Universitat de Girona
Escola Politècnica Superior

Projecte/Treball Final de Carrera

Estudi: Eng. Tècn. Informàtica de Gestió. Pla 2001

Títol: Generador d'Horaris d'un centre docent

Document: Memòria

Alumne: Oriol Villaret Lloret

Director/Tutor: Mateu Villaret Ausellé
Departament: Informàtica i Matemàtica Aplicada
Àrea: Llenguatges i Sistemes d'Informació (LSI)

Convocatòria (mes/any): Juny/2006

Índex

1. Introducció

1.1. El problema

És coneguda la problemàtica que tenen els centres docents a l'hora de crear els horaris escolars: normalment els ocupa mig estiu, utilitzen mètodes rudimentaris (anar provant de col·locar cartolines sobre un suro fins a trobar una solució) i es basen en horaris d'anys anteriors. El resultat sol ser una única solució que normalment no permet cap modificació i a més no sol ser aprovada per tothom, però és l'única a la que s'ha pogut arribar satisfactòriament.

Amb el temps, han sorgit alguns programes que resolen aquest problema i permeten deixar de banda les cartolines, suros i xinxetes. Aquests programes comporten un gran costs de càlcul, ja que intenten resoldre un problema np-complet amb moltes variables i restriccions a tenir en compte i poden tardar entre 2 i 7 dies en trobar una solució, així que el problema segueix sent evident, ja que mai s'aconsegueix la solució que es busca a la primera ni a la segona, així que les escoles segueixen tardant entre dos setmanes i un mes a trobar una solució més o menys acceptable. El principal inconvenient, però, és que aquests programes estan fets de forma genèrica per a tots els tipus de centres decents i generalitzant la forma de treballar de cadascun d'ells. Cada centre docent té la seva manera d'organitzar les classes, assignar professors, limitar horaris, etc. Presuposar una forma de realitzar totes aquestes tasques el més genèric possible, acostuma a ser un inconvenient per tothom.

1.2. L'objectiu

El meu projecte va orientat a aconseguir una aplicació que aconsegueixi resoldre els trencaclosques dels centres docents d'una manera ràpida, versàtil concentrant-me en les necessitats d'un possible client potencial, i que aconsegueixi moltes possibles solucions per aquest problema. Per fer-ho, haurem de tenir en compte tot un conjunt de restriccions:

- Les hores lectives i no lectives dels grups
- Les hores disponibles dels professors
- Cada grup podrà cursar com a molt una assignatura en un moment concret
- Cada professor podrà impartir com a molt una assignatura en un moment concret i en un grup concret.
- Un grup no podrà rebre classe d'una mateixa assignatura més d'un cop al mateix dia.
- Un professor pot, o no, impartir classe a tots els grups d'un mateix curs, a grups d'altres cursos o cursos sencers d'una o vàries assignatures.
- Hi pot haver hores prefixades on s'indicarà el professor que imparteix una assignatura concreta a un grup en un dia i hora especificats

1.3. Com?

Utilitzaré el Visual Bàsic per aconseguir, omplir i mostrar el contingut d'una base de dades feta amb Microsoft Access. Per generar el resultat, utilitzaré el llenguatge de programació lògica Prolog, que al ser un llenguatge declaratiu i utilitzant una llibreria de restriccions que permet fer modificacions d'una forma molt versàtil, aconseguiré un entorn de programació ideal per atacar aquest tipus de problema. Amb una interacció del Visual Bàsic amb el Prolog podré mostrar d'una manera elegant i fàcil d'entendre el resultat obtingut.

2. Alternatives de disseny

Podem agrupar les alternatives de disseny en 3 grans blocs: Metodologia, eines i modelització.

2.1. Metodologia (Backtraquing Vs Constraints)

El backtracking consisteix a resoldre problemes aplicant un mètode sistemàtic de tempteig. El procediment bàsic és descompondre el procés en feines parcials. Per cadascuna d'elles provem un camí cap a una possible solució considerant-lo com la solució correcte. Si es descobreix que per aquest camí no s'arriba a la solució desitjada, es torna a enrera i s'assigna un altre camí com a solució correcte de forma sistemàtica fins a aconseguir la solució desitjada.

Els problemes de resolució de constraints consisteixen en donar valors a un conjunt de variables de domini finit a partir d'unes restriccions. S'entenen com a variables de domini finit aquelles a les que se'ls pot assignar un conjunt finit de possibles valors (domini). Les restriccions sobre les que es treballa delimiten els valors de les variables entre elles.

El Constraint Programing (CP) consisteix en definir unes restriccions sobre les variables que delimiten el problema, en programació lògica sense que unifiquin, si ho fessin deixarien de ser variables de domini finit i passarien a ser constants (no ens interessa).

Un exemple de problema de resolució de constraints i de backtracking molt conegut és el de les N-reines. El joc és molt simple, consisteix en col·locar N reines en un tauler de NxN posicions de forma que no es matin entre elles. Dues reines es maten entre elles si estan a la mateixa fila, columna o diagonal.

Per resoldre el problema, primer n'hem d'escollir una representació:

- Pel backtracking, tindrem:

  i: Enter
  Encertat: Boolea
  Fila: TAULA [1..8] DE Boolea
  Columna:TAULA [1..8] DE Enter
  Diagonal: TAULA [2..16] DE Boolea
  Contradiagonal: TAULA [-7..7] DE Boolea

- Per la resolució de constraints: Podem utilitzar una llista de N posicions on cada element de la llista és una variable de domini finit de 1 fins a N. Cada posició de la llista representarà la columna en la que es troba la reina, i el valor de la variable que conté representarà la fila en la que es troba. Així que una de les restriccions ja està solucionada, ja que mai podrem posar dues reines a la mateixa columna.

En algunes assignatures hem solucionat aquest problema amb backtracking, i el codi és prou conegut per tothom així com el seu resultat. Podem aconseguir una resposta satisfactòria per el problema si el nombre de reines és inferior o igual a vuit, però per més reines, el temps de resposta és totalment inacceptable. Amb resolució de constraints, però no tenim coneixença ni del codi ni del temps de resposta.

Com hem dit abans, amb la representació escollida mai podrem posar dues reines a la mateixa columna, faltarà doncs controlar que no coincideixin les files ni les diagonals. Per les files utilitzarem el predicat **all_different/1** de Prolog. Aquest predicat forçarà que totes les variables que se li passen per paràmetre dins d'una llista siguin diferents entre elles, d'aquesta manera, mai coincidiran dues reines a la mateixa fila.

Finalment, per controlar que a les diagonals de les reines no n'hi hagi cap altre, crearem un nou predicat que donada una llista de N posicions, agafarà el cap de la llista i controlarà que les posicions de la llista que el precedeixen no es trobin en les seves diagonals, i un cop comprovat això, farà la crida recursiva per la resta de la llista. Per tant també haurem de definir la funció que donat un element, controli que en les seves diagonals no hi hagi cap dels elements de la llista.

El codi en Prolog quedaria de la següent forma;

```
dames(N,L):-
      length(L,N),
      domain(L,1,N),
      all_different(L),
      noesmengen(L,1),
      labeling([],L).

noesmengen([],_).
noesmengen([X|XS],N):-
      M is N+1,
      diagonal ((X,N),(XS,M)),
      noesmengen(XS,M).

diagonal (_,([],_)).
diagonal ((X,N),([Y|Ys],M)):-
      A #= X-Y,
      B #= Y-X,
      C is N-M,
      A #\= C,
      B #\= C,
      M2 is M+1,
      diagonal ((X,N),(Ys,M2)).
```

On "noesmengen/2" controla que les dames no es mengin en les seves diagonals i "diagonal/2" controla només la diagonal d'una dama concreta amb una llista de dames. Es pot veure que el predicat "noesmengen/2" cridarà tantes vegades "diagonal/2" com dames tingui dins la llista.

A partir d'aquests predicats, podem fer les crides següents al Prolog que retornarà un resultat gairebé instantani;

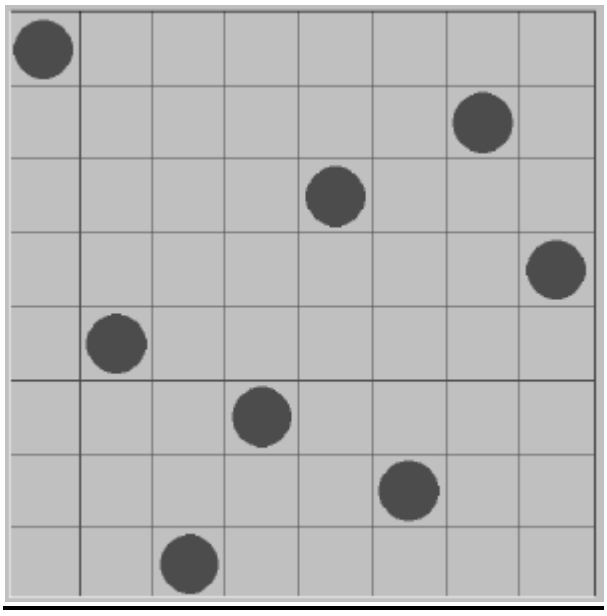
```
| ?- dames(4,L).
L = [2,4,1,3] ?

| ?- dames(8,L).
L = [1,5,8,6,3,7,2,4] ?

| ?- dames(12,L).
L = [1,3,5,8,10,12,6,11,2,7,9,4] ?

| ?- dames(16,L).
L = [1,3,5,2,13,9,14,12,15,6,16,7,4,11,8,10]?

| ?- dames(20,L).
L = [1,3,5,2,4,13,15,12,18,20,17,9,16,19,8,10,7,14,
6,11] ?
```



Podem veure doncs la gran diferència que hi ha entre la solució de les N-Reines en backtracking i en resolució de Constraints. Mentres que les tècniques de backtracking aconsegueixen una solució per a com a molt 8 reines, la resolució de constraints aconsegueix una solució per al triple de reines amb un temps molt raonable.

2.2. Eines (SICStus Prolog Vs GNUProlog)

Un cop vist que la resolució de constraints és més eficient per a solucionar aquest tipus de problemes, s'ha de decidir amb quin tipus de tecnologia treballarem. Tant SICStus Prolog com GNUProlog admeten resolució de constraints, hauria estat interessant utilitzar GNUProlog al tractar-se de programari lliure, però la interacció amb altres programes és molt limitada i això fa que SICStus Prolog sigui el millor candidat, ja que admet interaccions amb c, Java, Visual Basic, Visual .Net i Tcl/Tk, mentres que GNUProlog, només les admet amb c i crear una interfície d'usuari hauria estat més complicat.

2.3. Modelització (all_differents Vs disjoint1)

Un cop escollides les eines i la metodologia, a l'hora d'implementar un model de dades van sortir dues possibilitats molt similars però amb unes grans diferències tant de rendiment com de resultat final.

Totes dues, eren representades amb una llista de tuples (Grup, LlistaAssignatures), on la llista d'assignatures era una llista de tuples (Assignatura, Professor, Hora) i representaven totes les sessions de les assignatures que s'imparteixen a aquest grup.

La diferència entre les dues representacions era la de contemplar la durada de les assignatures o no. En un cas, totes les assignatures tenien la mateixa durada d'una hora. En l'altre, aquesta durada depenia de l'assignatura que estàvem tractant. Certament, la majoria de les assignatures tenen una durada d'una hora, però hi ha alguna excepció com Educació Física, que sol ser de dues hores degut al temps que es perd als vestidors i les dutxes.

- Amb la primera opció podríem utilitzar el predicat **all_differents/1** de Prolog, que donada una llista d'hores, força que totes siguin diferents. Per exemple: ens trobem en el cas que tenim tres assignatures de durades diferents que són "Catala1" de durada **una** hora, "Casetella1" de durada **una** hora i "Educació Física1" de durada **dues** hores.
  Amb el predicat **all_different**/1 de Prolog, forçarà que tots els inicis de les assignatures siguin diferents, però no tindrà en compte les hores finals, deixant una estructura similar a aquesta:

Hora1	Català 1
Hora2	Educació Física 1
Hora3	Castellà 1
Hora4	

  Al no tenir en compte l'allargada de l'assignatura, es trobarà que la classe que es cursa immediatament després d'una assignatura de més d'una hora, se solapa amb aquesta. El problema es podria solucionar introduint una assignatura diferent per a cada hora de l'assignatura de més d'una hora, i forçant d'alguna manera que aquestes s'imparteixin de forma consecutiva, però aquesta seria una solució poc elegant.

- Amb el predicat **all_differetn/1**, però, no n'hi ha prou per la segona opció, ja que forçant que siguin diferents no aconsegueix evitar els solapaments per a les assignatures de més d'una hora. La segona alternativa era la d'utilitzar el predicat **disjoint1**/1 de Prolog. Aquest predicat requereix no només que les hores que volem restringir siguin diferents, sinó que te en compte també la seva llargada. D'aquesta manera no assignarà l'inici d'una assignatura entre l'espai d'inici i final d'una altra, evitant així els solapaments. Aquesta ha estat l'opció escollida.

Hora1	Català 1
Hora2	Educació Física 1
Hora3	
Hora4	Castellà 1

3. Entrevistes

Per a orientar-me i resoldre alguns dubtes del projecte, vaig anar a fer una entrevista amb en Manel Teodoro, cap d'estudis d'escoles Freta a Calella. En aquesta entrevista vaig poder veure com funcionava la seva aplicació de confecció d'horaris i com ho feien abans de tenir-la.

Em va explicar que anteriorment, els horaris eren un autèntic trencaclosques gairebé impossible de solucionar. Utilitzant un tauler de suro i unes cartolines amb els noms de les assignatures i dels professors, anaven provant possibles distribucions d'aquestes cartolines per a tots els horaris de tots els grups. Aconseguir una única solució requeria que durant mig estiu dues persones a jornada completa estiguessin provant combinacions. Aquesta problemàtica es va solucionar quan van optar per informatitzar la gestió del seu horari escolar. Ara tenen una aplicació que els gestiona tant les aules com els horaris dels grups i els professors. Aquesta aplicació, però tarda dos o tres dies per aconseguir una solució vàlida. El problema és evident, ja que si la solució obtinguda no és del tot acceptable, per buscar-ne un altre tornarà a tardar el mateix temps.

Després de veure com funcionava el seu gestor d'horaris, vaig interessar-me per la forma de funcionar de l'escola.

- Vaig preguntar-li si algun cop s'havien trobat en el cas que més d'un professor pugui impartir una assignatura a un grup concret. Em va contestar que normalment quan un professor s'incorporava a la plantilla, ja se li assignava un conjunt d'assignatures a impartir a uns grups concrets i cap altra professor podrà impartir aquestes assignatures a aquests grups.
- Per altra banda, també li vaig preguntar per la distribució de les sessions de les assignatures, més concretament, si aquestes havien d'estar separades un mínim d'hores per tornar-se a impartir. Em va contestar, que totes les sessions de cada assignatura s'han de cursar en dies diferents, però no necessàriament a la mateixa hora.
- També em va interessar el tema d'"hores de centre". Aquestes hores s'assignen als professors amb la finalitat de cobrir possibles substitucions d'alguna absència d'algun altre professor com perquè el professor tingui temps per preparar les classes que ha d'impartir en la resta del seu horari. Acostuma a haver-hi entre dos i quatre professors que tenen hores de centre al mateix moment i cadascun d'ells té unes sis hores de centre a la setmana, depenent del nombre d'hores de classe que tingui.

4. Planificació

La idea d'aquest projecte, va sorgir a finals del curs 2004-2005, quan vaig anar a parlar amb en Mateu Villaret per demanar-li que em fes tutor del projecte aquest curs. Portava tres possibles propostes de projecte que li vaig presentar, una d'elles era la confecció d'una aplicació per gestionar els punts dels clients d'una botiga amb els que podrien aconseguir regals. Una altra proposta era la de crear un joc d'escacs que es basés en tècniques d'intel·ligència artificial, i la tercera i última proposta és la de confeccionar una aplicació que generés horaris per un centre docent.

Després de discutir les opcions, vam arribar a la conclusió que la millor d'elles era la última, i em va dir que la millor manera d'atacar aquests problemes era amb programació lògica i resolució de constraints, per tant havia d'aprendre no només un nou llenguatge de programació, sinó que havia d'aprendre un nou paradigma de programació molt diferent al que estava acostumat. Per això vaig decidir matricular-me de l'assignatura de Llenguatges de Programació.

El problema era evident, d'aquesta forma, no podria començar el projecte fins a mitjans del curs 2005-2006 quan acabés l'assignatura, per això vaig decidir que durant el primer semestre d'aquest curs, desenvoluparia tota la interfície gràfica amb el Visual Bàsic, desenvoluparia també la base de dades amb Microsoft Acces i el lligam entre aquestes dos parts. Després, al finalitzar l'assignatura, desenvoluparia el codi amb Prolog i crearia el lligam amb la interfície d'usuari.

Així que l'estructura d'hores quedava d'aquesta forma:

18 setmanes del primer quatrimestre.

