi che ne conseguirebbero sono una migliore e sicura gestione degli impegni scolastici i quali a volte vengono dimenticati. Quindi tramite questo strumento tutti impegni verranno ricordati e ripetuti, agevolando la comunicazione all'interno dell'istituto. L'utilizzo di questo prodotto consentirebbe di:

- ❖ Migliorare la gestione delle varie tipologie degli impegni (Consigli d'istituto, Scrutini, Consigli di classe, Conferenze, Assemblea genitori, ecc...);
- ❖ Associare ad ogni impegno la Data, il Giorno, L'ora , il Luogo e i destinatari coinvolti
- ❖ Controllare con precisione tutte le comunicazioni;
- ❖ Avere una certa sicurezza sui dati memorizzati impedendo manomissioni.

Ho scelto questo argomento tra quelli che il professor Azzani ci ha suggerito per l'approfondimento della tesina, perché mi è sembrato utile dal punto di vista scolastico consentendomi di applicare le conoscenze che ho acquisito in questi 5 anni di studio.

Nel corso dello sviluppo del programma si è reso necessario l'apprendimento di varie tecnologie che verranno ampiamente descritte nel corso di questo documento.

La mia tesina è suddivisa in 4 capitoli:

- Nel capitolo 1 verranno descritte le varie tecnologie utilizzate: in particolare ho utilizzato Microsoft Access per costruire sia il DataBase che il programma

di data-Entry, Borland Delphi5 per il programma di messaggistica e infine il linguaggio Uml che mi serve per fare i diagrammi dei casi d'uso;

- Nel capitolo 2 vedremo come è stato strutturato il database che si suddivide in 4 Tabelle;
- Nel capitolo 3 descriveremo il programma di DataEntry realizzato in MS Access che verrà utilizzato per gestire tutti i dati su cui si basa il Programma di Messaggistica;
- Nel capitolo 4 verrà illustrato il programma di messaggistica che si occupa dell'invio degli sms, delle Email e della visualizzazione sul pannello scorrevole degli impegni.

Capitolo 1 - Tecnologie utilizzate

I DATABASE:

Che cosa è un database?

Un database è un insieme organizzato d'informazioni. L'organizzazione è l'elemento fondamentale e richiede una fase di progettazione del db. Si crea un database nel momento in cui si raccolgono e si schedano dati per uno scopo specifico. Nella gestione dei database vengono utilizzati database operativi e database analitici.

I database operativi vengono utilizzati per raccogliere, modificare e mantenere dati su base aggiornata. I tipi di dati raccolti sono dinamici cioè cambiano costantemente e riflettono informazioni sempre aggiornate. Un database analitico raccoglie e traccia dati storici che sono legati ad un preciso momento. I tipi di dati raccolti sono statici cioè non vengono mai modificati. Le informazioni ottenute da un database analitico riflettono un momento particolare in cui i dati sono stati raccolti e non sono aggiornati.

Esistono vari tipi di modelli di database. Il modello più diffuso dei database è il db relazionale, si tratta di db la cui organizzazione segue un modello teorico sviluppato fin dagli anni '70 dai ricercatori Boyce e Codd presso i laboratori Ibm. Si tratta del modello di db più diffuso.

Nella mia tesina ho utilizzato un database di tipo operativo. Questo database mi è servito per fare delle tabelle; utilizzando i comandi sql, il programma effettua delle "query" (interrogazioni) per cercare nel database le varie informazioni (ricerca dei destinatari, degli impegni, del tipo impegno e tabella luoghi). Un database è "un raccoglitore dei dati" strutturato in maniera particolare: ci sono varie tabelle in relazione ed ogni tabella è costituita da un insieme di righe

(record) e colonne (campi). I campi corrispondono ai vari pezzetti di informazione che si vogliono memorizzare (ad esempio una tabella clienti avrà i campi nome, cognome, telefono). I record corrispondono alle varie entità di cui si vogliono memorizzare le informazioni (ad esempio in una tabella clienti ogni record corrisponderà ad un cliente). Ogni database viene studiato e strutturato in base alle esigenze del programma; ad ogni tabella viene assegnato un indice primario (chiamato primary key) allo scopo di evitare inserimenti doppi dei record e di indirizzare velocemente le interrogazioni fatte; tutto ciò significa che un errato studio di database può compromettere seriamente il buon funzionamento del programma che lo usa rendendo necessario la loro riscrittura (sia del database che del programma).

ImpCodice	Tipo Impegno	ImpDescrizione	ImpGiorno	ImpOra	Luogo	ImpDataInizio	ImpDataFineP	Imp
1	Conferenza	popoli oppressi	15/04/04	17:00	Aula Magna	28/04/04	29/04/04	
2	scrutini	Scrutini di amr	25/12/00	18:00	Biblioteca	20/05/04	20/05/04	
3	Collegio Docc	Ordine del Gion	26/05/04	15:30	Aula Magna	24/05/04	26/05/04	
4	Consiglio d'ls	Ordine del Gion	25/05/04	17:30	Aula Consigli	23/05/04	29/05/04	
7	Conferenza	POPOLI Oppre:	01/06/04	18:00	Aula Magna	01/04/04	03/04/04	
8	scrutini	esempio scrutir	03/12/04	13:00	Salainsegnar	01/12/04	03/12/04	
11	Conferenza	POPOLI OPPR	03/06/04	13:00	Aula Consigli	01/06/04	03/06/04	
12	Collegio Docc	Collegio Docent	03/06/04	18:00	Aula Consigli	01/06/04	03/06/04	
*	(Contatore)	0						

Figura 1 vediamo la tabella e i vari campi

Il linguaggio SQL

Il linguaggio SQL (Structured Query language) è un linguaggio strutturato che viene utilizzato quando si ha esigenza di interrogare, gestire e inserire dei dati in un database.

Questo linguaggio consente di aver accesso ai dati contenuti in un database relazionale (Informix, Oracle, Sybase, Microsoft Sql server, Access, ecc.) e descrive le informazioni che chi utilizza il programma intende visualizzare. Il linguaggio SQL ci permette anche di strutturare un database consentendone l'elaborazione. Come nella maggior parte dei linguaggi anche questo presenta alcune varianti a seconda degli standard presenti.

Nella fattispecie in un database relazionale i dati vengono memorizzati in tabelle; ogni riga viene chiamata "Record" e ogni colonna viene chiamata "Campo". In primo luogo dobbiamo dire che ogni campo può contenere al suo interno informazioni di diversi tipo (Stringhe, numeri, date e ore, ecc.); le operazioni più comuni eseguite sulle tabelle sono: SELECT (per selezionare degli elementi), INSERT (per inserire un nuovo record), UPDATE (per aggiornare un record), DELETE (per eliminare un record).