- 10 setmanes x 4 hores per setmana = 40 hores per la confecció de la base de dades en Microsoft Acces
- 8 setmanes x 4 hores per setmana = 32 hores per la confecció de l'interficie gràfica amb el Visual Basic. Disseny de pantalles, requisits i implementació

16 setmanes del segon quatrimestre

- 10 setmanes x 4 hores per setmana = 40 hores per la modelització i programació de constraints en Prolog
- 6 setmanes x 4 hores per setmana = 24 hores per la interacció entre tots els components i fer la memòria.

Donant un total de **136** hores aproximadament

Evidentment, ja m'esperava que la cosa no seria tant senzilla, ja que l'idoni hauria estat fer-ho al revés, començar per la modelització i acabar amb l'intefície d'usuari. Sabia que em sortirien complicacions, canvis a la base de dades i a la interfície gràfica, no obstant això, treballar abans amb aquestes dues parts, m'ha fet agafar una dinàmica que després, amb les modificacions, m'ha estat molt més senzill de treballar-hi. Quedant-me una estructura com aquesta:

18 setmanes del primer quatrimestre.

10 setmanes x 4 hores per setmana = 40 hores per la confecció de la base de dades en Microsoft Acces

8 setmanes x 4 hores per setmana = 32 hores per la confecció de l'interficie gràfica amb el Visual Basic. Disseny de pantalles, requisits i implementació

16 setmanes del segon quatrimestre

12 setmanes x **6** hores per setmana = **72** hores per la modelització i programació de constraints en Prolog i ***modificacions a l'inteficie gràfica i a la Base de Dades***.

4 setmanes x **8** hores per setmana = **32** hores per la interacció entre tots els components i fer la memòria.

3 dies finals x 8 hores per dia = 24 hores per acabar de confeccionar la memòria i solucionar petits errors de l'aplicació.

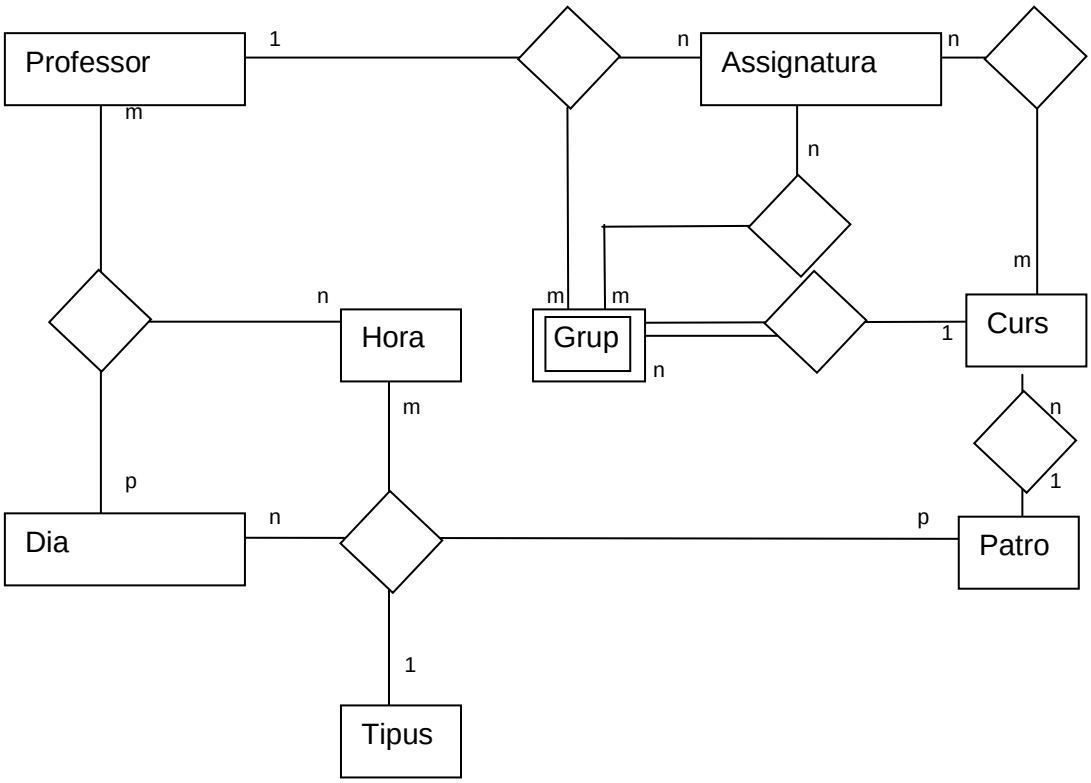
14 reunions x **1** hora per reunió = **14** hores de reunions amb el meu tutor. (gairebé totes les setmanes del segon quatrimestre i algunes del primer)

Que em dona un total de **220** hores aproximadament

5. Model de dades

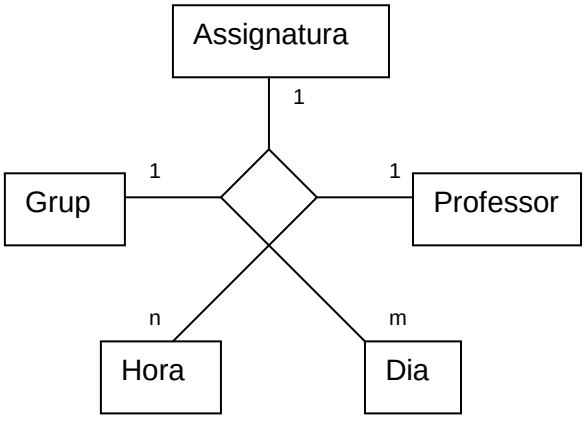
A continuació, mostrem l'estructura de la base de dades vista des del model entitat – relació i del model relacional

5.1. Model Entitat-Relació

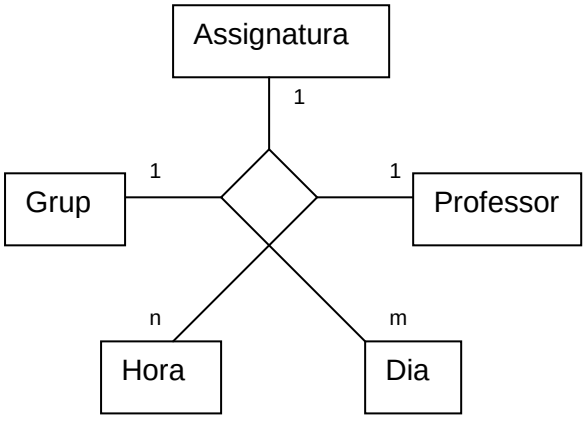


A més, cal afegir a aquest model les relacions entre Professor, Dia, Hora, Grup, que defineixen els Resultats Prefixats en els que l'usuari pot decidir amb antelació part de la solució final dels horaris. Així com una relació entre les mateixes taules on es guardarà el resultat definitiu.

ResultatsPrefixats



Resultat



5.2. Model Relacional

Professor: ID, DNI, Nom, Cognom1, Cognom2, Adreça, CodiPostal, Ciutat, Pais, Telefon, Email
Assignatura: ID, Nom, Durada, Sessions
Dia: ID, Nom
Hora : ID, Nom
Tipus : ID, Nom
Curs: ID, Nom, *IdPatroHorari*
Grup: ID, Nom, *IdCurs*
Patro: ID, Nom
AssigProfeGrup: *IdGrup, IdAssig, IdProfe*
AssigGrup: *IdGrup, IdAssig*
AssigCurs: *IdCurs, IdAssig*
HorariProfe: *IdDia, IdHora, IdProfe*
HorariPatro: *IdDia, IdHora, IdTipus, IdPatro*
ResultatPrefixat: *IdGrup, IdAssig, IdProfe, IdDia, IdHora*
Resultat: *IdGrup, IdAssig, IdProfe, IdDia, IdHora*

Podem veure a l'estructura de la base de dades dues coses interessants:

- La primera la veiem a les taules AssigGrup i AssigCurs. Aquestes dues taules contenen les assignatures que s'imparteixen als Grups i als Cursos respectivament. El que fa aquestes taules especials, és que un Curs està format per varis Grups i que per tant, una relació entre un Curs i una Assignatura és com moltes relacions entre tots els Grups d'aquest curs amb aquesta Assignatura. Aquesta repetició de dades es controla a la interfície gràfica, que no permet que un Grup i el Curs al que pertany tinguin una relació amb la mateixa Assignatura a la taula corresponent.

- La segona es troba en les taules de ResultatPrefixat i Resultat. Les dues taules contenen exactament el mateix tipus de dades, però la diferència és que una mostra els resultats que ha fixat l'usuari i l'altra els resultats que ha donat l'aplicació. Així doncs, per no repetir dades, entre les dues taules s'hauria de confeccionar el resultat final de forma que la taula resultat hauria de tenir totes les hores fixades per l'aplicació a excepció de les que ha fixat l'usuari, que es troben a la taula de ResultatPrefixat.

Aquesta forma de treballar no només és complexa i poc entenedora, sinó que amés no permet modificar les hores fixades per l'usuari sense alterar el resultat obtingut anteriorment. Per exemple, si hem arribat a un resultat vàlid, però hem decidit esborrar les hores prefixades per l'usuari, el resultat que ens quedarà serà incomplet obligant a generar un nou resultat que segurament no serà el mateix. Així doncs per solucionar el problema, el resultat final es trobarà a la taula de Resultat i les dades repetides seran les de ResultatPrefixat que hem de mantenir per tenir-ne constància.

6. Desenvolupament

El projecte s'ha desenvolupat en una interfície gràfica d'usuari feta amb Visual Basic per la seva facilitat a l'hora de connectar-lo tant amb la part del còmput com amb la base de dades. La part del còmput s'ha realitzat amb SICStus Prolog, que inclou resolució de constraints (restriccions) sobre variables de domini finit i s'adequa al projecte. Finalment he escollit una base de dades feta amb MS Access per a la seva facilitat de manipulació i elaboració.

6.1. Implementació de la base de dades

A continuació es mostra la implementació en Acces del model de dades

Taula: PROFESSOR

Camp	Tipus	
Id	Numèric	Clau primària (autoincremental)
DNI	text(9)	
Nom	text(15)	
Cognom1	text(15)	
Cognom2	text(15)	
Adreça	text(15)	
CodiPostal	text(5)	
Ciutat	text(15)	
País	text(15)	
Telèfon	Numèric	
Email	text(50)	

Taula: ASSIGNATURA

Camp	Tipus	
Id	Numèric	Clau primària (autoincremental)
Nom	text(15)	
Durada	Numèric	
Sessions	Numèric	

Taula: **DIA**

Camp	**Tipus**	
Id	Numèric	Clau primària (autoincremental)
Nom	text(15)	

Taula: **HORA**

Camp	**Tipus**	
Id	Numèric	Clau primària (autoincremental)
Nom	text(15)	

Taula: **TIPUS**

Camp	**Tipus**	
Id	Numèric	Clau primària (autoincremental)
Nom	text(15)	

Taula: **CURS**

Camp	**Tipus**	
Id	Numèric	Clau primària (autoincremental)
Nom	text(15)	
idPatróHorari	Numèric	Clau forana

Taula: **GRUP**

Camp	**Tipus**	
Id	Numèric	Clau primària (autoincremental)
Nom	text(15)	
idCurs	Numèric	Clau forana

Taula: **PATRO**

Camp	**Tipus**	
Id	Numèric	Clau primària (autoincremental)
Nom	text(15)	

Taula: **ASSIGPROFEGRUP**

Camp	Tipus	
IdGrup	Numèric	Clau primària / Clau forana
IdAssig	Numèric	Clau primària / Clau forana
IdProfe	Numèric	Clau primària / Clau forana

Taula: **ASSIGGRUP**

Camp	Tipus	
IdGrup	Numèric	Clau primària / Clau forana
IdAssig	Numèric	Clau primària / Clau forana

Taula: **ASSIGCURS**

Camp	Tipus	
IdCurs	Numèric	Clau primària / Clau forana
IdAssig	Numèric	Clau primària / Clau forana

Taula: **HORARIPROFE**

Camp	Tipus	
IdDia	Numèric	Clau primària / Clau forana
IdHora	Numèric	Clau primària / Clau forana
IdProfe	Numèric	Clau primària / Clau forana

Taula: **HORARIPATRO**

Camp	Tipus	
IdDia	Numèric	Clau primària / Clau forana
IdHora	Numèric	Clau primària / Clau forana
IdTipus	Numèric	Clau primària / Clau forana
IdPatró	Numèric	Clau primària / Clau forana