Per esempio se in una tabella chiamata "Anagrafica" vengono riportate le colonne: Nome, Cognome, Indirizzo, data di nascita possiamo, tramite l'istruzione select, prendere in considerazione solo alcune colonne:

```
SELECT Nome, Cognome  
FROM Anagrafica;
```

Dopo aver digitato questo comando otterremo un'altra tabella generata dal programma che contiene solo le colonne Nome e Cognome. Per quanto riguarda

i nomi delle colonne non è possibile utilizzare il carattere “Spazio” al loro interno, In quanto non sopportato dall’SQL. Analizzando le due righe di codice scritto sopra a scopo di illustrativo notiamo che alla fine di ogni istruzione deve essere presente un (;) per segnalare la fine del comando stesso. Esistono anche altre istruzioni sopra citate, per esempio INSERT che ci consente di aggiungere alla nostra tabella un altro record (riga), oppure UPDATE che ci permette di aggiornare il contenuto di un record o una colonna, o DELETE che abilita ad eliminare un record o il suo contenuto.

Per poter mettere in relazione dei valori è necessario avere a disposizione della simbologia che ci permetta di compiere dei confronti. E quindi useremo **“OPERATORI RELAZIONALE”**:

Gli operatori relazionali vengono utilizzati assieme alla clausola WHERE (DOVE) per specificare che solamente alcune righe (dette anche “record” del DBMS) della tabella verranno visualizzate in base al criterio specificato nell’istruzione Where.

=	UGUALE
<>	DIVERSO
<	“MINORE DI....”
>	“MAGGIORE DI....”
<=	“MINORE O UGUALE A ...”
>=	“MAGGIORE O UGUALE A...”

MICROSOFT ACCESS

Microsoft Access è un gestore di DATABASE (DBMS: DATABASE MANAGEMENT SYSTEM) relazionali. Oltre al motore del database (quella parte cioè del programma che gestisce la memorizzazione, ricerca , ecc..dei dati). Access include vari strumenti per la manipolazione dei dati, tra cui un vero e proprio linguaggio di programmazione derivato da Visual Basic (si chiama Visual Basic for Application ed è presente in tutto il pacchetto office).

OGGETTI DI UN DATABASE ACCESS: un database di access è costituito dai seguenti elementi:

TABELLE: Contengono tutti i dati memorizzati, tutti gli oggetti successivi non possono memorizzare alcun dato, ma si limitano a “ Pescare” i dati delle tabelle.

QUERY:si utilizzano per interrogare le tabelle, cioè per selezionare e cambiare i dati nel modo opportuno.

MASCHERE:si utilizzano per presentare i dati a video in modo significativo e comodo da gestire. Ad esempio permettono di vedere eventuali elementi multimediali (come immagini) memorizzate nel db, cosa non possibile direttamente nelle tabelle .

REPORT: si utilizzano per presentare i dati in stampa; come per le maschere consentono di stampare i dati con una grafica impossibile da ottenere nelle tabelle.

MACRO: consentono di creare dei programmi per la gestione dei dati senza bisogno di programmare (senza cioè scrivere direttamente le istruzioni).

VISUALIZZAZIONE DELLE TABELLE:

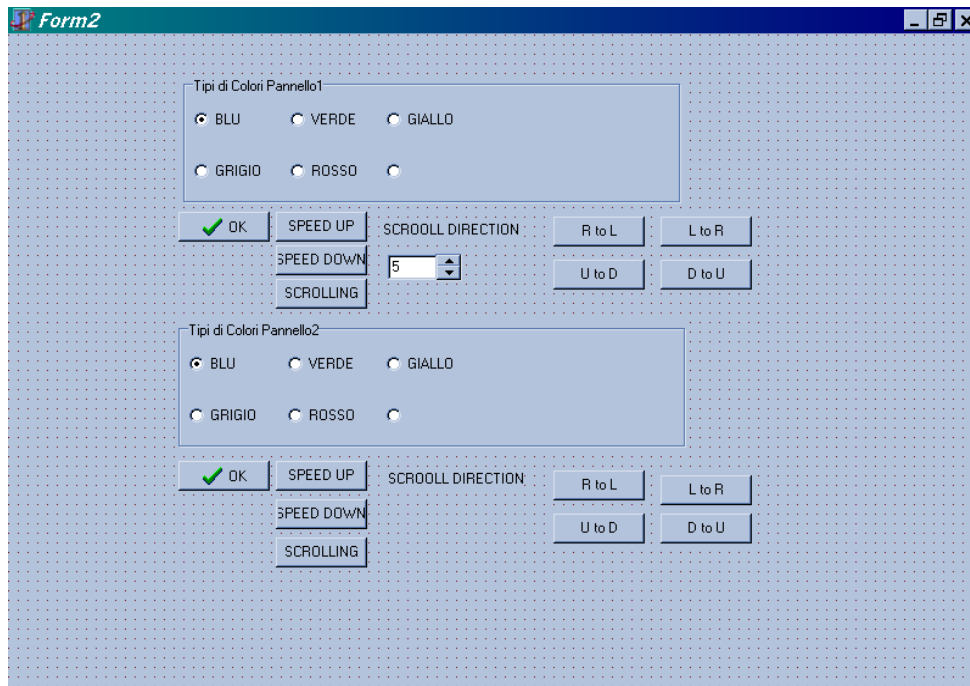
In Access le tabelle possono essere visualizzate in due modi possibili:

VISUALIZZAZIONE FOGLIO DATI: permette di visualizzare e gestire i dati .

I dati sono mostrati in una tabella simile ai fogli di Excel

VISUALIZZAZIONE STRUTTURA: permette di definire i campi costituenti la tabella.

Borland Delphi 5 Enterprise



Borland Delphi 5 è il linguaggio di programmazione che ho utilizzato per gestire impegni scolastici .

Il **Delphi** nasce come evoluzione del "Borland Turbo Pascal". Il **Pascal** è un linguaggio ad alto livello che è stato sviluppato alla fine degli anni sessanta dal professor Niklaus Wirth a Zurigo. Il suo scopo era quello di realizzare un linguaggio comprendente un piccolo numero di concetti fondamentali di programmazione sufficienti ad insegnarla in forma di disciplina logica e sistematica; nello stesso tempo voleva un linguaggio che fosse facile ed efficiente da implementare sulla maggior parte dei calcolatori. Il suo successo nel conseguire questo obiettivo può essere misurato dalla rapida diffusione

dell'uso del PASCAL sia come linguaggio usato nella didattica dei principi di programmazione (si usa tutt'oggi nelle università), sia come linguaggio effettivo per scrivere software di base ed applicativo.