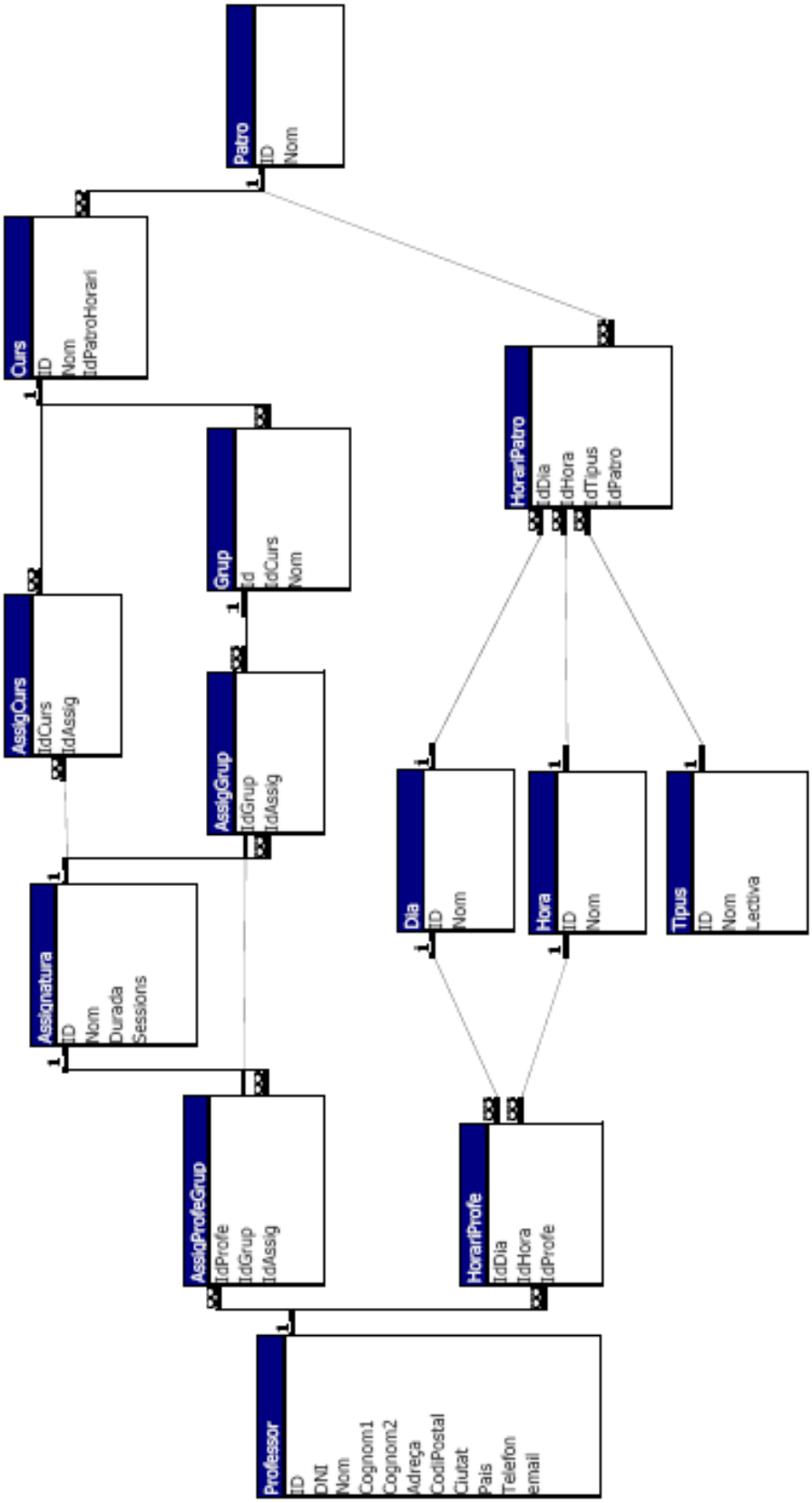
Taula: **RESULTATPREFIXAT**

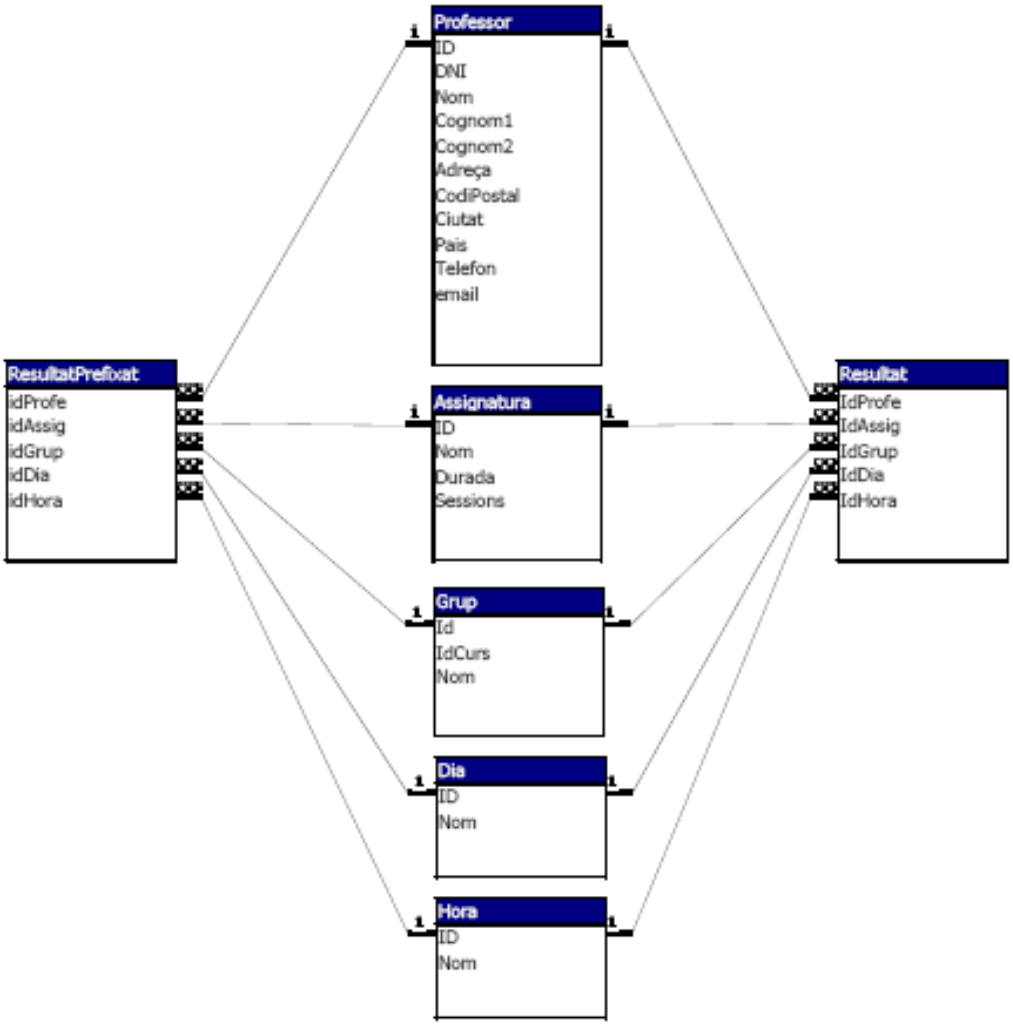
Camp	Tipus	
IdGrup	Numèric	Clau primària / Clau forana
IdDia	Numèric	Clau primària / Clau forana
IdHora	Numèric	Clau primària / Clau forana
IdAssig	Numèric	Clau forana
IdProfe	Numèric	Clau forana

Taula: **RESULTAT**

Camp	Tipus	
IdGrup	Numèric	Clau primària / Clau forana
IdDia	Numèric	Clau primària / Clau forana
IdHora	Numèric	Clau primària / Clau forana
IdAssig	Numèric	Clau forana
IdProfe	Numèric	Clau forana

6.2. Relacions entre taules




Professor
ID
DNI
Nom
Cognom1
Cognom2
Adreça
CodiPostal
Ciutat
Pais
Telefon
email
ResultatPrefixat
idProfe
idAssig
idGrup
idDia
idHora
Assignatura
ID
Nom
Durada
Sessions
Resultat
IdProfe
IdAssig
IdGrup
IdDia
IdHora
Grup
Id
IdCurs
Nom
Dia
ID
Nom
Hora
ID
Nom

6.3. Interfícies entre els components: MS Access, Visual Basic i SICStusProlog

6.3.1. Visual Basic – MS Access

Per poder guardar, esborrar o modificar les dades, és necessari crear un lligam entre la interfície d'usuari i la base de dades.
Visual Bàsic 6.0 conté uns elements anomenats "Data" que permeten fer-ho, creant una connexió amb la base de dades. Al principi de l'aplicació, es demana a l'usuari que esculli la base de dades sobre la que es treballarà o que en creï una de nova a partir d'una ja existent o en blanc, i un cop escollida o creada, es connecta un element "Data", que es troba al menú principal, a aquesta base de dades creant un únic cop la connexió.
Treballar d'aquesta manera té un desavantatge important; es manté una connexió a memòria amb la base de dades que estarà la major part del temps inactiva. L'avantatge, però, és que treballant d'aquesta manera aconseguim no haver de crear una connexió cada vegada que vulguem fer un accés a la base de dades, amb el temps que això comporta.

6.3.2. Visual Basic – SICStus Prolog

SICSTus Prolog proporciona una interfície per a carregar i cridar programes Prolog des de Visual Bàsic. D'aquesta manera podem crear un lligam entre la interfície d'usuari i el codi Prolog obtenint una resposta i transformant-la per a que sigui intel·ligible per a l'usuari.
Les funcions que es poden realitzar són;

- Passar una "query" a Prolog;
- Avaluar una "query" en Prolog;
- Obtenir el valor (string o enter) assignat a una variable per a l'avaluació de la "query" en prolog;
- Obtenir informació sobre excepcions a la "query".

Abans però caldrà crear la connexió amb el SICSTus Prolog i carregar el/s fitxer/s que contingui/n els predicats amb les funcions:

- **PrologInit**() – carrega i inicialitza l'interfície amb SICSTus per a Visual Basic. Retorna 1 si tot ha estat correcte i -1 altrament.
- **PrologQueryCutFail**(*query*) – troba la primera solució de la *query* passada com a String, talla i falla, retorna 1 si s'ha realitzat correctament.
  Això ens permet realitzar crides com *"if PrologQueryCutFail ("load_files(app(sudoku))") <> 1 then Go Error"*, que carrega el codi que es troba al fitxer sudoku.pl
- **PrologOpenQuery**(*query*) – obre la *query* (String) i ens torna un identificador (enter). Si aquest identificador és -1, significa error.
- **PrologNextSolutions**(*qid*) – executa la query *qid* (l'enter que ha retornat al fer OpenQuery) i dóna la següent solució. Retorna 1 si tot ha anat correctament, -1 si hi ha hagut error i 0 si no hi ha més solucions.
- **PrologGetLong**(*qid*,*var*,*valor*) – posa a *valor* l'enter que se li ha donat a la variable *var* (String) durant l'execució de la query *qid*. També pot ser PrologGetString(*qid*,*var*,*valor*).
- **PrologCloseQuery**(*qid*) – tanca la query *qid*.

A més, per a la correcta utilització de les llibreries per a la interfície, necessitem que la llibreria SICSTus Prolog i Visual Basic (vbsp.dll) estigui al Path del sistema operatiu.

6.4. El codi Prolog

Com hem explicat a l'apartat de modelització dins d'alternatives de disseny, hem escollit un model de dades que ens permet assignar durades a les assignatures, però també hi ha altres dades de les que no hem parlat com els professor, grups, etc. En els següents apartats, mostrem una sèrie de fets i de regles que ens mostren la resta del nostre model de dades i que ens ajuden a aconseguir un resultat.

6.4.1. Fets

professor/2: conté dos camps, el primer d'ells és un identificador que el diferencia de la resta de professors i el fa únic, i el segon és una llista d'hores prohibides on aquest professor no pot impartir classe. Aquestes hores han estat mapejades de forma que la primera hora de dimarts és la immediatament següent de l'última de dilluns i així per a tots els dies.

Ex. `professor(5,[53,54,55,56,57,58,59,60,61,62,63,64,65]).`

assignatura/3: conté 3 camps, el primer d'ells és un identificador que la fa única i la diferencia de la resta d'assignatures. El segon camp és la quantitat d'hores que dura cada sessió d'aquesta assignatura, i el tercer és el nombre de sessions que se n'imparteixen cada setmana.

Ex. `assignatura(1,1,3).`

grup/4: conté 4 camps, el primer d'ells i com ja és habitual, és un identificador que el fa únic i el diferencia de la resta de grups, el segon, és una llista que conté identificadors d'assignatures que ha de cursar aquest grup, el tercer és una llista d'hores lectives del grup i el quart és precisament el contrari, una llista amb les hores no lectives del grup.

Ex. `grup(1, [1,5,11,15,19,23,27,31,35,39,43,47],`
`[2,15,28,41,54,3,16,29,42,55,5,18,31,44,57,6,19,32,45,58,9,22,35`
`,48,61,10,23,36,49,62,11,24,37,50,63],`
`[1,14,27,40,53,4,17,30,43,56,7,20,33,46,59,8,21,34,47,60,12,25,3`
`8,51,64,13,26,39,52,65]).`

profeAssigGrup/3: conté 3 camps, el primer d'ells és un identificador d'un professor, el segon és un identificador d'una assignatura, i el tercer, és una llista d'identificadors de tots els grups a que el professor imparteix l'assignatura corresponent a l'identificador.

Ex. `profeAssigGrup(1,1,[1,2,3]).`

resultatPrefixat/1: conté un únic camp que consisteix en una llista de tuples amb 4 elements cadascuna. El primer és un identificador de grup, el segon és un identificador d'assignatura, el tercer un identificador de professor i el quart diu l'hora en que aquest professor impartirà classe de l'assignatura especificada a un grup en concret.

Ex.

```
resultatPrefixat([(1,1,1,2),(1,1,1,54),(2,1,1,16),(2,1,1,29)]).
```

En tots aquests fets, es poden utilitzar paraules als llocs dels identificadors per fer-ho més intel·ligible, com per exemple `assignatura('Mates1',1,3).` Però en cap cas, es podran substituir les hores per algun altre tipus de variables.

6.4.2. Regles

Hpag(*?Resultat* **)** és el predicat que dóna la solució del nostre horari en funció del fets esmentats anteriorment. Per aconseguir-ho, primer de tot obtindrà els grups amb les seves assignatures corresponents i les hores en les que s'impartiran, encara no definides però si amb un domini finit, amb el predicat **obtenirGrups(** *?Grups* **)** que explicarem més endavant. Posteriorment, restringirà les hores d'aquestes assignatures amb el predicat **restringirHores(** *+Grups* **)** on se li dóna com a paràmetre el resultat del primer predicat, i aquest restringeix les hores corresponents. Finalment, amb el predicat **obtenirHores(** *+Grups***,** *?Hores***)**, obtenim una llista plana de les hores (encara no definides, però si restringides), per poder cridar el predicat **labeling(** *:Opcions***,** *+Hores***)** de Prolog, on *Opcions* és una llista d'opcions de cerca i *Hores* una llista de variables de domini finit, que se'ls hi donarà valors concrets. Finalment, cridem el predicat **aplanarTot(** *+Grups***,** *?Resultat***)**, que retorna un resultat apte per l'aplicació Visual Basic.

```
hpag(Grups2):-
      obtenirGrups(Grups),
      restringirHores(Grups),
      obtenirHores(Grups, Hores),
      labeling([],Hores),
      aplanarTot(Grups,Grups2).
```

aplanarTot(*+List1***,** *?List2* **)**, on *List1* és una llista de tuples amb la següent estructura: (*IdGrup*, *ListAssigs*) on *ListAssigs* és una llista de llistes amb tuples (*idProfe idAssig*, *Hora*). Ex:
"[(Grup**,[[(1,1,3),(1,1,15),(1,1,22)],[(2,1,5),(2,1,16),(2,1,21)],...]**),(Grup,[...]),...]", i List2 és una llista amb les mateixes tuples, però amb la llista de llistes transformada en una llista plana de les mateixes tuples. Ex:
"[(Grup**,[(1,1,3),(1,1,15),(1,1,22),(2,1,5),(2,1,16),(2,1,21),...]**),(Grup,[...]),...]"

```
aplanarTot([],[]).
aplanarTot([(X1,X2)|XS],[(X1,Y2)|YS]):-
      aplanar(X2,Y2),
      aplanarTot(XS,YS).
```

En el conjunt de predicats podem diferenciar clarament tres parts; la primera és la d'obtenció de dades, on per tots els grups s'obté una llista de les assignatures que han de cursar, els professors que les imparteixen i les hores en les que es cursen. Aquestes hores, de moment no tenen valor, ni tant sols estan restringides, només tenen un domini possible definit per les hores lectives del grup. La segona part serà la de restringir els valors d'aquestes hores i finalment la tercera la de donar-los valors reals.

6.4.2.1. Primera Part

L'objectiu d'aquesta primera part és aconseguir una llista de tuples "`[(Grup1,[[(Assig1,Profe1,Hora1),(Assig1,Profe1,Hora2),...],[(Assig2,Profe2,Hora3),...] ,...]),(Grup2,[...]),...]`". Amb tots grups que hi hagi a la part de fets i cadascun d'ells acompanyat d'una llista de tuples que contenen l'assignatura, el professor que les imparteix, i l'hora de la classe. Podem veure que cada assignatura té vàries sessions per setmana, així que dins de els llistes dels grups, agruparem les tuples-assignatura que siguin iguals per subllistes per aconseguir una estructura més ordenada i intel·ligible. `(Grup,[LlistaTuplesAssig1,LlistaTuplesAssig2,...])`.

Per fer-ho primer de tot s'ha d'aconseguir una llista de tots els grups amb el predicat **getGrups**(*+List, ?LlistaGrups*), que com a primer paràmetre li passarem una llista buida, ja que utilitza una immersió per controlar si un grup ja ha estat tractat o no i com a segon paràmetre retorna una llista amb els identificadors de tots els grups. Després, cridarem al predicat **obtAssig**(*+LlistaGrups, ?LlistaGrupsAssigs*), que com a primer paràmetre se li dóna una llista amb tots els identificadors dels grups, i com a segon, retorna una llista amb el format desitjat per a tots els grups

```
obtenirGrups(Grups):- getGrups([],G), obtAssig(G,Grups).
```

Com hem dit, el predicat getGrups/2, utilitza una immersió per aconseguir la llista de tots els grups i controlar que no en repeteixi cap. Per fer-ho utilitza el mètode **member**(*?Element, ?List*) de Prolog que ens diu si un element pertany a una llista.

```
getGrups(X ,[A|XS]):-
      grup(A,_,_,_),
      \+member(A,X),
      getGrups([A|X], XS),
      !.
getGrups(_,[]).
```

El predicat **obtAssig**(*+LlistaGrups, ?LlistaGrupsAssigs*), ens crea l'estructura que necessitem a partir d'una llista de grups. Per a cada grup de la llista crida al predicat **getAssigs**(*+Grup, +List, ?LlistaAssigs*) que donat un *Grup* i una llista d'assignatures tractades d'aquest grup, inicialment li donem una llista buida ja que utilitza una immersió, obté la llista de les seves assignatures amb l'estructura requerida.

```
obtAssig([],[]).
obtAssig([X|XS], [(X,Y)|YS]):-
      getAssigs(X,[],Y),
      obtAssig(XS,YS).
```

getAssigs/3 Aconsegueix totes les assignatures que cursa un grup amb l'estructura de sessions i assignatures que volem a partir d'una immersió per controlar les assignatures que ja s'han tractat. Primer de tot, obté la llista d'assignatures que cursa el grup amb el fet **grup**/4, després escull una assignatura amb el predicat de Prolog **member**/2, que obté un element de la llista d'assignatures, després, amb el mateix predicat, comprova que aquesta no es trobi en la llista d'assignatures tractades. Un cop escollida l'assignatura que tractarem, obtenim el nombre de sessions per setmana del fet **assignatura**/3 i el professor que la imparteix a partir del predicat **obtenirProfessor**(*+Grup, +Assig, ?Professor*). Finalment, a partir d'aquests elements, crida al predicat **obtSessions**(*+Grup, +Professor, + Assignatura, +Session, ?Resultat*), que donat un Grup, Professor, Assignatura i nombre de Sessions, retorna una llista amb l'estructura desitjada.

```
getAssigs(Nom,L,[Res|YS]):-
    grup(Nom,Llista,_,_),
    member(Assig,Llista),
    \+member(Assig,L),
    assignatura(Assig,_,N),
    obtenirProfessor(Nom, Assig, Profe),
    obtSessions(Nom, Profe, Assig, N, Res),
    getAssigs(Nom,[Assig|L],YS),
    !.
getAssigs(_,_,[]).
```

obtenirProfessor/3 aconsegueix el professor que impartirà una assignatura a un grup concret amb el fet **profeAssigGrup**/3 i el predicat de Prolog **member**/2

```
obtenirProfessor(Grup, Assig, Profe):-
     profeAssigGrup(Profe,Assig,List),
     member(Grup,List).
```

obtSessions/4 crea una llista amb tantes tuples (*Assig,Profe,Hora*) com nombre de sessions li corresponen a la setmana a aquesta assignatura, també assigna un domini a les hores en les quals es pot impartir l'assignatura a partir de les hores lectives del grup. Primer de tot obté les hores lectives del grup amb el fet **grup**/4, després, obté un domini amb el predicat **creardominis**(*+Llista, ?Domini*), que donada una llista d'enters, retorna una llista amb el domini corresponent a tots els elements de la llista. Finalment, assigna aquest domini a la variable corresponent, a la hora que s'impartirà aquesta assignatura.

```
obtSessions(Grup,Profe, Assig, 1,[(Assig,Profe, Z)]):-
      grup(Grup,_,H,_),
      creardominis(H,Dom),
      Z in_set Dom.
obtSessions(Grup,Profe, Assig, N,[(Assig,Profe, Z)|Res]):-
      N>1,
      grup(Grup,_,H,_),
      creardominis(H,Dom),
      Z in_set Dom,
      M is N-1,
      obtSessions(Grup,Profe,Assig,M,Res).
```

Creardominis/2 només crida a la funció de Prolog **list_to_fdset**(*+List, ?Dom*) que a partir d'una llista d'enters, retorna un domini per aplicar a una variable.

```
creardominis(L,S):-list_to_fdset(L,S).
```

6.4.2.2. Segona Part

Aquesta segona part consisteix a restringir el valor de les variables de les hores. Per fer-ho, necessitarem seguir tres passos. Primer de tot restringirem les hores que ja han estat prefixades al Visual Basic i que tindran un valor concret (no seran de domini finit), això ho farem amb el predicat **horesPrefixades**(*+Llista , +Grups*), que donada una llista de tuples (Grup, Assig, Profe, Hora), assignarà valors a les variables de la nostra estructura *Grups*. Després, controlarem els solapaments entre les hores de cada Grup, ja que un grup no pot fer dos assignatures al mateix temps, amb el predicat **noSolapEntreHores**(*+Grups)*. Un cop fet això, cridarem al predicat **controlDies** (*+Grups*) per restringir que aquestes assignatures no es puguin cursar dos cops al mateix dia, i finalment, controlarem que les assignatures no es puguin cursar durant les hores que el professor que les imparteix no tingui disponibles i que un professor no pugui impartir dues assignatures al mateix temps amb el predicat **controlProfessor**(+ *Grups*).

```
restringirHores(Grups):-
      resultatPrefixat(X),
      horesPrefixades(X,Grups),
      noSolapEntreHores(Grups),
      controlDies(Grups),
      controlProfessor(Grups).
```

horesPrefixades/2 crida al predicat **hpre**(*+Tupla, +Grups*) per cada element de la llista d'hores prefixades. **hpre**/2 aconsegueix la tupla (Grup,LlistaAssigs) que li correspon al grup de la llista *Grups*, i amb ella crida al predicat **fixarResultat** (*+Tupla, +Lista*) on amb la tupla (Assignatura, Hora) extreta del resultat prefixat, assignarà valor a una variable de la Llista d'assignatures del grup.

```
horesPrefixades([],_).
horesPrefixades([X|XS], Grups):-
      hpre(X,Grups),
      horesPrefixades(XS,Grups).
```

```
hpre((Grup,A,_,C),[(Grup,Y)|_]):-
      fixarResultat((A,C),Y).
hpre((Grup,A,B,C),[(Grup2,_)|YS]):-
      Grup \= Grup2,
      hpre((Grup,A,B,C),YS).
```

fixarResultat/2 busca la llista corresponent a l'assignatura de la tupla del primer paràmetre dins de la llista de llistes del segon. Un cop trobada, intenta unificar el resultat prefixat amb el primer element d'aquesta, si no ho aconsegueix, significa que aquesta hora ja ha estat prefixada anteriorment, i ho intenta amb la següent i així successivament fins que té èxit.

```
fixarResultat((A,B),[[(A,_,B)|XS]|_]).
fixarResultat((A,B),[[(A,_,_)|XS]|_]):-
      fixarResultat((A,B),[XS]).
fixarResultat((A,B),[[(D,_,_)|_]|YS]):-
      D \= A,
      fixarResultat((A,B),YS).
```

La segona restricció que s'aplica a les hores, és que les hores no se solàpin entre elles en un mateix grup. Això ho farem amb el predicat **noSolapEntreHores**(*+Grup*). Primer de tot, aplanem la llista de llistes de cada grup en una sola llista amb el predicat **aplanar**(*+LlistaLlistes, ?LlistaPlana*) per tractar-les totes igual. Després utilitzem el predicat **generardisjoint**(*+Llista, ?Disj*) que donat una *Llista* de tuples (Assig,Profe,Hora), genera una llista *"Disj"* de tipus "f(HoraInici,Durada)" per cada element de la llista. Després, amb el fet **grup**/4, obtenim les Hores prohibides del grup, aquelles en les que no es pot impartir classe, i amb el predicat **disjHoresProibides**(*+Llista, ?Disj*), obtenim una altra llista de tipus "f(Hora,Durada)", que es concatenarà amb l'anterior amb el predicat **concatenar***(+List1, +List2, ?ListResultat*) que s'utilitzarà per cridar el predicat **disjoint1**(*+List*) de prolog explicat a l'apartat 2.2. Aquest predicat aconsegueix que, tenint en compte aquestes hores i les seves durades, no se solàpin entre elles.

```
noSolapEntreHores([]).
noSolapEntreHores([(Grup,X)|XS]):-
      aplanar(X,Y),
      generarfdisjoint(Y,Disj1),
      grup(Grup,_,_,Hores),
      disjHoresProibides(Hores,Disj2),
      concatenar(Disj1,Disj2,Z),
      disjoint1(Z),
      noSolapEntreHores(XS).


generarfdisjoint([],[]).
generarfdisjoint([(Nom,_, Hora)|XS],[f(Hora,B)|YS]):-
      assignatura(Nom, B, _),
      generarfdisjoint(XS,YS).
```

```
aplanar([[]|L],T):- aplanar(L,T),!.
aplanar([X|L],T):-
      list(X),
      aplanar(X,A),
      aplanar(L,B),
      concatenar(A,B,T),!.
aplanar([X|L],[X|B]):-aplanar(L,B),!.
aplanar(X,X):- \+list(X).

list([_|_]).
```

La tercera restricció que es fa a les hores és **controlDies**(*+Grups*). Aquest restringeix les hores perquè una assignatura no es cursi més d'un cop al mateix dia per a cada grup, per fer-ho crida al predicat **diferentsDiesGrup** *(+Llista*) on es passa com a paràmetre la llista de llistes de tuples (Assignatura,Professor,Hora) del grup.

```
controlDies([]).
controlDies([(_,X)|XS]):-
      diferentsDiesGru(X),
      controlDies(XS).
```

Per cada llista de la llista de llistes que correspon a una mateixa assignatura, cridem el predicat **diferentsDies**(*+Llista*) que aconsegueix el dia en que es cursa cada sessió amb el predicat **difDi**(*+LlistaTuples, ?LlistaDies*) de la mateixa assignatura i força que aquests siguin diferents amb el predicat de Prolog també explicat a l'apartat 2.2. **all_different**(*+List*)

```
diferentsDiesGru([]).
diferentsDiesGru([X|XS]):-
      diferentsDies( X),
      diferentsDiesGru(XS).
```

```
diferentsDies(X):-
      difDi(X,Y),
      all_different(Y).

difDi([],[]).
difDi([(_,_,X)|XS],[Y|YS]):-
      horesdia(N),
      Y #= (X-1)/N,
      difDi(XS,YS).
```

Finalment, la ultima restricció de les hores és la que controla els professors, tant que les seves assignatures es cursin en horari disponible, com que no puguin estar impartint diferents assignatures al mateix moment. Per fer-ho, transformem la llista de Grups amb les seves hores en una llista plana on no ens importen els grups [(Assig,Profe,Hora)...], això ho aconseguim amb el predicat **obtenirTuples**(*+Grups, ?LlistaTuples*). Obtenim també tots els professors que existeixen amb el predicat **obtenirProfes**(*+LlistaTractats ?LlistaResultat*) que utilitza una immersió per controlar els professors que ja hem aconseguit. Amb aquests dos resultats, cridem al predicat **controlProfes**(*+LlistaProfessors, +LlistaTuples*).

```
controlProfessor(Grups):-
      obtenirTuples(Grups,Tuples),
      obtenirProfes([],Profes),
      controlProfes(Profes,Tuples).

obtenirTuples([],[]).
obtenirTuples([(_,X)|XS],Z):-
      aplanar(X,Y),
      obtenirTuples(XS,YS),
      concatenar(Y,YS,Z).
```

```
obtenirProfes([],[Nom|XS]):-
      professor(Nom,_),
      obtenirProfes([Nom],XS),!.
obtenirProfes(X,[Nom|XS]):-
      professor(Nom,_),
      \+member(Nom,X),
      obtenirProfes([Nom|X],XS),!.
obtenirProfes(_,[]).
```

controlProfes/2 és el que s'encarrega d'aplicar les restriccions relacionades amb els professors a les hores en que s'imparteix una assignatura. Per fer-ho tractarà cada professor de la llista de professors, al primer paràmetre, aconseguint primer de tot una llista apta pel predicat **disjoint1** del Prolog. Això ho farà amb el predicat **disjointProfe**(*+Profe, +LlistaTuples, ?Disj*) amb totes les hores que aquest imparteix. També obté una llista d'hores en la que el professor no pot impartir classe a partir del fet **professor**/2 i en crea una segona llista **disjoint1** amb el predicat **disjHoresProibides**(*+Hores, ?Disj*) per concatenar-la a l'anterior i cridar al predicat disjoint1 de Prolog. Així controlarà les dues restriccions alhora.

```
controlProfes([],_).
controlProfes([X|XS],Tuples):-
      disjointProfe(X,Tuples,Disj1),
      professor(X,Hores),
      disjHoresProibides(Hores,Disj2),
      concatenar(Disj1,Disj2,Disj),
      disjoint1(Disj),
      controlProfes(XS,Tuples).
```

disjointProfe/3 obté a partir d'un professor i una llista de tuples (Assig,Profe,Hora), una llista de f(Hora,Durada) per a totes les assignatures que imparteix aquest professor, on *Hora* és una variable de domini finit i la *durada* és la durada d'aquesta assignatura.

```
disjointProfe(_,[],[]).
disjointProfe(Profe,[(Assig,Profe,Hora)|XS],[f(Hora,B)|YS]):-
      assignatura(Assig,B,_),
      disjointProfe(Profe,XS,YS),!.
disjointProfe(Profe,[_|XS],YS):-
      disjointProfe(Profe,XS,YS).
```

disjointHoresProibides/2, és similar al predicat anterior, però en aquest cas s'obtindrà f(Hora,1) per a cada *Hora* de la llista que es passa com a primer paràmetre.

```
disjHoresProibides([],[]).
disjHoresProibides([X|XS],[f(X,1)|YS]):-
      disjHoresProibides(XS,YS).
```

6.4.2.3. Tercera Part

ObtenirHores/2 aconsegueix una llista amb totes les hores de totes les assignatures de tots els grups i les uneix en una sola llista amb l'objectiu de fer un **labeling**/2 sobre ella.

```
obtenirHores([],[]).
obtenirHores([(_,Llista)|XS],Z):-
      aplanar(Llista,List),
      obtenirHores2(List,Y),
      obtenirHores(XS,YS),
      concatenar(Y,YS,Z).

obtenirHores2([],[]).
obtenirHores2([(_,_,X)|XS],[X|YS]):- obtenirHores2(XS,YS).

concatenar([],Y,Y).
concatenar([X|XS],Y,[X|Z]):- concatenar(XS,Y,Z).
```

6.4.3. Optimització

Com hem vist a l'apartat 6.4.2., el predicat al que demanàvem la solució, estava format per tres grans parts, la primera d'elles "obtenir Grups", la segona era la de "restringir Hores" i la tercera la de "obtenirHores". Aquesta última aconseguia una llista de les hores (variables de domini finit) per poder-les passar com a paràmetre al **labeling/2** de Prolog i aconseguir un valor correcte per a totes elles. **labeling/2**, però requereix un altre paràmetre que anteriorment hem considerat una llista buida.

```
labeling([],Hores),
```

Aquesta llista, és una llista d'opcions de cerca que controla l'ordre en que les variables de domini finit reben un valor concret. Les opcions són:

- **leftmost** – Se selecciona la variable de més a l'esquerra.
- **min** - Se selecciona la variable de més a l'esquerra amb la fita inferior més petita.
- **max** - Se selecciona la variable de més a l'esquerra amb la fita superior més gran.
- **ff** - Se selecciona la variable de més a l'esquerra amb el domini més petit.
- **ffc** - Se selecciona la variable amb el domini més petit.
- **step** - Fa una elecció binària entre X #= B i X #\=B, on B és inferior o fita superior de X.
- **enum** - Fa una elecció múltiple per a X corresponent als valors del seu domini.
- **bisect** - Fa una elecció binària entre X #=< M i X #> M, on és M el punt mig del domini de X.
- **up** - el domini és explorat en ordre ascendent.
- **down** - el domini és explorat en ordre descendent.

Al fer proves sobre aquestes diferents opcions contra el mateix problema, podem veure els següents resultats:

- **sense opcions** – 17 segons
- **leftmost** – 21 segons
- **min** – més d'un minut
- **max** – més d'un minut
- **ff** – 1.5 segons
- **ffc** – 1.5 segons
- **step** – 24 segons
- **enum** – 23 segons
- **bisect** – 18 segons
- **up** – 20 segons
- **down** – 4 segons

Podem veure que hi ha una clara diferència de temps en les diferents opcions escollides, però no podem escollir una opció o una altre en funció d'aquesta prova, ja que és del tot insuficient. Aquest apartat vol mostrar la importància d'unes bones opcions de cerca per el nostre problema, però no pretén decidir quines són, ja que per això s'hauria de fer un estudi detallat de totes elles.

6.5. Visual Basic

En aquest apartat es mostren alguns dels elements més importants utilitzats al mòdul d'Interfície gràfica amb Visual Basic. Gairebé tots ells contenen un conjunt de propietats comunes i que s'usen constantment a l'aplicació.

- **Visible** (true/false) ens diu si l'element es visible per a l'usuari o està ocultat

- **Caption** ("String") ens diu el text que conté l'element. En alguns elements, aquesta propietat es diu **Text**, com a les caixes de text o les llistes.

- **Enabled**(true/false) ens diu si aquest element es pot seleccionar i que realitzi la seva funció al fer-ho.

Els elements més utilitzats són:

- **Label**: És una zona delimitada on es pot introduir un text.

- **TextBox:** És una caixa que conté o no un text. Si la propietat Enabled es troba a cert, aquesta caixa permet escriure-hi un text

- **ComboBox:** És un deplegable que conté una llista d'elements. Si la propietat Enabled es troba a cert, permet deplegar-lo i seleccionar-ne un.

- **ListBox:** És una llista d'elements amb una barra d'Scrol que ens permet moure'ns per ella. Si la propietat enabled es troba a cert, ens permet seleccionar un element.

- **CommandButton:** És un botó amb un text dins seu (caption). Si la seva propietat Enabled es troba a cert, ens permet clicar-lo i executar el codi que té assignat.

- **Data:** Aquests elements són els que ens permeten crear un lligam amb la nostra base de dades i el Visual Basic. Al principi de l'aplicació, s'assigna la base de dades sobre la que treballarem a un d'aquests elements i cada cop que volem accedir-hi, utilitzarem una variable Database i li donarem el valor daquest Data. Amb ella podem realitzar insercions, modificacions i consultes a la base de dades. Les consultes, però necessiten un altre element anomenat Recordset, al que se li donarà el valor del resultat de la consulta i s'iterarà sobre ell per mostrar-la.

- **MsFlexGrid:** És l'element utilitzat per crear totes les graelles de l'aplicació. Aquest element conté un conjunt de files i de columnes que confeccionen una graella de cel·les. A cadascuna d'elles se li pot assignar un color de fons, un text i fins i tot un color de text. També es poden definir funcions al seleccionar una cel·la concreta aconseguint la seva posició amb els atributs col i row (columna i fila).

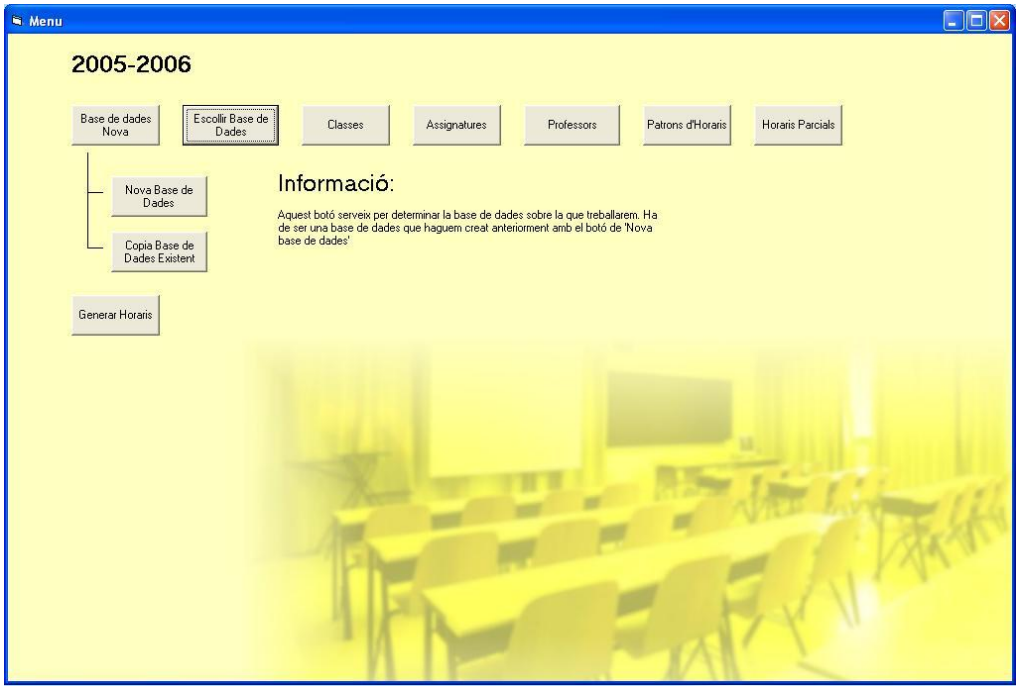
- **CommonDialog:** Aquest element ens permet mostrar una sèrie de menús estàndards com el de imprimir, obrir o guardar un fitxer amb les funcions showPrinter, showOpen, showSave..