Oggi le varie versioni di Delphi sono disponibili in differenti edizioni:

- ❖ Le "**Standard**" sono, come dice il nome stesso, il livello base di Delphi per la realizzazione di applicazioni semplici che non richiedono l'utilizzo di database e l'accesso ad Internet, perlopiù indirizzate a studenti, hobbisti, principianti.
- ❖ Le "**Developer**" o "**Professional**" sono rivolte a coloro che utilizzano il Delphi per lavoro e che necessitano di supporto per database e internet.
- ❖ Le "**Enterprise**", come dice il nome, sono rivolte alle aziende che necessitano del supporto per il controllo di versione, accesso a database sever, internet, internazionalizzazione del software.
- ❖ Il Delphi é "object oriented" (cioè orientato agli oggetti) quindi la stesura del codice sorgente non è l' unica operazione da fare ma è accompagnata da un' impostazione grafica composta da finestre, bottoni, caselle di testo e molti altri oggetti.
- ❖ Tra i vari componenti possiamo vederne alcuni:

- ❖ **FORM:** è un interfaccia grafica tipica di ogni applicazione basata su sistemi operativi di Windows. Essa ha la funzione di permettere all' utente di interagire con l' applicativo e quindi impartigli degli "ordini" tramite l' uso dei vari oggetti "visuali" che il programmatore include al suo interno, allo scopo di soddisfare le specifiche del programma.



- ❖ **BUTTON:** è uno degli oggetti standard che permette di interagire con il programma. La sua funzione principale è quella di far eseguire determinate operazioni alla sua pressione, che vengono stabilite dal programmatore tramite la scrittura di istruzioni particolari che vengono eseguite in sequenza.



- ❖ **LABEL:** è un' "etichetta" che non può essere modificata direttamente dall'utente che ha uno scopo principalmente indicativo (non può essere usata per fare una descrizione su più righe).



- ❖ **EDIT:** è una casella di testo sviluppata su un' unica riga che permette all'utente di inserire stringhe di testo allo scopo di fornire dei dati al programma. Questo può servire, per esempio, quando l'utente deve effettuare una ricerca o quando deve inserire una stringa richiesta dal programma stesso (per esempio una password).



- ❖ **MEMO:** è un oggetto con le stesse funzioni della Edit ma con la differenza che il testo immesso può essere distribuito su più righe. Per esempio lo si utilizza per l' inserimento di commenti o appunti.



- ❖ **CHECK BOX:** è un oggetto che ci permette di selezionare o meno un' opzione.



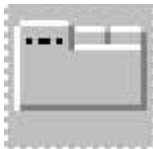
- ❖ **RADIO BUTTON:** è anch' esso un oggetto grafico, ma al contrario della check box ci consente di selezionare solo una delle varie opzioni.



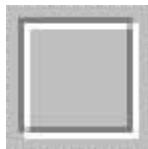
- ❖ **COMBO BOX:** è una casella combinata che ci consente di selezionare una delle voci contenute al suo interno. Questo strumento ci permette di mantenere una visione globale delle possibili decisioni da prendere.



- ❖ **PAGE CONTROL:** è un componente che permette di distribuire i vari controlli (button, edit, label, ...) su più pagine attivabili cliccando sulle relative "linguette".



- ❖ **PANEL:** è un oggetto che ha lo scopo di "abbellire" il nostro programma; ci permette inoltre di inserire al suo interno degli oggetti, dividendoli magari per categoria, in modo da ottenere un' estetica il più intuitiva possibile.



- ❖ **ADO QUERY:** è un oggetto non visuale che permette di eseguire qualsiasi operazione su un database di tipo Access. Per svolgere il suo

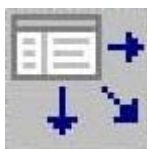
compito utilizza il linguaggio SQL. Le operazioni solitamente più eseguite sono: ricerche (in Sql vengono chiamate select), inserimenti di nuovi dati (insert), modifiche a dati esistenti (update), eliminazione di dati (delete).



- ❖ **ADO CONNECTION:** è un oggetto non visuale e permette all' oggetto precedente di comunicare con il database fisico. Assieme ad Ado Query questo piccolo oggetto rappresenta uno degli oggetti più importanti del Delphi.

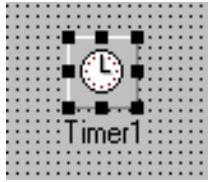


- ❖ **DATA SURCE:** è lo strumento che permette di far leggere le informazioni contenute in un database Msaccess (.mdb) all'intero sistema.



- ❖ **TIMER:** Gli oggetti *TTimer* causano un evento *OnTimer* con cadenza periodica. La durata del periodo è specificata nella proprietà *interval*, espressa in millisecondi. All'interno della funzione associata a tale evento che serve a generare degli eventi dopo un tempo che può essere stabilito

dal programmatore.



LOG FILE: è un oggetto non visuale che permette di registrare in un file di testo (log file) gli eventi più importanti di quel specifico programma.

Con questo riusciamo a risalire ad eventuali problemi o solamente ad avere un resoconto del lavoro svolto.



Microsoft PowerPoint

Microsoft PowerPoint è uno strumento del pacchetto Office utilizzato da noi per lo sviluppo delle presentazioni grafiche di questo progetto.

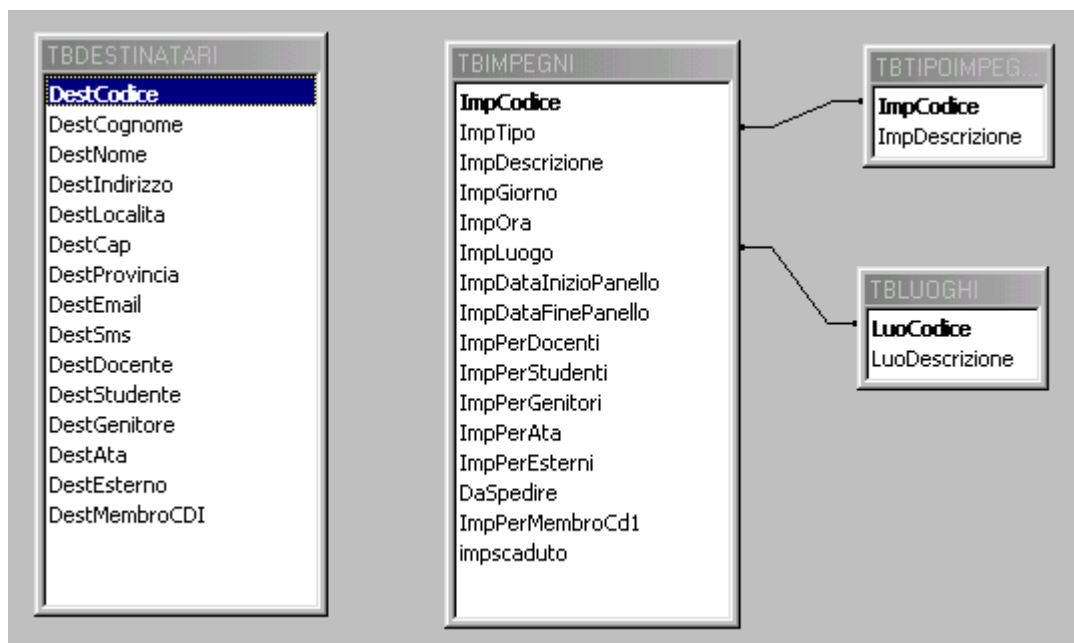
Questa applicazione permette di creare una serie di “diapositive” visualizzabili in una sequenza stabilita e contenenti immagini, testi e schemi utili per l’ esposizione al pubblico.