7. L'aplicació

Tot seguit descriurem les funcionalitats finals de l'aplicació pas a pas tot seguint les diferents interfícies desenvolupades.

7.1. Menú Principal

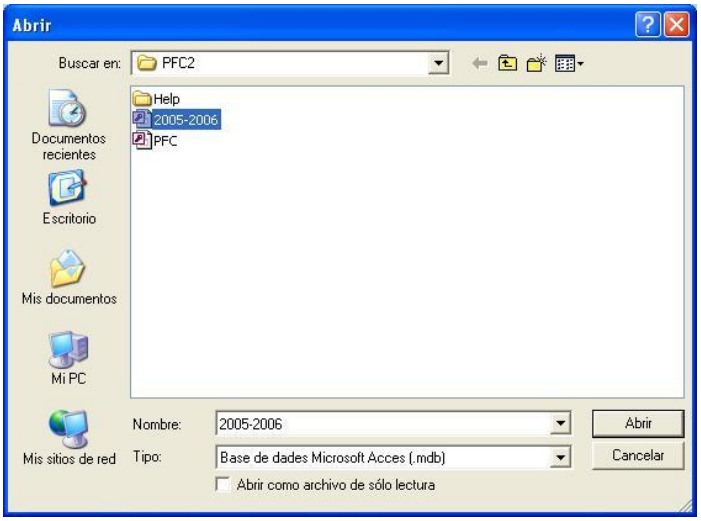
Al iniciar l'aplicació, aquest és el primer menú que ens apareix. És sobre el que es basa tota l'aplicació, ja que des d'ell es pot accedir pràcticament a tot arreu. Inicialment però només dóna a escollir dues de les opcions: "Escollir Base de Dades" i "Base de Dades Nova".



Les dues opcions acaben lligant la nostra aplicació amb una base de dades (buida o no) i només deixa escollir aquestes perquè per iniciar la nostra aplicació és essencial que estigui lligada a una base de dades sobre la que poder treballar. Un cop seleccionada la base de dades, ja podrem accedir a la nostra aplicació lliurement. A l'apartat de Base de dades Nova, apareixen dues noves opcions: "Copia Base de Dades Existent" i "Nova Base de Dades". Escollirem una opció o una altra depenent del que vulguem fer.

7.2. Escollir La base de dades

Tant per al menú “Escollir Base de Dades” com per el de “Copiar Base de dades Existent”, apareixerà un menú que ens obliga a seleccionar una base de dades ja existent, en un cas per obrir-la i en un altre per copiar-la. Per escollir una base de dades, és bàsic tenir-ne una de creada. Aquest menú només accedirà a bases de dades de MS Acces i només funcionarà amb les bases de dades creades a partir del model de l’aplicació.



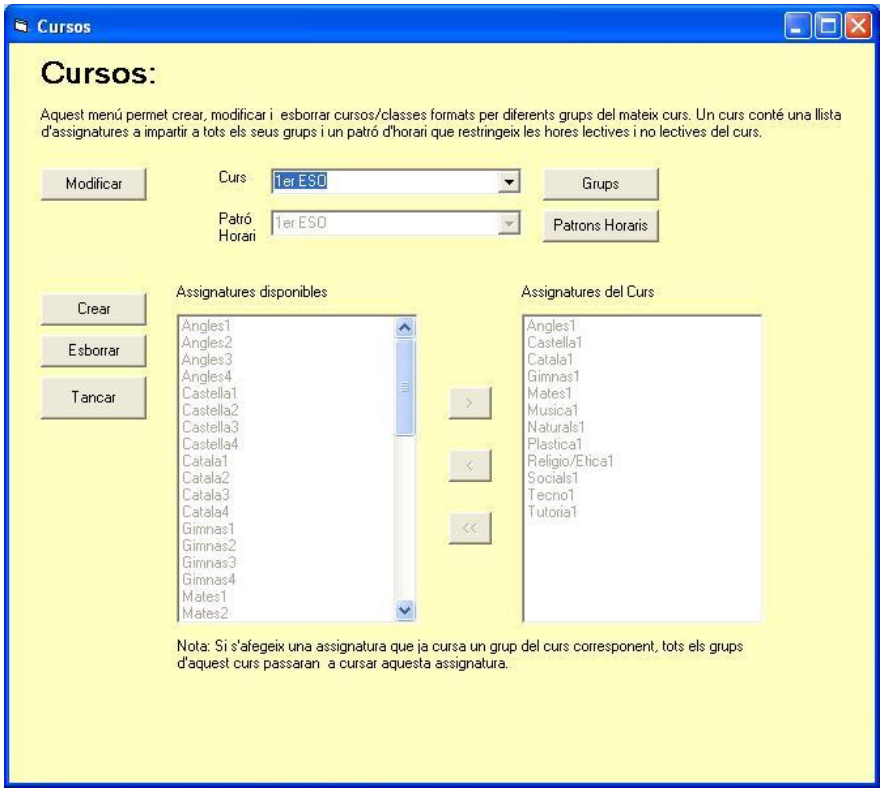
La seva utilització és prou coneguda per tothom; es busca la base de dades sobre la que volem treballar i s’apreta el botó “Abrir” per tornar a l’aplicació i seguir treballant, en cas d’apretar “cancelar”, l’aplicació seguirà inhabilitada.

7.3. Guardar una nova Base de dades

A aquesta finestra només s'hi podrà accedir des de "Nova base de Dades" i "Copiar Base de Dades existent". En el segon cas, haurem hagut de escollir anteriorment quina base de dades és la que volem copiar. Tant si la base de dades és nova com copiada, escollirem el nom amb el que la volem guardar i el lloc. Finalment, si tot ha anat correctament, el menú estarà tot operatiu per començar a treballar sobre la base de dades que acabem de guardar.

7.4. Curs

El menú de Cursos és l'encarregat de crear, modificar i esborrar els Cursos de la base de dades. Podem veure que hi ha un desplegable amb tots els cursos ja introduïts anteriorment i que quan n'escollim un d'ells s'actualitzen totes les seves propietats a la resta del menú.



Aquestes propietats són:

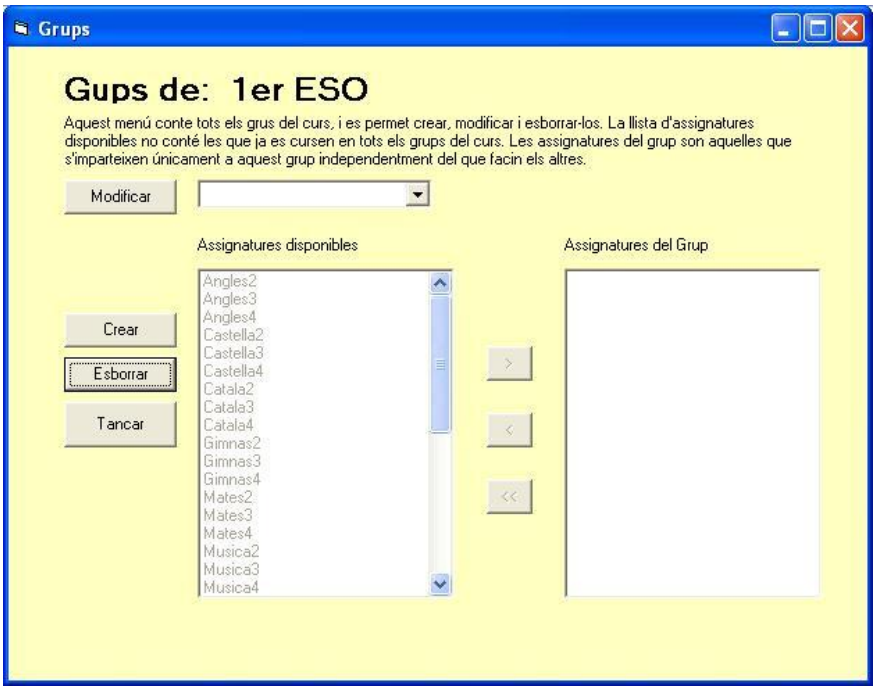
- Patró Horari: És el Patró de l'Horari que s'assigna a aquest curs, ja l'explicarem més detalladament en el seu apartat, només dir que serveix per delimitar les hores lectives i no lectives del curs.
- Assignatures disponibles: És una llista amb totes les assignatures existents.
- Assignatures del Curs: És una llista amb totes aquelles assignatures que estan assignades a aquest curs i per tant a tots els seus grups.

També podem veure en aquest menú tres opcions; una de crear, una de modificar i una altra d'esborrar. Les dues últimes requereixen haver seleccionat un curs del desplegable anteriorment. Les opcions de modificar i crear, habiliten conjunt d'accions d'aquest menú: escriure un nom (nou o no) del grup, modificar/escollir el patró que desitgem amb el desplegable "Patró Horari" i finalment afegir a "Assignatures del Curs" les assignatures que vulguem de la llista d'assignatures. També podem veure que han desaparegut o inhabilitat les altres accions i que per el contrari n'han aparegut unes de noves que són les d'Acceptar i Cancel·lar, que guarda els canvis realitzats a la base de dades en el cas d'Acceptar o les desestima en el cas de Cancel·lar.

Finalment tenim també dos botons més; un de "Grups" i un altre de "Patrons Horaris" que ens envien directament als menús corresponents.

7.5. Grup

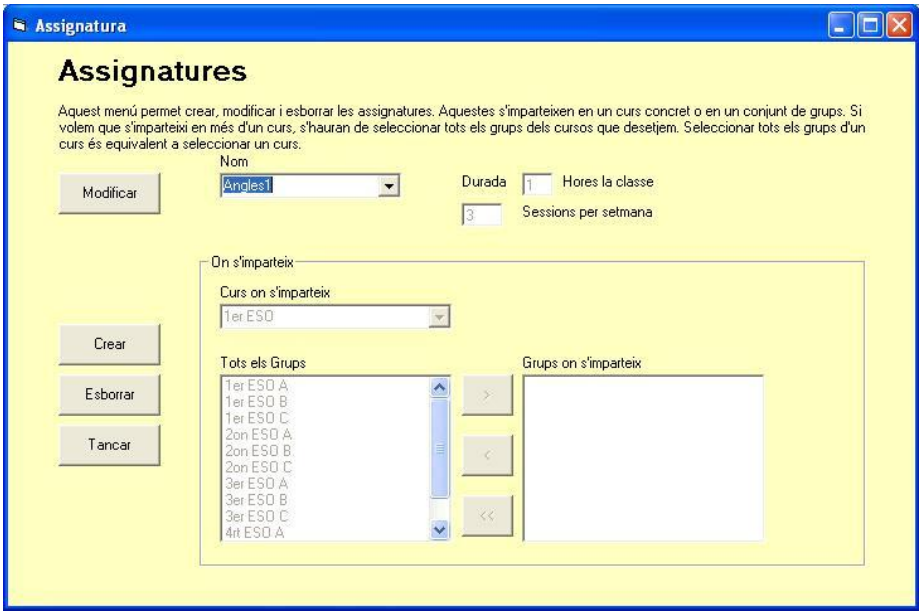
Aquest és el menú encarregat de crear, modificar i esborrar els grups que pertanyen a un curs. A aquest menú només s'hi pot accedir a través del menú Cursos, on després de seleccionar-ne un, es dóna aquesta opció.



A part de les opcions de modificar, esborrar i modificar que ja eren presents en el menú de Cursos, podem veure dues llistes d'assignatures d'aquest grup. Aquestes llistes també eren al menú anterior, però amb una petita diferència. En aquest menú, a la llista d'assignatures disponibles, no hi ha cap de les assignatures assignades al curs anteriorment, ja que es considera que si el curs del que forma part un grup ja cursa un conjunt d'assignatures, aquestes es s'imparteixen a tots els grups que en formen part. Independentment d'això, al grup en concret que estem tractant, se li poden assignar un conjunt d'assignatures que no han estat assignades al seu curs i que únicament aquest grup (juntament amb els altres que segueixin aquest mètode) cursarà.

7.6. Assignatures

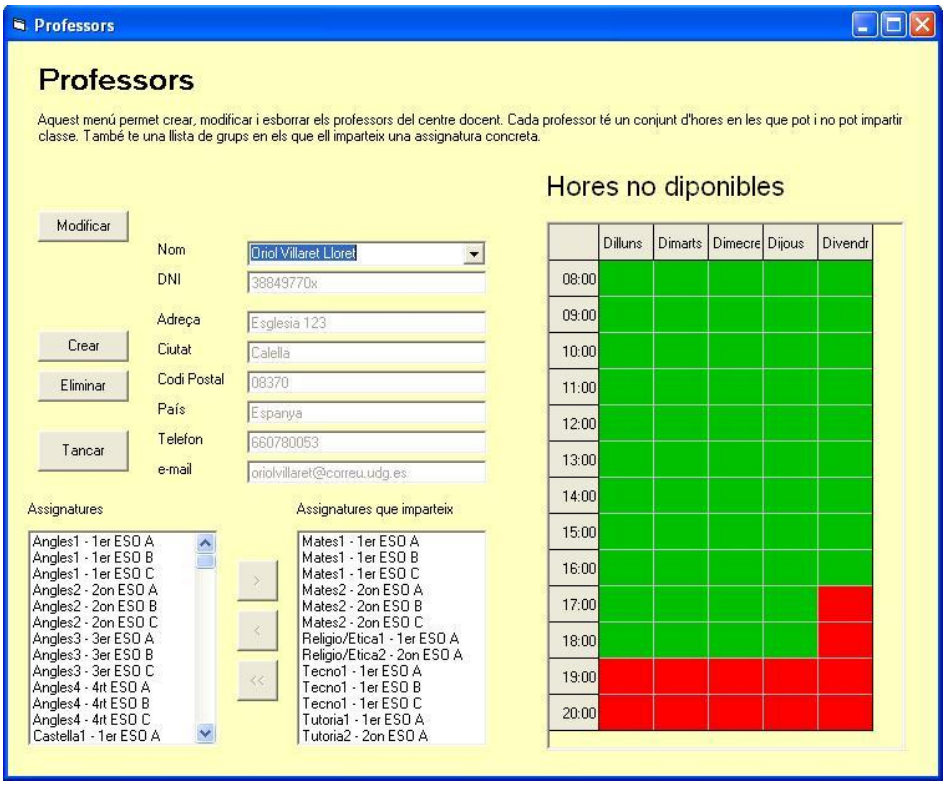
Seguint amb el tema de les assignatures, cursos i grups, arribem al menú d'Assignatures. Aquest menú disposa d'un desplegable amb totes les assignatures existents en la nostra base de dades. Al escollir-ne una, veiem que s'actualitzen els camps de durada, que indica el nombre d'hores que dura cada sessió d'aquesta assignatura, i el camp sessions, que indica el nombre de sessions que s'imparteixen d'aquesta assignatura cada setmana. També podem veure un apartat de "On s'imparteix". Com em vist al menú de "Grups", una assignatura pot ser cursada en tot un Curs (tots els grups d'un mateix curs) o en un o varis grups. Podem veure també que seleccionar tots els grups d'un mateix curs és exactament el mateix que seleccionar simplement el curs.



Les opcions Modificar i Crear, habiliten la opció de crear/modificar el nom del grup, la durada, el nombre de sessions i on s'imparteix una assignatura. Esborrar, no només esborra l'assignatura, sinó que esborra també les relacions que té aquesta amb els cursos i els grups on s'imparteix.

7.7. Professors

El menú de Professors és un dels menús que té més contingut de tots. També té un desplegable que conté tots els professors de la base de dades, i al seleccionar-ne un, s'actualitzen totes les seves dades: DNI, Adreça, Ciutat, Codi Postal, País, Telèfon, e-mail. Totes aquestes dades no són necessàries per a la generació de l'horari, però d'aquesta forma es pot portar un petit control dels professors i encarar el projecte a possibles ampliacions com la gestió del professorat.



Amés d'aquestes dades, però, també trobem dues llistes: "Assignatures" és una llista amb tuples Assignatura–Grup amb totes les assignatures que s'imparteixen a cada grup, i "Assignatures que imparteix" és una llista amb les mateixes tuples, però que indica quines assignatures imparteix aquest professor i a quins grups les imparteix.

Finalment, es pot veure una graella amb l'horari de disponibilitat del professor; en verd les hores disponibles i en vermell les no disponibles (les que no pot impartir classe).

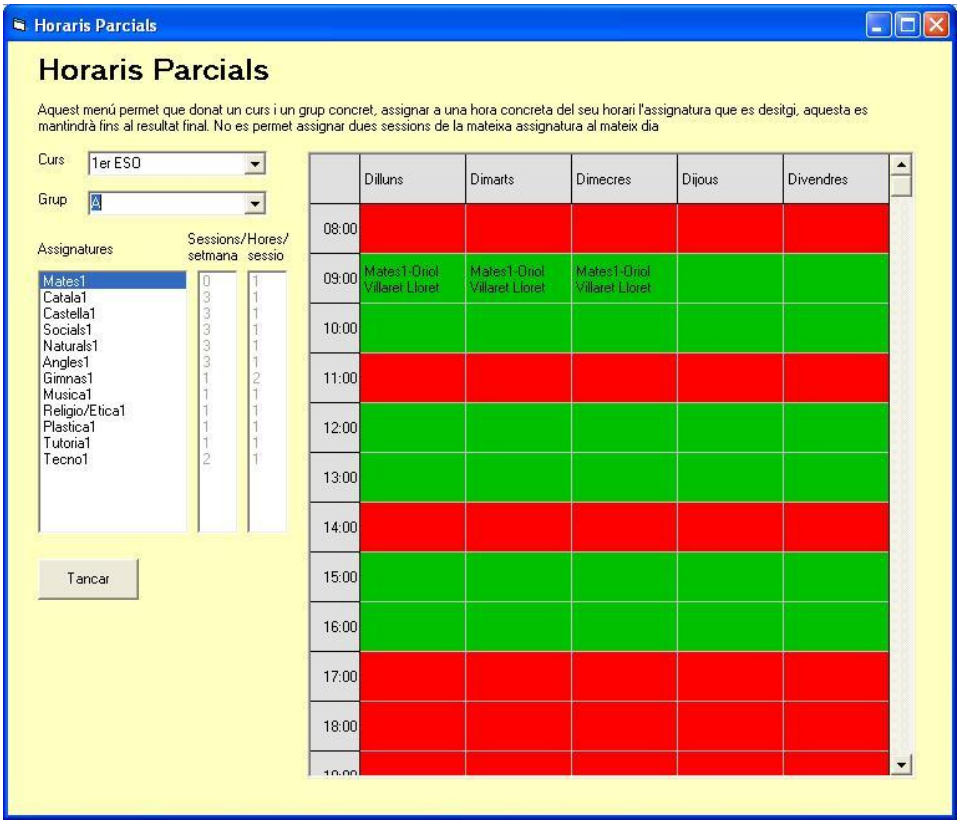
Les accions Modificar i Crear habiliten la modificació dels camps esmentats. Per modificar l'horari de disponibilitat del professor n'hi ha prou de

clicar sobre la casella de la graella de la qual volem modificar el color, també es pot clicar sobre una hora en concret i es modificarà el color de tots els dies en aquella hora i igualment podem clicar sobre un dia i totes les hores d'aquell dia es modificaran. Cal tenir en compte que només un professor podrà impartir una assignatura a un grup concret, així que si tenim un professor que imparteix una assignatura a un grup i li assignem aquesta mateixa tasca a un altre professor, el que ho feia anteriorment quedarà directament descartat.

En cap cas, l'eliminació d'un professor comportarà alguna modificació d'assignatures o de grups. Només s'eliminaran les dades del professor, el seu horari i les relacions que tenia amb les assignatures i grups. Així que si eliminem un professor que impartia una assignatura a un grup, aquest grup es quedarà sense professor d'aquesta assignatura i se n'hi haurà d'assignar un altre.

7.8. Horaris Parcials

Aquest menú és l'encarregat de fixar les hores en les quals es cursa una assignatura en un grup concret. Consta de dos desplegables, el primer conté una llista amb tots els cursos, i el segon s'habilita quan s'ha seleccionat un curs, i s'omple amb tots els grups corresponents a aquest curs. Al seleccionar un d'aquests grups, apareixen una llista d'Assignatures a l'apartat de "Assignatures", aquestes són les assignatures que s'imparteixen en aquest grup, hi ha tant les que s'imparteixen al seu curs com les que s'imparteixen en el grup en concret. Al costat de les assignatures, hi ha dues llistes que es corresponen amb aquesta, "Sessions per setmana" i "Hores per sessió". A cada assignatura li correspon un valor de cada una de les llistes.



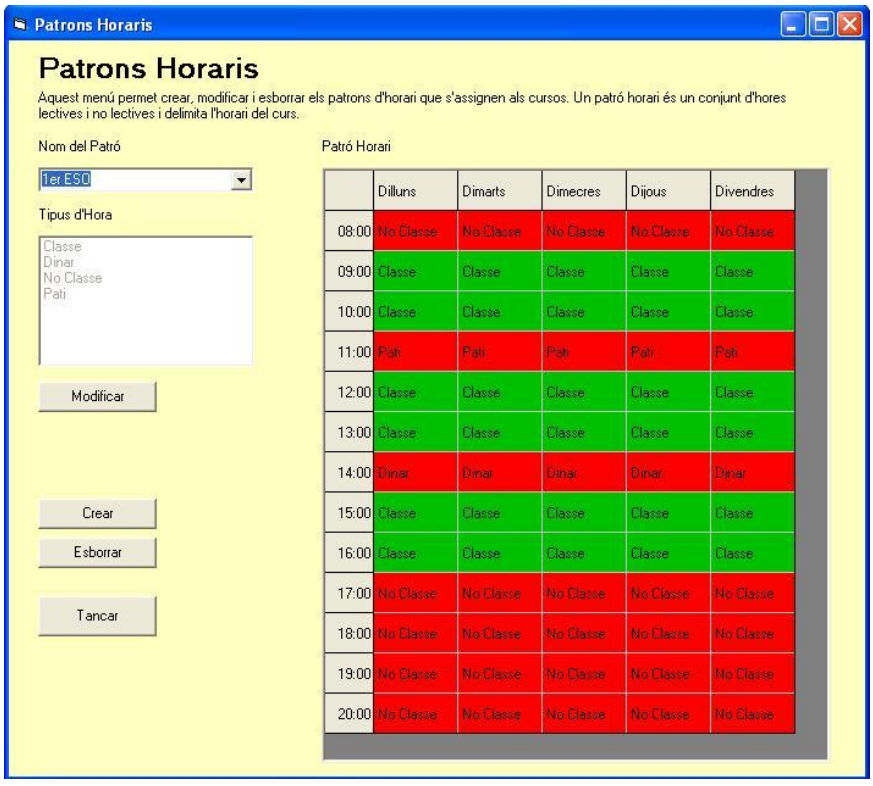
Al seleccionar una d'aquestes assignatures, podem "fixar-la" a l'horari que apareix a la part dreta del menú amb un doble-click. D'aquesta forma, s'assigna una hora a l'assignatura escollida que es mantindrà fins a aconseguir el resultat final. Cal destacar que no es pot fixar la mateixa assignatura dues vegades al mateix dia i que al afegir la segona

desapareixerà la primera. També es pot veure que a mesura que es fixen assignatures, el seu nombre de sessions va disminuint.

També es pot "desfixar" una assignatura tornant a fer doble-click sobre ella a l'horari, aquesta desapareixerà i el nombre de sessions tornarà a augmentar.

7.9. Patrons d'Horari

Els patrons d'Horari ja els hem esmentat anteriorment a l'apartat de "Cursos" i hem avançat que ens diuen les hores lectives i no lectives que té un curs. Aquest menú té un desplegable amb tots els patrons introduïts fins el moment. Al seleccionar-ne un es carrega a la graella de la part dreta el patró en qüestió. Aquest consta d'un conjunt d'hores lectives (en verd) i no lectives (en vermell). Més concretament, a les hores lectives les anomenarem de "Classe" i les no lectives poden ser de varis tipus, de "Pati", de "Dinar" o simplement de "No Classe". Podem crear, modificar o eliminar un Patró Horari (per als dos últims s'ha d'haver seleccionat anteriorment) i podem assignar els tipus d'hora, escollint primer a la llista de "Tipus d'Hora" i clicant després a la graella. Si se selecciona una posició concreta, el valor seleccionat de la llista s'atribuirà a aquesta posició de la graella, si es selecciona un dia, s'assignarà aquest valor a tota la columna i si se selecciona una hora, s'assignarà a tota la fila.

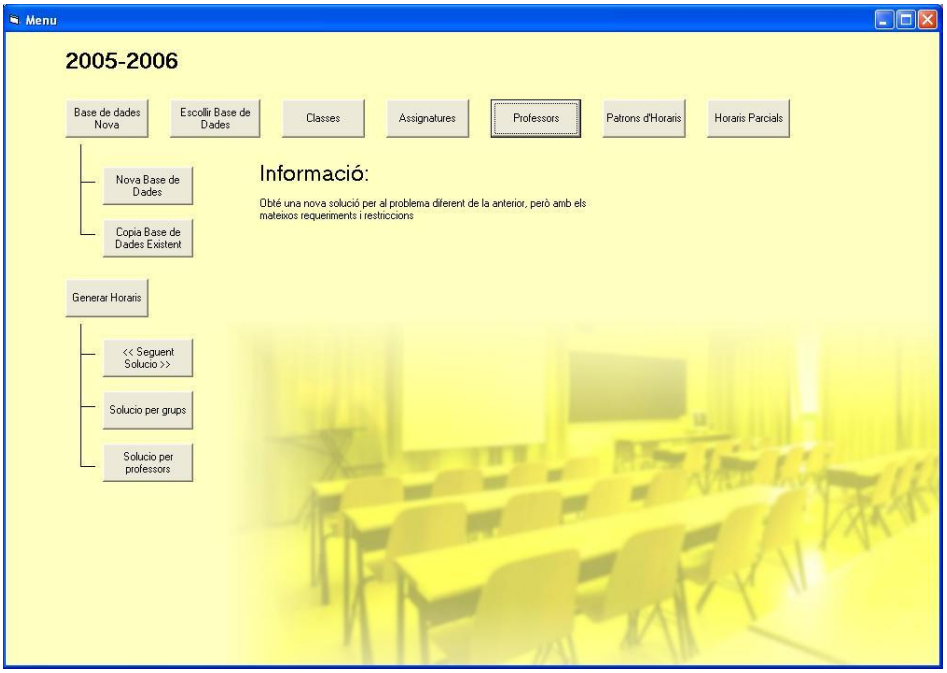


Un mateix Patró d'Horari pot estar assignat a varis cursos així com cada curs pot tenir un patró horari per a ell sol.

7.10. Generar Horaris

Aquesta opció del menú principal, no ens porta a cap altre menú, sinó que simplement ens mostra una nova opció, la de “Solucionar”. Amés, també crea un fitxer de fets per al Prolog a partir de totes les dades que hem anat elaborant. El més lògic potser hauria estat que només clicar aquesta opció, ja generés l’horari resultat directament, però cal tenir en compte que generar aquest resultat, en segons quins casos, pot ser extremadament costós i pot tardar molta estona.

Posem per cas, que l’usuari encara no ha acabat d’introduir les dades i sense voler-ho clica aquesta opció. En aquest cas no només es pot trobar en que l’aplicació pot arribar a tardar molta estona a tornar a estar altre cop disponible, sinó que l’horari resultant no li servirà de res, ja que encara no te totes les dades introduïdes. Si realment vol generar l’horari, només ha de clicar la opció de “Solucionar”.



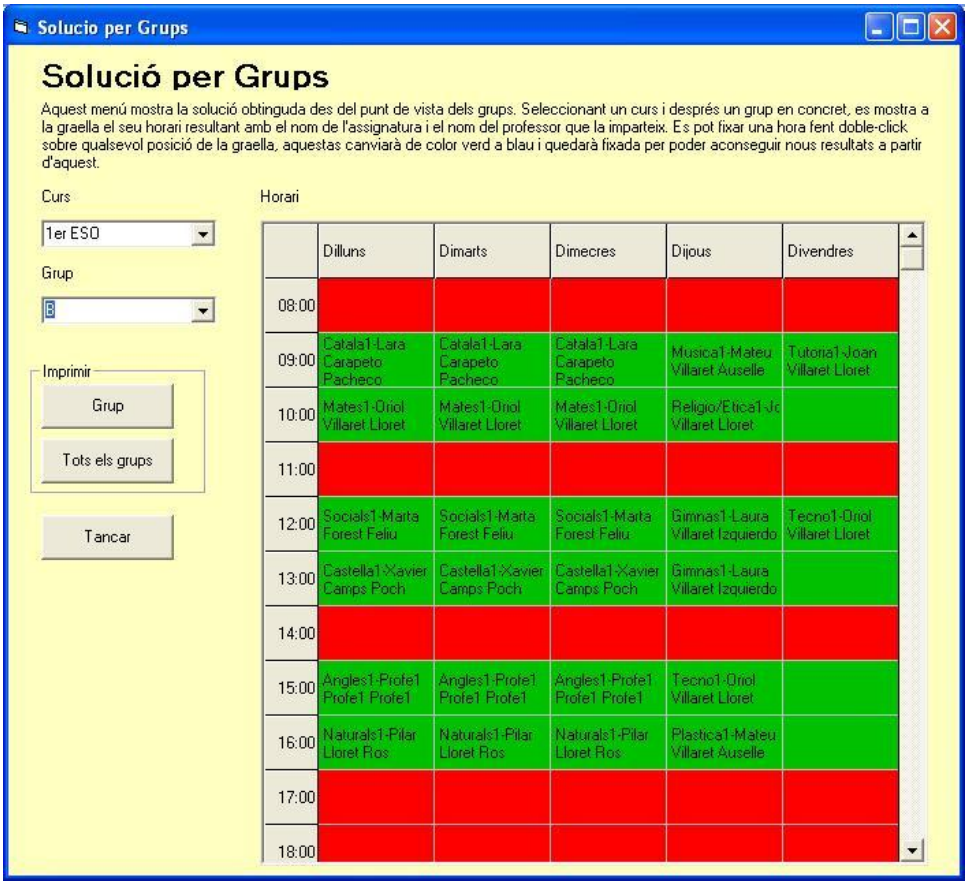
Aquesta opció crea un lligam amb el Prolog i li demana una solució per al problema que s’ha anat construint amb els altres menús. Un cop el Prolog retorna una solució aquest l’interpreta i la guarda a la Base de Dades per a poder mostrar-la a l’usuari de forma entenedora amb els menús “Solució per Grups” i “Solució per Professors” que hauran aparegut al menú.

Finalment també haurà aparegut la opció d'aconseguir una nova solució per al mateix problema si aquesta no ens acaba d'agradar. Aquesta opció es deshabilitarà tota sola si ens trobem a l'última solució possible. Si no hi hagués cap solució per al problema o si hi hagués algun problema amb el lligam Visual Basic – SICSTus Prolog, es mostraria el missatge corresponent a l'usuari i no s'habilitarien les demés opcions.

7.11. Resultat Per Grups

Aquest menú és bàsicament il·lustratiu. Podem seleccionar del desplegable “Curs” el curs desitjat i seguidament del desplegable “Grup” el Grup sobre el que volem veure el seu horari final.

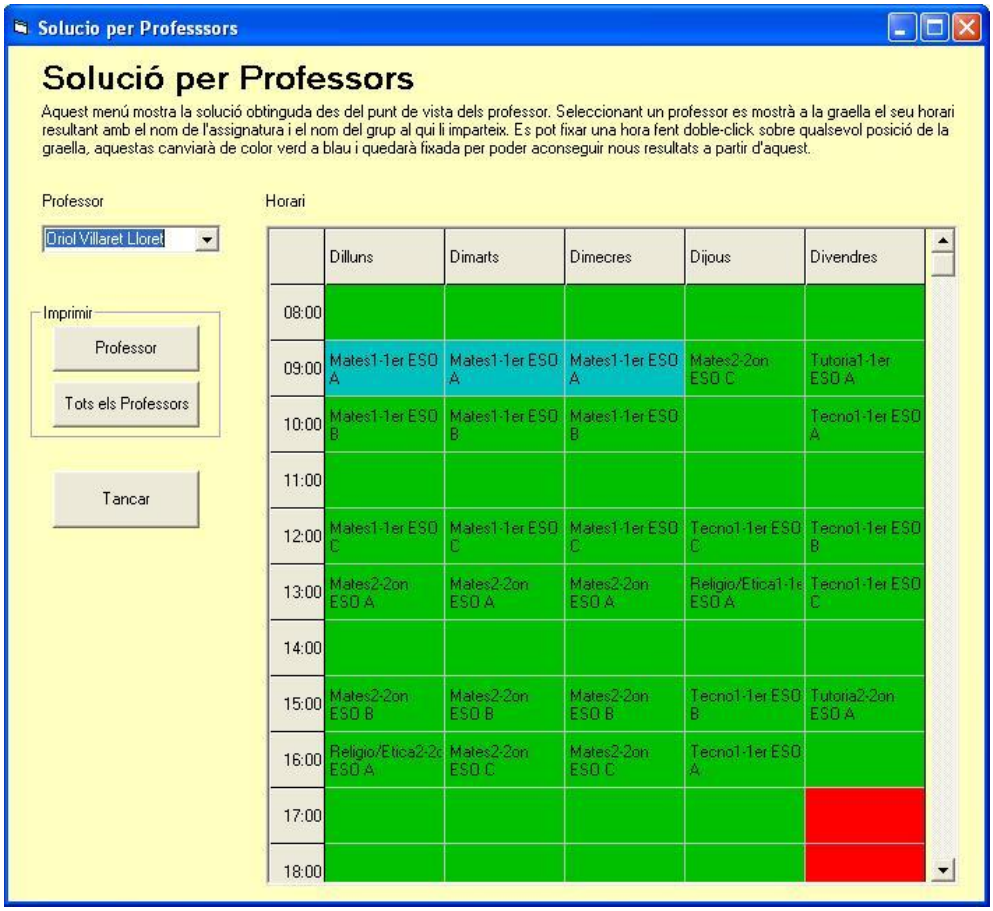
Al seleccionar el curs desitjat, es carregarà el seu patró horari, i al seleccionar el grup, s’omplirà aquest amb el resultat aconseguit. Arribats a aquest punt, ens pot interessar imprimir la solució , per fer-ho tenim la opció d’Imprimir tant per un sol grup com per tots els grups col·lectivament.