Lo scorrimento delle diapositive può essere comandato manualmente (un click del mouse o la pressione di un tasto) oppure può essere temporizzato.

La prima ipotesi è utilizzata in quanto è più efficiente nel caso che ci sia un presentatore che espone l’ argomento.

Capitolo 2 - Struttura del database

La struttura del database che ho fatto in Access è la seguente:



1. Tabella Tipo Impegni
2. Tabella Impegni
3. Tabella Luoghi
4. Tabella destinatari

TABELLA TIPO IMPEGNI:

Questa tabella è utilizzata per memorizzare le varie tipologie degli impegni. Per ogni tipologia viene inserito un codice gestito automaticamente da Access e la descrizione (es. Consiglio d’istituto, Consiglio di classe, Conferenze, Colloqui con i genitori, Gli Scrutini ecc..)

TABELLA IMPEGNI:

Questa tabella è utilizzata per memorizzare tutti gli impegni che si desiderano gestire. Ad ogni impegno è associato un codice (es. 1,2,3....), la tipologia (Selezionata dalla tabella precedente), la descrizione(es. Conferenza che può riguardare l’Africa e così via), Il giorno e l’ora, il Luogo (selezionato dalla tabella luoghi (es.Aulamagna , Aulavideo , ecc..), data iniziale e finale di

visualizzazione dell'impegno sul pannello scorrevole(se la data iniziale e finale sono vuote l'impegno non verrà visualizzato sul pannello). I campi (ImpPerDocente, ImpPerStudente, ImpPerGenitore, ImpPerAta, ImpPerMembroConsiglio d'istituto, servono per indicare quali destinatari devono essere avvisati tramite un mezzaggio Sms o una Email.

TABELLA LUOGHI:

Questa tabella è utilizzata per memorizzare i nomi dei luoghi. Per ogni luogo viene inserito un codice gestito automaticamente da Access e la descrizione (es. AulaMagna, AulaVideo, Laboratorio le3 ecc.)

TABELLA DESTINATARI:

Questa tabella è utilizzata per memorizzare tutti i destinatari. Per ogni destinatario viene inserito un codice gestito automaticamente da Access .

Il cognome , il nome , indirizzo, località , cap e provincia del destinatario; questi campi sono tutti obbligatori e no possono essere omessi . I campi (di tipo Fleg) Docente, studente, Genitore, Ata, Esterno, Membro CDI servono per indicare che tipo di ruolo assume il destinatario (es. Singh assume il ruolo di studente). Ho utilizzati i campi Fleg che permettono ad ogni destinatario di assumere anche ruoli diversi (es. Azzanni assume il ruolo di docente e anche il membro d'istituto) oppure (un docente può anche essere il genitore di uno studente dell'istituto).

Cap. 3 - Programma di data-entry realizzato con Ms Access

Dopo aver fatto le tabelle si procede con la realizzazione delle maschere della gestione dei dati. Prima di tutto creo una maschera chiamata FrmAvvio che

permette di accedere a tutte le altre maschere. Queste maschere servono per chi gestisce i dati (es. nella nostra scuola può essere la segretaria).

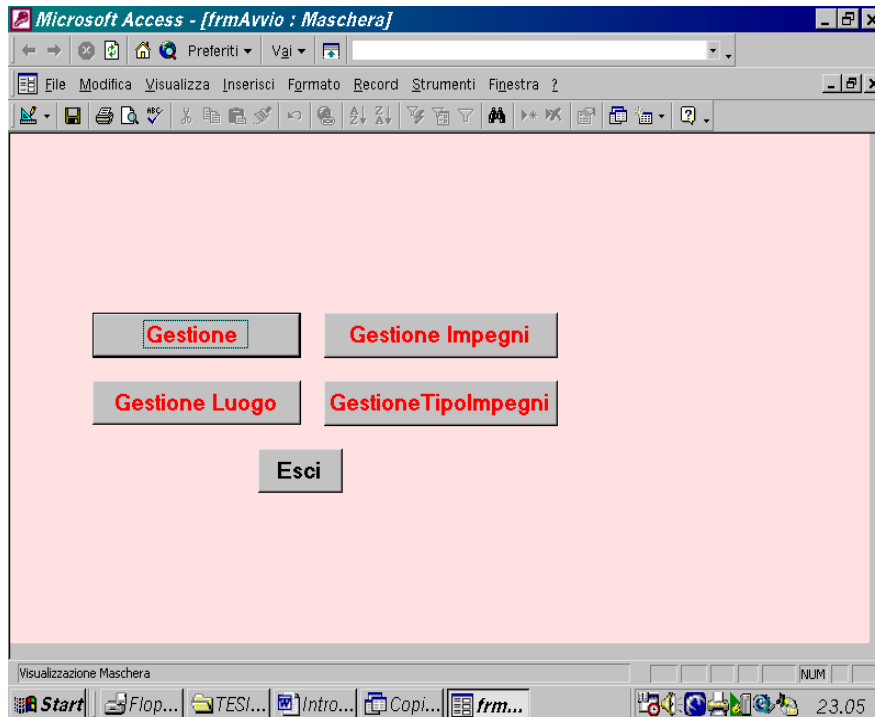


Figura 2 - Maschera di Avvio

Nella figura 1 vediamo che abbiamo la possibilità di scegliere la maschera che si vuole visualizzare (Luoghi, Impegni, TipoImpegni, Destinatari); per chiudere l'intero programma della figura clicco il tasto CHIUDI.

1)Cliccando sul pulsante Gestione luoghi si accede alla maschera FrmLuoghi dalla maschera di avvio.questa maschera permette di modificare un luogo esistente e di inserire i nuovi luoghi cliccando sul pulsante NUOVO o Eliminare i luoghi non più necessari cliccando sul pulsante ELIMINA. Per uscire dalla maschera Frmluoghi clicco sul pulsante ESCI. In fase di inserimento il codice del luogo viene attribuito automaticamente dal sistema.