També ens pot interessar fixar part del resultat aconseguit per buscar més solucions a partir d’aquest. Per fer-ho no hem de retrocedir i entrar de nou a “Resultat Prefixat”, sinó que podem fer doble-clic sobre qualsevol posició de la graella i aquesta es “fixarà”. Podem fixar tantes caselles com vulguem i desfixar-les de la mateixa manera. Al fer-ho el color de fons canviarà de blau (fixada) a verd (desfixada). Les caselles vermelles (no lectives) no es poden fixar ja que pel fet de ser no lectives ja ho estan.

7.12. Resultat Per Professors

Com en el menú anterior, aquest és bàsicament il·lustratiu. Es pot seleccionar del desplegable “Professor” el professor del qual desitgem saber l’horari i visualitzar-lo a la graella de la part dreta. Podem imprimir també un professor sol com tots els professors col·lectivament amb els diferents botons d’impressió i podem fixar i desfixar hores fent doble-clic sobre les caselles que ho desitgem.



8. Posada en marxa

Per instal·lar l'aplicació, només s'ha d'executar el fitxer d'instal·lació "setup.exe" i seguir les instruccions d'instal·lació.

També ens hem d'assegurar que tenim les variables d'entorn inicialitzades correctament pel SICSTus Prolog. La variable "Path" ha de contenir el camí on es troba el fitxer vbsp.dll del SICSTus, normalment "C:\Archivos de programa\SICStus Prolog 3.12.2\bin" depenent de la versió de SICStus que tinguem.

L'aplicació s'ha d'instal·lar i executar sobre un entorn MS Windows i la màquina sobre la que s'executa ha de tenir instal·lat el SICStus Prolog.

9. Possibles ampliacions

Hi ha diferents punts que aquest projecte no abarca i que crec que, si més no, es mereixen ser esmentats en aquest apartat per a tenir-los en compte de cara a possibles ampliacions.

- Un punt fort sobre les possibles ampliacions que aquest projecte no contempla, ni ho pretén, és tot l'apartat de gestió d'aules. Normalment una assignatura s'ha d'impartir obligatòriament en un tipus d'aula concreta tant per qüestions d'espai com de material de la mateixa aula, ordinadors a les aules d'informàtica, taules de dibuix a les aules de plàstica o instruments a les aules de música. Aquestes aules són recursos limitats de l'escola i no sempre les utilitzen els mateixos professors ni els mateixos grups. La gestió d'aquests recursos, assignar professors, hores i grups, seria una interessant ampliació del projecte.

- Un altre punt important a tenir en compte per a una possible ampliació és la possibilitat que una mateixa assignatura sigui cursada per tots els grups alhora, d'aquesta manera s'aconseguiria una coordinació perfecte per a fer desdoblaments de grups, com sol ser el cas dels crèdits variables, o les assignatures comunes a Batxillerats com Català, Castellà o Anglès, on de tres grups se'n poden fer dos per estalviar-se un professor i una aula.
  Aquesta ampliació no seria molt complicada de cara al Prolog, tan sols s'hauria de definir l'assignatura corresponent de forma que se'n restringissin les hores en les que s'imparteix i així coincidissin per tots els grups. Segurament també s'hauria de modificar l'apartat de professors, ja que el nombre de grups canviaria momentàniament per a fer el desdoblament.

- Una altre possible ampliació és la de contemplar les “hores de centre” que ha de fer cada professor. Les hores de centre són un conjunt d’hores en les quals els professors han de romandre disponibles dins de l’escola per a possibles imprevistos, com per exemple que un professor es posi malalt i hagi de fer una substitució o simplement per vigilar l’esbarjo. Durant totes les hores lectives de la setmana, hi ha d’haver un mínim de professors un, dos o tres, que han de cobrir aquestes hores de centre per a possibles imprevistos de l’escola. Es podria definir un predicat en Prolog que assignés als professors aquestes hores de centre de forma equitativa entre tots ells i que retornés el resultat als horaris dels professors.

- Una alternativa que vaig desestimar després d’entrevistar-me amb el cap d’estudis d’Escoles Freta a Calella, era la possibilitat que en un grup concret, donada una assignatura, poguessin impartir-la diferents professors i que el programa decidís d’entre ells quin és el millor candidat. Em va comentar que era una proposta interessant, però que, normalment, ja contracten el professorat sabent quines assignatures impartirien i a on les impartirien. Tot i això algun cop s’han trobat amb aquesta possibilitat, llavors, simplement han escollit quin era el professor més adequat per aquell grup concret i aquella assignatura.

10. Conclusions

El resultat final és una aplicació que genera els horaris de tots els grups i professors d'un centre docent on l'usuari pot, en tot moment, decidir què vol canviar, on ho vol canviar i des d'on ho vol canviar, obtenint així una gran versatilitat.

No abarca el mateix ventall de possibilitats dels grans productes destinats a generar els mateixos horaris, però al estar enfocat de cara a un centre en concret, el converteix en una bona eina de suport sobre la qual treballar. Amés, l'aplicació queda oberta a possible ampliacions o modificacions, que al treballar amb un llenguatge declaratiu seran fàcilment realitzables.

Tot i que hi ha hagut altres projectes que resolien problemes semblants amb els horaris d'un centre docent: un resolia la problemàtica de les aules i un altre la gestió dels horaris de cara a l'alumnat, aquests es basaven en horaris ja confeccionats pel propi centre. Crec que la idea de realitzar un projecte que generi aquests horaris és original i que s'han complert tots els objectius que es pretenia.

Personalment, he après molt amb aquest projecte, sobretot programació lògica i resolució de constraints, però també he après molt sobre Visual Basic, recerca d'informació en fonts diverses i tot l'apartat de documentació d'un projecte.

11. Bibliografia

- SICStus Prolog User's Manual, Release 3.12.2
  Swedish Institute of Computer Science
  Maig 2005

- Introducció a la Programació Lògica
  Mateu Villaret
  Abril 2005

- Introducció als esquemes algoritmics: 5. Recorregut de Grafs
  Joan Surrell i Saurí
  2004-2005

- Aprenda Visual Basic 6.0 como si estuviera en primero
  Javier Garcia de Jatón, José Ignacio Rodríguez, Alfonso Brazález
  Agost 1999

- http://www.lawebdelprogramador.com

- http://www.programacion.com/foros/32/

12. Agraïments

M’agradaria donar les gràcies a tota la gent que m’ha ajudat durant la realització d’aquest projecte

A Escoles Freta i sobretot a en Manel Teodoro, cap d’estudis d’Escoles Freta de Calella per ensenyar-me com funciona una escola per dintre i orientar-me en el món de la docència sempre que li he demanat.

A en Mateu Villaret, director del projecte, per la seva paciència i per la seva ajuda i suport.

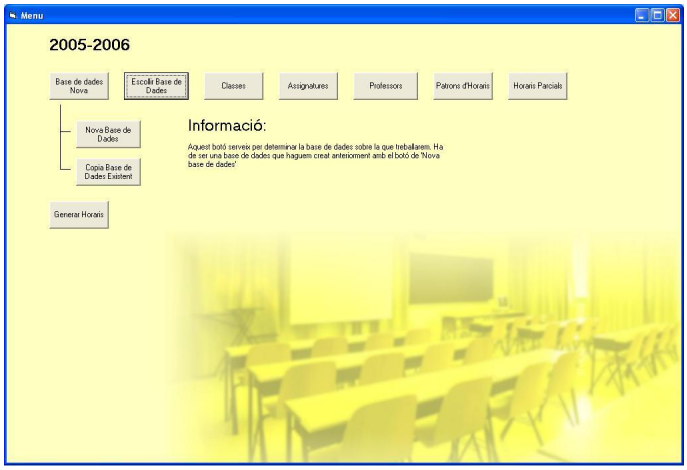
Donar les gràcies especialment a la Lara pel seu suport moral durant tot el projecte i bona part de la carrera.

Help - Menú

- Menú Principal
    1. Escollir base de dades
    2. Guardar una nova Base de Dades
    3. Classes
    4. Grups
    5. Assignatures
    6. Professors
    7. Horaris Parcials
    8. Patrons Horaris
    9. Generar Horaris
        - Resultat per Grup
        - Resultat per Professor

Menú Principal

Al iniciar l'aplicació, aquest és el primer menú que ens apareix. És sobre el que es basa tota l'aplicació, ja que des d'ell es pot accedir pràcticament a tot arreu. Inicialment però només dona a escollir dues de les opcions: Escollir Base de Dades i 'Base de Dades Nova'.

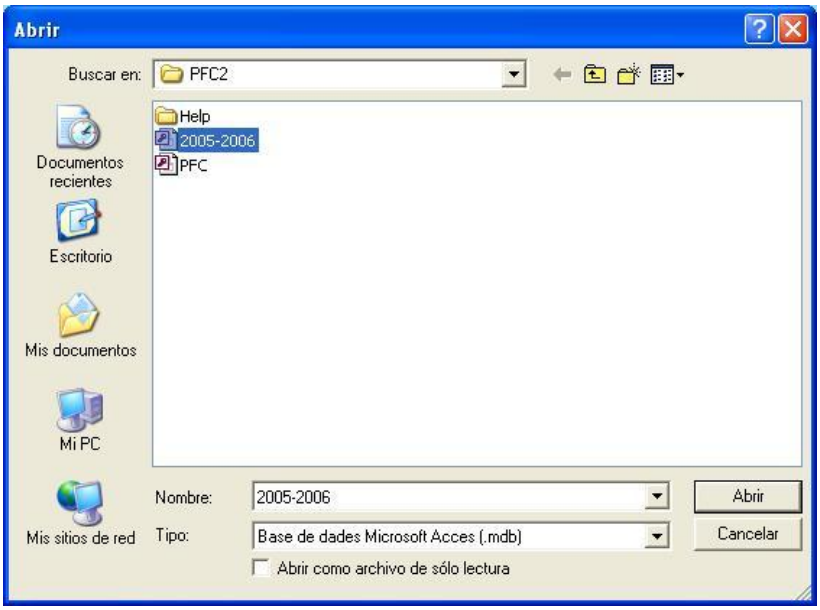


Les dues opcions acaben lligant la nostra aplicació amb una base de dades (buida o no) i només deixa escollir aquestes perquè per iniciar la nostra aplicació és essencial que estigui lligada a una base de dades sobre la que poder treballar. Un cop seleccionada la base de dades, ja podrem accedir a la nostra aplicació lliurement. Al apartat de Base de dades Nova, apareixen dues noves opcions: 'Copia Base de Dades Existent' i 'Nova Base de Dades'. Escollirem una opció o una altra depenent del que vulguem fer.

tornar

Escollir Base de Dades

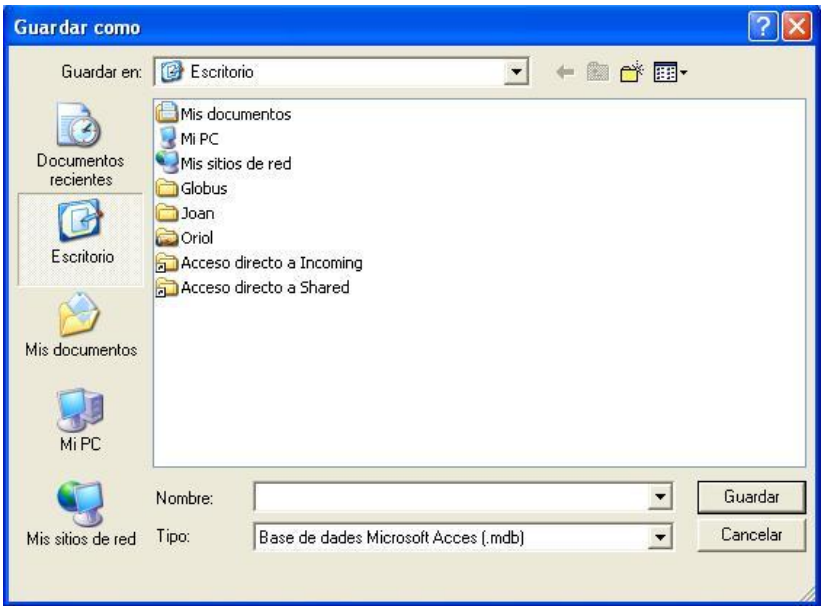
Tant per al menú Escollir Base de Dades com per el de Copiar Base de dades Existent, apareixerà un menú que ens obliga a seleccionar una base de dades ja existent, en un cas per obrir-la i en un altre per copiar-la. Per escollir una base de dades, és bàsic tenir-ne una de creada. Aquest menú només accedirà a bases de dades de MS Acces i només funcionarà amb les bases de dades creades a partir del model de l'aplicació. La seva utilització és prou coneguda per tothom; es busca la base de dades sobre la que volem treballar i s'apreta el botó 'Abrir' per tornar a l'aplicació i seguir treballant, en cas d'apretar 'cancelar', l'aplicació seguirà inhabilitada.



tornar

Guardar Base de Dades

A aquesta finestra només s'hi podrà accedir des de 'Nova base de Dades' i 'Copiar Base de Dades existent', totes dues del Menú principal. En el segon cas, haurem hagut de escollir anteriorment quina base de dades és la que volem copiar. Tant si la base de dades és nova com copiada, escollirem el nom amb el que la volem guardar i el lloc. Finalment, si tot ha anat correctament, el menú estarà tot operatiu per començar a treballar sobre la base de dades que acabem de guardar.

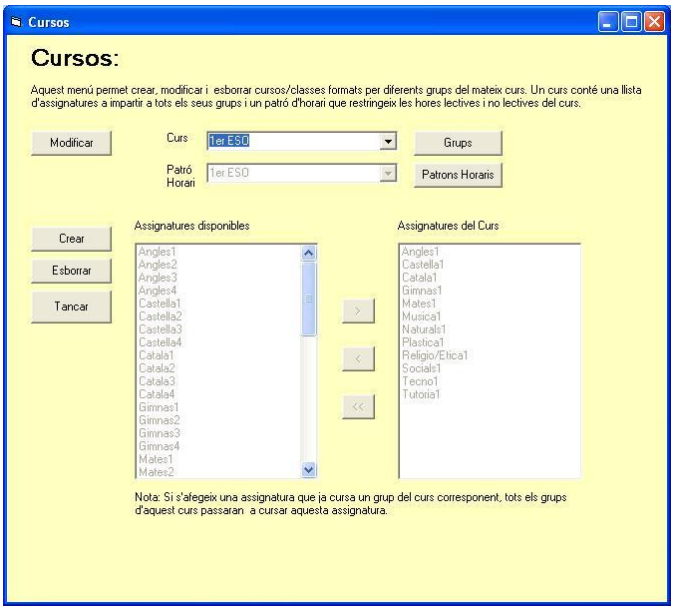


Es recomana utilitzar noms de fitxer de bases de dades intuïtius. Per exemple, si estem creant una nova base de dades per al curs 2005-2006, podem utilitzar noms com '2005-2006' o simplement '2005'

tornar

Classes o Cursos

El menú de Cursos és l'encarregat de crear, modificar i esborrar els Cursos de la base de dades. Podem veure que hi ha un desplegable amb tots els cursos ja introduïts anteriorment i que quan n'escollim un d'ells s'actualitzen totes les seves propietats a la resta del menú.