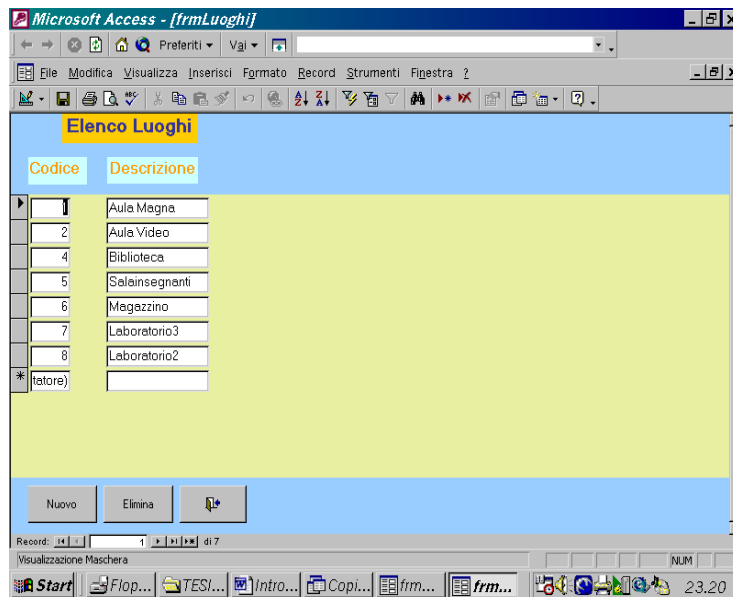


Figura 3 - Maschera Elenco Luoghi

In questa maschera partendo dall'alto sinistra possiamo vedere la colonna del codice e a destra vediamo la descrizione dei luoghi (es. AulaMagna, Biblioteca, Magazzino ecc..).

2) cliccando sul pulsante gestione tipo Impegno si accede alla maschera frmTipoImpegno dalla maschera di Avvio. Anche questa maschera permette di modificare un tipoimpegno esistente, Inserire un nuovo tipo d'impegno clicco sul pulsante NUOVO o eliminare il record che non mi interessa cliccando sul pulsante ELIMINA. Per uscire dalla maschera FrmtipoImpegni clicco sul pulsante ESCI. In fase di inserimento il codice dell'impegno viene attribuito automaticamente dal sistema.

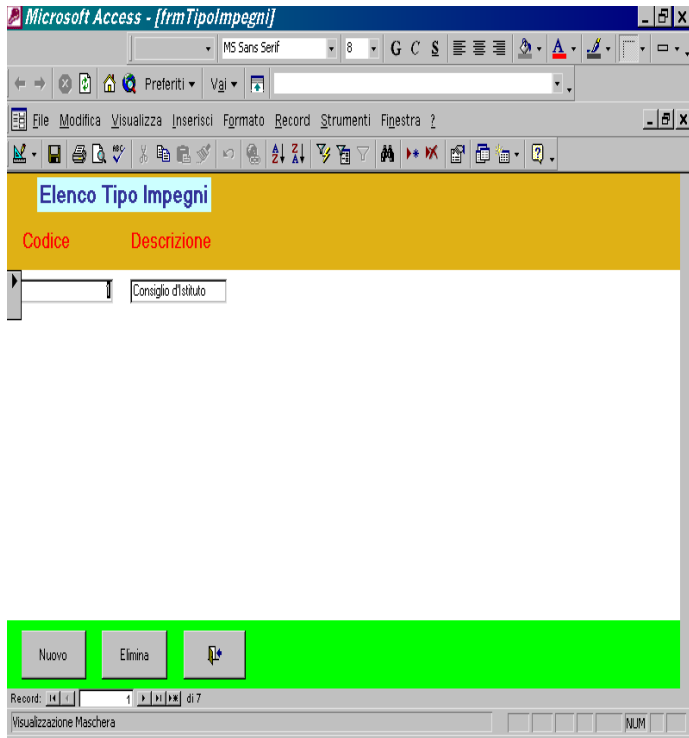
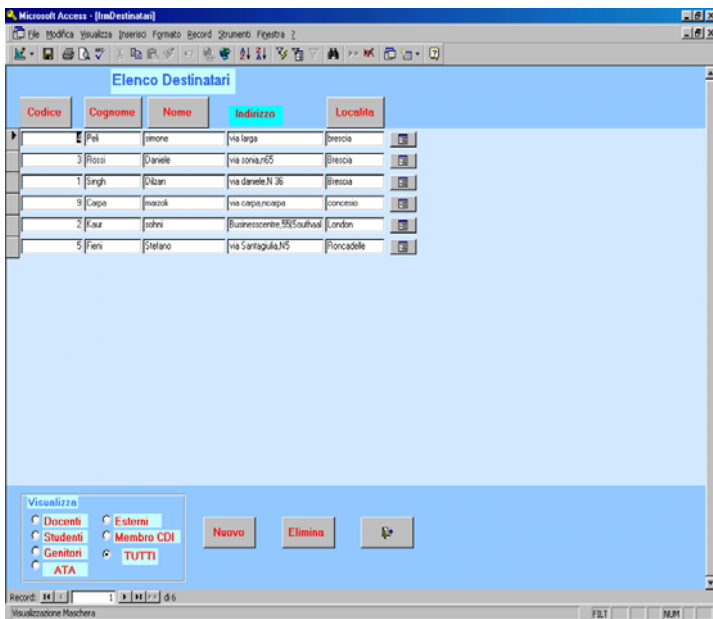


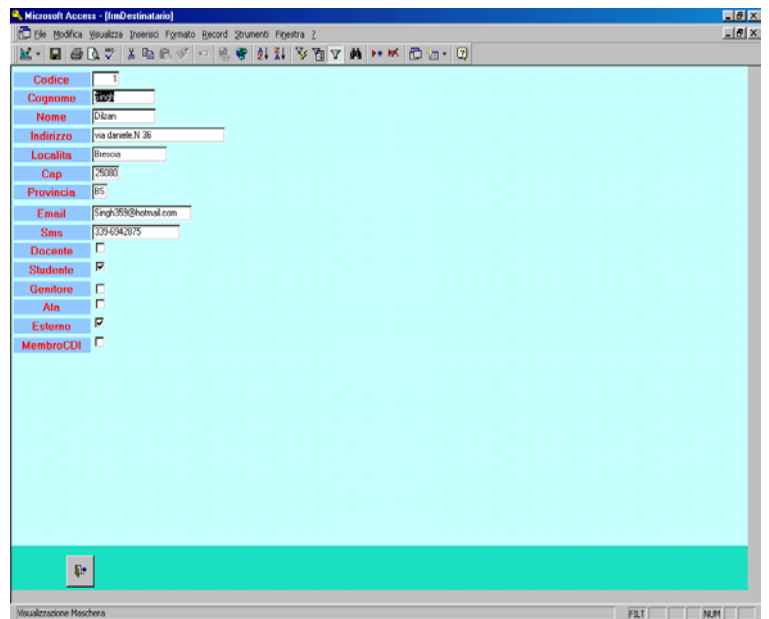
Figura 4 Maschera elenco Tipo Impegni

In questa maschera partendo dall'alto a sinistra vediamo la colonna del codice e in alto a destra vediamo la descrizione del tipo d'impegni (es. consiglio d'istituto, assemblea genitori, consiglio della classe ...).

3) Cliccando sul pulsante Gestione Destinatari si accede alla maschera



Maschera FrmDestinatari



Maschera FrmDestinatario

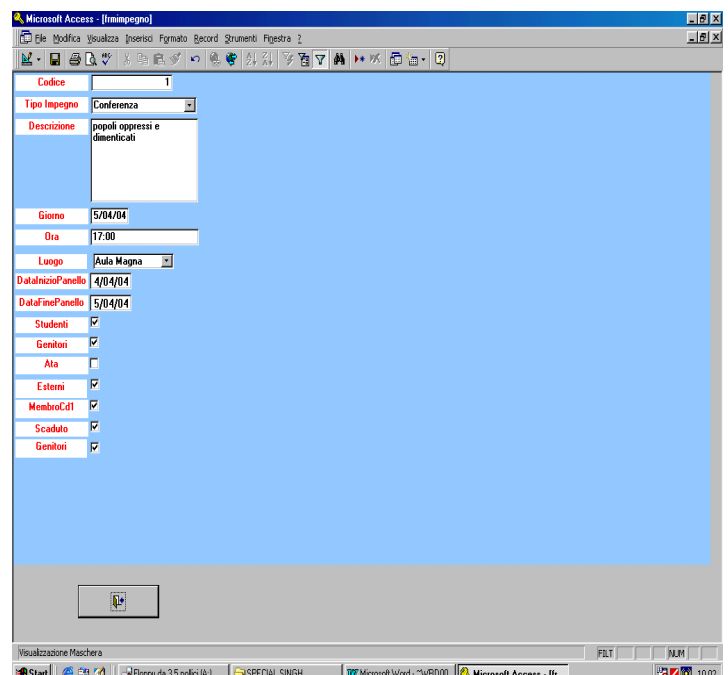
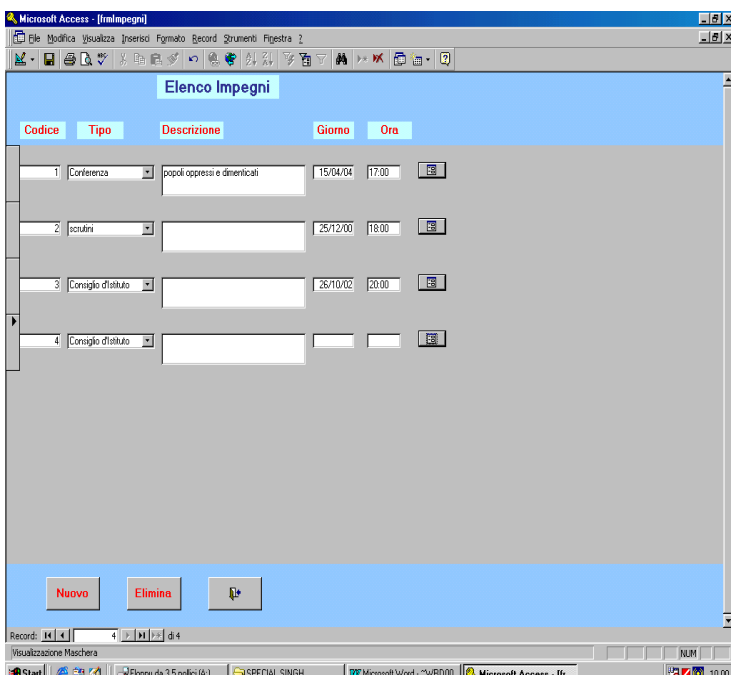
frmdestinatari dalla maschera di avvio. In questa maschera non si vedono tutti i campi completi dei destinatari; per vederli passo alla scheda (maschera chiamata FrmDestinatario) con il pulsantino alla destra di ogni destinatario. L'elenco dei destinatari si può filtrare in base ai diversi ruoli assunti dai destinatari(es. Docente, Studente, Ata, Membro d'istituto, Esterno ecc...); E' anche possibile ordinare attraverso i pulsanti di intestazione delle varie colonne;

Tali pulsanti sono: Codice(es. clicco CODICE il programma mi ordina per i CODICI 1,2,3,4...), Cognome, Nome(es. clicco sul pulsante COGNOME il programma mi ordina per il carattere alfabetico iniziando da A fino alla Z) , infine LOCALITA' (es. clicco sul pulsante LOCALITA' il programma mi ordina per il carattere alfabetico iniziando da A fino alla Z);

Si può eliminare il RECORD che non è più necessario cliccando sul pulsante ELIMINA. Per aggiungere il pulsante " NUOVO " che visualizza una scheda vuota (MASCHERA FRMDESTINATARIO) pronta per l'inserimento.

Per uscire dalla maschera FRMDESTINATARI clicco sul pulsante ESCI. In fase di inserimento il codice del destinatario viene attribuito automaticamente dal sistema.

4) Cliccando sul pulsante gestione Impegni si accede alla maschera FrmImpegni



dalla maschera di avvio. Anche in questa maschera tutti i campi completi degli impegni non si vedono e per vederli passo alla scheda (Maschera chiamata FrmImpegno) con il pulsante alla destra dei vari impegni.

Si può eliminare il RECORD che non è più necessario cliccando sul pulsante ELIMINA. Per aggiungere il pulsante “ NUOVO ” che visualizza una scheda vuota (FrmImpegni) pronta per l’inserimento.

Per uscire dalla maschera FRMDESTINATARI clicco sul pulsante ESCI. In fase di inserimento il codice del destinatario viene attribuito automaticamente dal sistema.

Capitolo 4 - Programma di Messaggistica realizzato con Delphi 5

Dopo aver fatto le tabelle e costruite maschere sono passato al programma Delphi 5 che mi permetterà di vedere sui due pannelli tutti gli impegni scolastici che riguardano i destinatari. Grazie alla collaborazione dei professori sono riuscito a fare un programma adeguato.

Prima di tutto sono andato a creare una nuova Form1 dove andrò mettere alcuni componenti che mi servivano in questo programma. Dalla libreria ADO ho utilizzato il componente ADOCONNECTION che mi permette di stabilir un collegamento con il DATABASE in formato ACCESS. Nel nostro caso è un solo file che contiene le tabelle. Sempre della libreria ADO ho utilizzato il componente ADOTABLE che mi permetterà di visualizzare una sola Tabella (es. nel mio Database per visualizzare tutti gli impegni scelgo la TBIMPEGNI).

Prima di tutto vado definire la connessione che fa riferimento ad ADOCONNECTION. Dopo aver scelto la TBIMPEGNI per poterne vedere i dati devo impostare True la proprietà ACTIVE.