Aquestes propietats són:

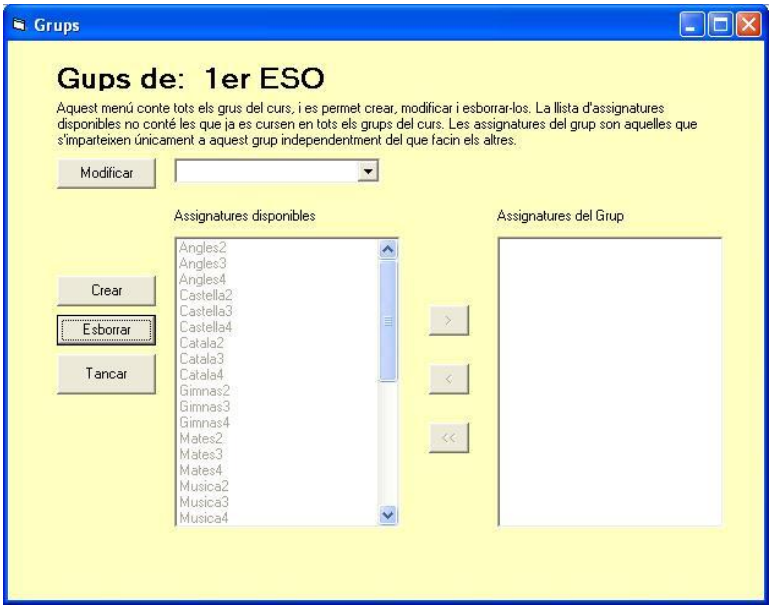
- Patró Horari: És el Patró de l'Horari que s'assigna a aquest curs, ja l'explicarem més detalladament en el seu apartat, només dir que serveix per delimitar les hores lectives i no lectives del curs.
- Assignatures disponibles: És una llista amb totes les assignatures existents.
- Assignatures del Curs: És una llista amb totes aquelles assignatures que estan assignades a aquest curs i per tant a tots els seus grups.

També podem veure en aquest menú tres opcions; una de crear, una de modificar i una altra d'esborrar. Les dues últimes requereixen haver seleccionat un curs del desplegable anteriorment. Les opcions de modificar i crear, habiliten conjunt d'accions d'aquest menú: escriure un nom (nou o no) del grup, modificar/escollir el patró que desitgem amb el desplegable 'Patró Horari' i finalment afegir a 'Assignatures del Curs' les assignatures que vulguem de la llista d'assignatures. També podem veure que han desaparegut o inhabilitat les altres accions i que per el contrari n'han aparegut unes de noves que són les d'Acceptar i Cancel·lar, que guarda els canvis realitzats a la base de dades en el cas d'Acceptar o les desestima en el cas de Cancel·lar. Finalment tenim també dos botons més; un de 'Grups' i un altre de 'Patrons Horaris' que ens envien directament als menús corresponents.

tornar

Grups

Per poder escollir aquest submenú, haurem hagut de seleccionar primer un curs . En aquest cas, es mostrarà un menú com aquest.

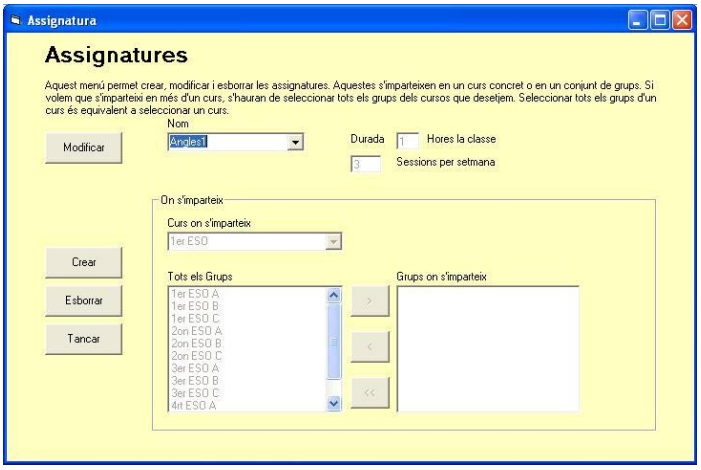


Podem veure que com a títol d'aquest submenú hi ha el nom del Curs pel qual hi hem accedit. També hi ha una pestanya desplegable que conté tots els grups ja existents. Per modificar o esborrar qualsevol grup, n'haurem d'haver seleccionat un. També podem veure una llista de les assignatures que se li poden assignar a aquest grup, ATENCIÓ: No totes, ja que aquest grup ja cursa unes assignatures al formar part d'un Curs . En aquest menú no fa falta cap condició per poder crear grups, només que n'indiquem el nom.

tornar

Assignatures

El menú Assignatures, consta de tres possibles opcions; modificar, esborrar i crear una assignatura. En els dos primers casos, modificar i esborrar, s'haurà d'haver seleccionat una assignatura amb anterioritat, en el cas de crear-la de nou, no farà falta. Una assignatura consta de tres atributs: Nom, Durada i nombre de sessions per setmana. Els noms de les diferents assignatures no poden ser repetits, així que si volem diferenciar Matemàtiques de primer d'E.S.O. de les de segon, haurem de diferenciar els seus noms, per exemple 'Matemàtiques 1' i 'Matemàtiques 2'

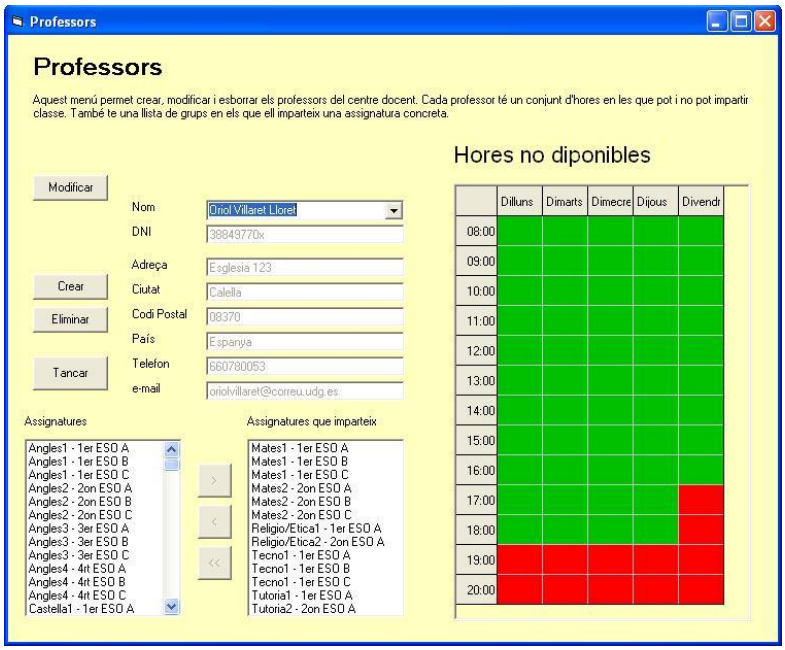


Finalment podem escollir a quin curs o grups es cursa aquesta assignatura. Si volem que l'assignatura es cursi a Primer d'E.S.O., seleccionarem el curs corresponent, però si per el contrari l'assignatura és exclusiva d'un o varis grups en concret, els seleccionarem individualment. Podem veure que és exactament el mateix seleccionar un curs sencer que tots els seus grups per separat, però per comoditat recomanem seleccionar directament el grup. També podem veure que no podem seleccionar cursos i grups alhora. Si es donés el cas, s'hauria de fer tota la selecció per grups. Per exemple, si a 'Primer d'E.S.O. A' es cursa una assignatura que també es cursa a tot segon d'E.S.O., seleccionarem els grups 'Primer d'E.S.O. A' i tots els grups de 'Segon d'E.S.O.'

tornar

Professors

El menú Professors, consta de tres possibles opcions; modificar, esborrar i crear una assignatura. En els dos primers casos, modificar i esborrar, s'haurà d'haver seleccionat un professor amb anterioritat, en el cas de crear-lo de nou, no farà falta. Podem veure que consta de molts camps amb les dades dels professors, aquests camps no son estrictament necessaris per a la gestió de l'horari, però recomanables per a la diferenciació i control dels professors al llarg dels anys.



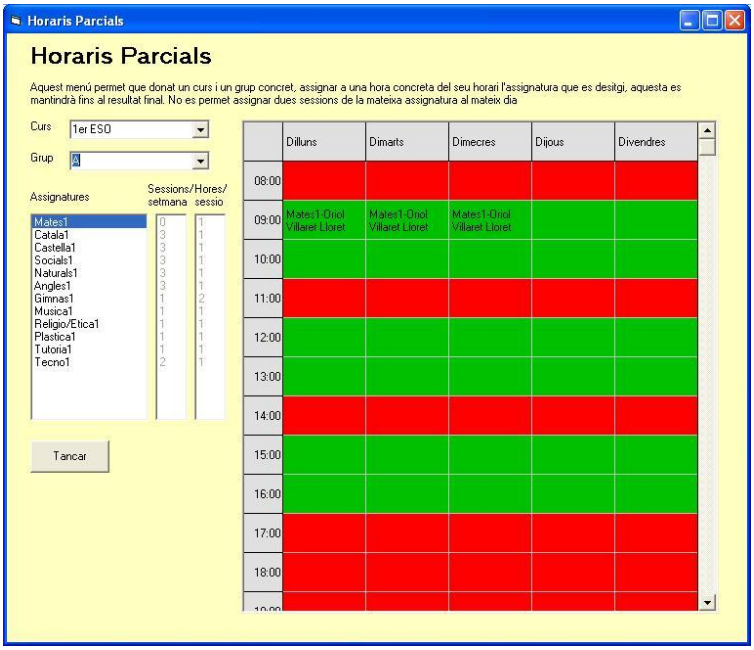
En el menú podem veure una part on es poden triar les assignatures que imparteix el professor i a quins grups ho fa. A diferència de l'apartat d'Assignatures, aquí no es pot seleccionar directament tot un curs s'han de seleccionar els grups per separat. MAI una mateixa assignatura podrà ser impartida a un mateix grup per diferents professors, Així que si diem que una assignatura és impartida a un grup per un professor i després diem el mateix per un altre, el primer deixarà d'impartir-la a l'instant.

Finalment, podem seleccionar el conjunt d'hores no disponibles del professor, que es marcaran en vermell. Si cliquem sobre el dia, es marcarà tot el dia i si marquem la hora, es marcarà aquesta hora de tots els dies.

tornar

Horaris Parcials

Aquest menú de l'aplicació és diferent dels altres, ja que no disposa de les opcions de crear, modificar i esborrar. La utilitat d'aquest menú és donar llibertat a l'usuari d'escollir i alhora minimitzar els càlculs, ja que l'usuari escollirà part de la solució.

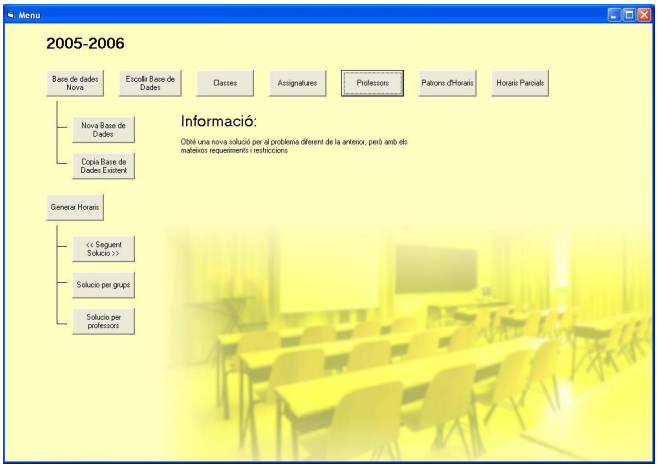


Per fer-ho hi ha una llista d'Assignatures que són les assignatures que s'imparteixen en aquest grup, hi ha tant les que s'imparteixen al seu curs com les que s'imparteixen en el grup en concret. Al costat de les assignatures, hi ha dues llistes que es corresponen amb aquesta, 'Sessions per setmana' i 'Hores per sessió'. A cada assignatura li correspon un valor de cada una de les llistes. Al seleccionar una d'aquestes assignatures, podem 'fixar-la' a l'horari que apareix a la part dreta del menú amb un doble-click. D'aquesta forma, s'assigna una hora a l'assignatura escollida que es mantindrà fins a aconseguir el resultat final. Cal destacar que no es pot fixar la mateixa assignatura dues vegades al mateix dia i que al afegir la segona desapareixerà la primera. També es pot veure que a mesura que es fixen assignatures, el seu nombre de sessions va disminuint. També es poden 'desfixar' una assignatura tornant a fer doble-click sobre ella a l'horari, desapareixerà i el nombre de sessions tornarà a augmentar.

tornar

Generar Horaris

Aquest opció del menú principal és una de les més simples i més importants d'aquesta aplicació. Inicialment només podrem escollir una opció, la de 'Solucionar'.

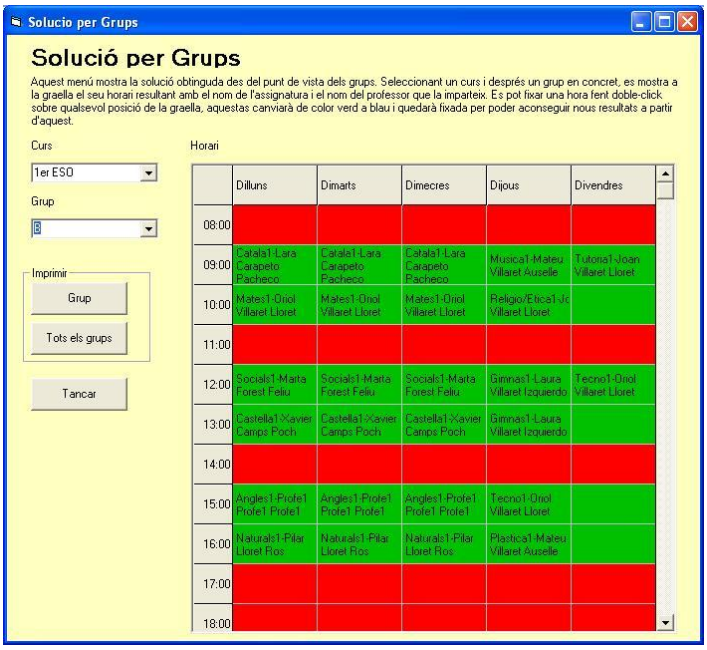


Aquesta opció crea un lligam amb el Prolog i li demana una solució per al problema que s'ha anat construint amb els altres menús. Un cop el Prolog retorna una solució aquest l'interpreta i la guarda a la Base de Dades per a poder mostrar-la a l'usuari de forma entenedora amb els menús 'Solució per Grups' i 'Solució per Professors' que ja hauran estat habilitats. Finalment tenim la opció d'aconseguir una nova solució per al mateix problema si aquesta no ens és del tot bona. Aquesta opció es deshabilitarà tota sola si ens trobem a la última solució possible. Si no hi hagués cap solució per al problema o si hi hagués algun problema amb el lligam Visual Basic - SICSTus Prolog, es mostraria el missatge corresponent a l'usuari i no s'habilitarien les demés opcions.

tornar

Resultat per Grups

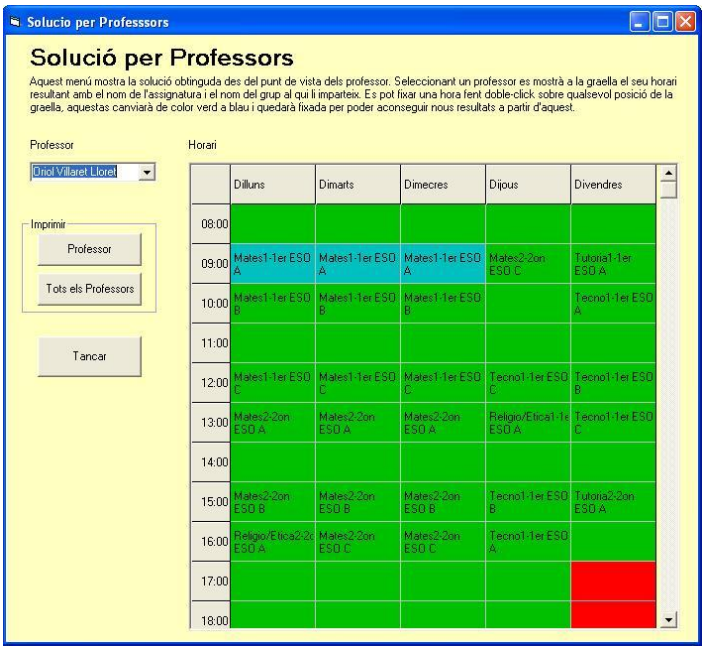
Aquest menú, igual que el menú Resultat per Professors, té la funció de mostrar el resultat obtingut a partir de les dades introduïdes. Amés, també ofereix la possibilitat de fixar hores de la solució per recalcular-la amb aquestes hores fixades fent doble-click sobre les hores que vulguem fixar, veurem que canvia a color blau. Aquest canvi afectarà també al menú d'horaris parcials i al menú de Resultat de Professors. En tots aquests menús podem desfixar les hores tornant a fer doble-click sobre l'hora a desfixar i veurem com torna al seu color original.



tornar

Resultat per Professors

Aquest menú té la funció de mostrar el resultat obtingut a partir de les dades introduïdes. Amés, també ofereix la possibilitat de fixar hores de la solució per recalcular-la amb aquestes hores fixades fent doble-click sobre les hores que vulguem fixar, veurem que canvia a color blau. Aquest canvi afectarà també al menú d'horaris parcials i al menú de Resultat de Grups. En tots aquests menús podem desfixar les hores tornant a fer doble-click sobre l'hora a desfixar i veurem com torna al seu color original.



tornar