Sempre nella libreria ADO ho utilizzato il componente **ADO QUERY** che è un oggetto non visuale che mi permette di eseguire qualsiasi operazione su un database di tipo Access. Per svolgere il suo compito utilizza il linguaggio SQL. Dalla libreria SYSTEM ho utilizzato il componente TIMER che fa un controllo se Impegno Scolastico scade fra due giorni dalla data che è in quel giorno.

Dopo aver finito di mettere tutti gli oggetti che mi servivano per visualizzare gli impegni ho selezionato l'oggetto LCD dalla libreria FREEWARE che sarebbe il mio pannello. Ho utilizzati due pannelli (Pannello1-Pannello2). Il primo pannello mostra la data e gli tipi degli impegni(es. 8/06/2004 Consiglio di classe...), l'altro pannello mostra il Luogo, l'ora e la descrizione del impegno(es. Biblioteca 16:00).

Dopo aver fatto varie prove scegliendone la dimensione esatta dei due pannelli e i colori adeguati ho scritto questo programma:

```
unit Unit1;

interface

uses

  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  LCDScreen, Grids, DBGrids, Db, ADODB, StdCtrls, ExtCtrls, DBCtrls,
  ComCtrls;

type

  TForm1 = class(TForm)
    Pannello1: TLCDScreen;
    Pannello2: TLCDScreen;
    ADOConnection1: TADOConnection;
    ADOTable1: TADOTable;
    ADOQuery1: TADOQuery;
    Timer1: TTimer;
    Pannello3: TLCDScreen;
    SB1: TStatusBar;
    procedure Pannello2Click(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure Timer1Timer(Sender: TObject);
  private
    { Private declarations }
  end;
end;
```

```
public
  { Public declarations }
  PosizioneRecord : Integer;
  Function RicercaImpegno(cod : integer):string;
  Function RicercaLuogo(cod : integer):string;
  procedure VisualizzaMessaggio;
end;
```

```
Const Ritardo = 30000;
```

```
var
  Form1: TForm1;
```

```
implementation
```

```
{ $R *.DFM }
```

```
procedure TForm1.VisualizzaMessaggio;
```

```
  var Impegno, Imp1,st : string;
      x,y,k,m : Integer;
      data1, data2 : TDateTime ;
```

```
begin
```

```
  Data1:=ADOTable1.fieldByName('ImpDataInizioPannello').AsDateTime;
  Data2:=ADOTable1.fieldByName('ImpDataFinePannello').AsDateTime;
  If ( Date >= data1) and (date <= data2) then
```

```
begin
```

```

sb1.Panels[2].Text := IntToStr(PosizioneRecord);
Pannello1.Lines.Clear;
Pannello1.Lines.Add('
'+DateToStr(ADOTable1.FieldByName('ImpGiorno').AsDateTime)+' '+
UpperCase(RicercaImpegno(ADOTable1.FieldByName('ImpTipo').AsInteger)))
;
Pannello3.Lines.Clear;
Pannello3.Lines.Add('ORE: '+AdoTable1.FieldByName('ImpOra').AsString+'
'+
UpperCase(RicercaLuogo(ADOTable1.FieldByName('ImpLuogo').AsInteger)));

Pannello2.Lines.Clear;
Pannello2.Lines.Add("");
Pannello2.Lines.Add("");
Pannello2.Lines.Add("");
Pannello2.Lines.Add("");
Pannello2.Lines.Add("");
Pannello2.Lines.Add("");
Pannello2.Lines.Add("");
Pannello2.Lines.Add("");
Pannello2.Lines.Add("");
Pannello2.Lines.Add("");
Pannello2.Lines.Add("");
Pannello2.Lines.Add("");
Pannello2.Lines.Add("");
Pannello2.Lines.Add("");
(* st:='ORE: '+ADOTable1.FieldByName('ImpOra').AsString;
Pannello2.Lines[1]:= st;

```

```
Pannello2.Lines[2]:=
UpperCase(RicercaLuogo(ADOTable1.FieldByName('ImpLuogo').AsInteger));
*)
```

```
Impegno := ADOTable1.FieldByName('ImpDescrizione').AsString;
```

```
m := 4;
```

```
Repeat
```

```
  Imp1 := Copy(Impegno,1,31);
```

```
  // ricerca posizione dei due punti :
```

```
  x := Pos(':',Imp1);
```

```
  // ricerca posizione ultimo spazio nella stringa
```

```
  y :=0;
```

```
  For k :=1 To Length(Imp1) do
```

```
    Begin
```

```
      If Imp1[k] = ' ' Then y := k;
```

```
    End;
```

```
If x <> 0 // ho trovato il carattere ":"
```

```
  Then
```

```
    Begin
```

```
      Pannello2.Lines[m]:= Copy(Imp1,1,x-1);
```

```
      Delete(Impegno,1,x);
```

```
      Inc(m);
```

```
    End
```

```
  Else
```

```
    Begin
```

```
      If k = 0 Then ShowMessage('NON TROVO SPAZI-> '+Imp1);
```

```

    If Length(Impegno) > 31 Then
        Begin
            Pannello2.Lines[m]:= Copy(Imp1,1,y);
            Delete(Impegno,1,y);
            End
        Else
            Begin
                Pannello2.Lines[m]:= Impegno;
                Impegno := "";
                End;

            Inc(m);
        End;
    Until (Length(Impegno) = 0) Or (m = 10);
    Timer1.Interval := Ritardo;
end
else
    Begin
        Pannello2.Lines.Clear;
        Pannello1.Lines.Clear;
        Timer1.Interval := 100;
    End;
end;

```

```

procedure TForm1.Pannello2Click(Sender: TObject);
begin
end;

```

```

// dato il codice numerico ricava la descrizione del
// tipo dell'impegno
Function TForm1.RicercaImpegno(cod : integer):string;
  var st : string;
Begin
  st := 'SELECT * From TBTIPOIMPEGNO '+
        'WHERE IMPCODICE='+IntToStr(cod);

  ADOQuery1.Close;
  ADOQuery1.SQL.Clear;
  ADOQuery1.SQL.Add(st);
  ADOQuery1.Open;
  If ADOQuery1.RecordCount <> 1 Then ShowMessage('ERRORE')
  Else result := ADOQuery1.FieldByName('IMPDESCRIZIONE').AsString;

End;

//
Function TForm1.RicercaLuogo(cod : integer):string;
  var st : string;
Begin
  st := 'SELECT * From TBLUOGHI '+
        'WHERE LUOCODICE='+IntToStr(cod);

  ADOQuery1.Close;
  ADOQuery1.SQL.Clear;
  ADOQuery1.SQL.Add(st);

```

```
ADOQuery1.Open;
If ADOQuery1.RecordCount <> 1 Then ShowMessage('ERRORE')
  Else result := ADOQuery1.FieldByName('LUODESCRIZIONE').AsString;

End;
```

```
procedure TForm1.FormShow(Sender: TObject);
begin
  Timer1.Interval := Ritardo;
  Pannello1.PixelOff := clGreen;
  Pannello1.PixelOn := clLime;
  Pannello1.Color := clBlack;
  AdoTable1.Open;
  PosizioneRecord := 1; // parto dal 1° record nel DataBase
  Pannello1.ScrollActive := True;
  Pannello2.ScrollActive := True;
  sb1.Panels[1].Text := 'RECORD n. '+IntToStr(ADOTable1.RecordCount);
  VisualizzaMessaggio; // esce il 1° messaggio
  Timer1.Enabled:= True;
end;
```

```
procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  AdoTable1.Close;
end;
```

```
procedure TForm1.Timer1Timer(Sender: TObject);
begin
```

```

ADOTable1.Next; // passo al successivo record
PosizioneRecord := PosizioneRecord +1;
If PosizioneRecord > ADOTable1.RecordCount
Then
  Begin
    ADOTable1.First;
    PosizioneRecord:=1;
  End
Else VisualizzaMessaggio; // esce il messaggio
end;

end.

```

```

(* procedure TForm1.ComboBox3Change(Sender: TObject);
begin
  case ComboBox3.ItemIndex of
    0 : begin {Blue}
      LCDScreen1.PixelOff := clNavy;
      LCDScreen1.PixelOn := clAqua;
      LCDScreen1.Color := clBlack;
      end;
    1 : begin {Gray}
      LCDScreen1.PixelOff := $00AAAAAA;
      LCDScreen1.PixelOn := clBlack;

```

```

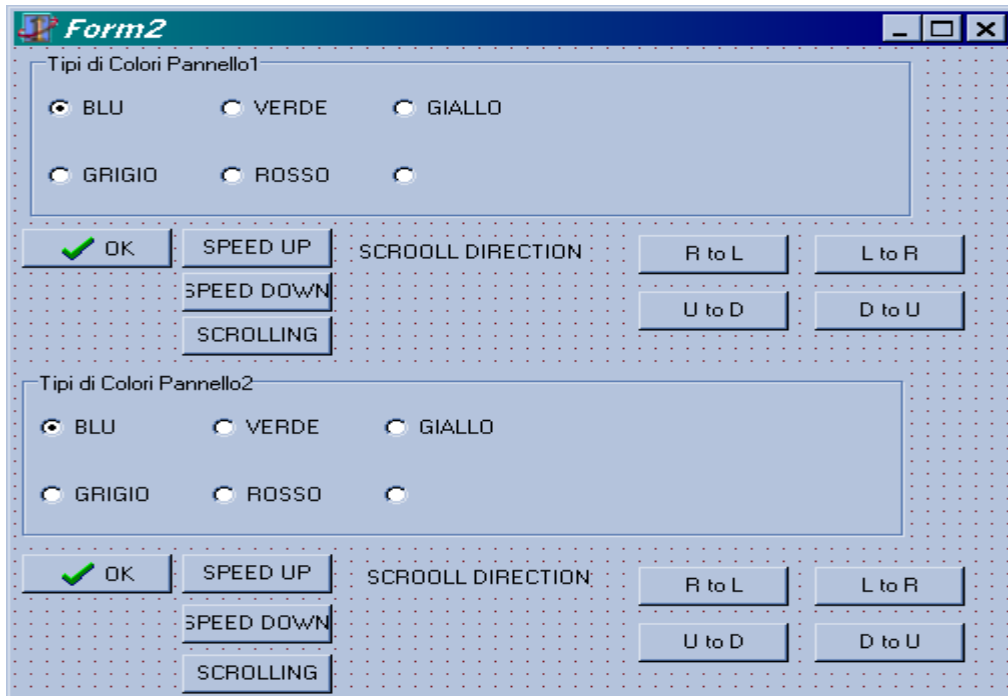
    LCDScreen1.Color := clSilver
    end;
2 : begin {Green}
    LCDScreen1.PixelOff := clGreen;
    LCDScreen1.PixelOn := clLime;
    LCDScreen1.Color := clBlack;
    end;
3 : begin {Red}
    LCDScreen1.PixelOff := clMaroon;
    LCDScreen1.PixelOn := clRed;
    LCDScreen1.Color := clBlack;
    end;
4 : begin {Yellow}
    LCDScreen1.PixelOff := clOlive;
    LCDScreen1.PixelOn := clYellow;
    LCDScreen1.Color := clBlack;
    end;
end;

end; *)

```

Dichiaro le variabili che sono x, y, k, m di tipo integer, all'inizio di questo programma vedremo che il Timer fa un controllo se Impegno scade fra due giorni. Se scade fra due giorni il programma comincia a far vedere Des.Impegno altrimenti se non scade comincia a scartare fino non trova Impegno che scade fra due giorni. Dopo che il programma funzionasse correttamente ho creato la Form2 che mi darà la possibilità di cambiare i colori(Blu,Rosso...), di aumentare o diminuire la velocità, far scorrere dalla sinistra alla destra , dalla

destra alla sinistra ,dall'alto verso il basso e dal basso verso l'alto ,aumentare o diminuire il tempo di permanenza dell'impegno. Questo sarà possibile quando il programma è già in esecuzione.



Alla fine vedremo il pannello scorrevole in esecuzione in questo modo:



DATABASE IN ENGLISH :

Database is a large store of information, set out in such a way that it is easy to keep up-to-date, and to find the information you want at any time. The information covers a particular topic and the items of information are grouped together. In most databases, all the information in a record is known as a field. The records in a database are usually arranged in some order. For example, alphabetical.

Databases are stored on magnetic disk, and a database program gives you access to the information on them. Most database programs allow you to create new records, delete ones you no longer need, alter the fields in records, sort the records in order, and find the records you want. Some have advanced features to combine records, count records, do calculations and create an index for quick access to the information.

Databases are used very widely in businesses, government departments, hospitals, colleges, banks, insurance companies and an increasing number of schools. They have replaced filing systems which used paper records, with index cards to tell you where to look. These filing systems were slow and awkward to use, and easy to get out of order.

Conclusione

Spero di essere stato chiaro nel spiegare il funzionamento e l'utilità di questo programma. Purtroppo per i motivi economici della scuola non ho potuto realizzare concretamente questo programma. Spero con tutto il cuore che questo progetto sia accettato dalla scuola come un strumento adeguato per ricordare vari impegni scolastici al personale della scuola e sia realizzato al più possibile da uno studente l'anno prossimo.

Alla fine voglio ringraziare tutti i miei professori che mi hanno aiutato a portare in termine questo progetto.