Eina per a generar l'assignació docent d'un centre d'ensenyament

Índex:

1.- Introducció

Un dels problemes més habituals avui en dia i que suposa una despesa important en temps és el que consisteix en la planificació i assignació de tasques en base a un horari. En general, ens trobarem amb grups d'individus amb unes certes habilitats i limitacions que han de dur a terme unes funcions seguint unes pautes horàries. A més a més, ens podem trobar amb diferents casuístiques que impliquin un augment de la complexitat del problema, per la qual cosa arribarà un moment que convindrà automatitzar en la mesura del possible aquest procés, basant-nos en unes dades d'entrada i amb la major agilitat per a l'usuari possible. En el nostre cas concret ens hem basat en l'assignació de la docència als professors d'un centre docent, departament o àrea, tenint en compte una jerarquia assignatura – tipologia – grup per un costat (cada assignatura pot tenir fins a quatre tipologies diferents -teoria, practiques, laboratori i altres- i cada tipologia pot tenir il·limitats grups), un grup il·limitat de professors per un altre i diferents variables que associen els dos grups a diferents nivells.

Actualment la manera més habitual d'afrontar aquest problema es basa en reunions i negociacions que es poden eternitzar durant dies o setmanes, donat que qualsevol petit canvi implica una altra ronda de tempteig per poder quadrar els horaris de tots els professors. També es troben que han de tenir en compte reduccions de jornada, és a dir, menor nombre d'hores disponibles d'un professor o especialistes que no poden assistir al centre cada dia o només en un horari reduït (per exemple a les tardes).

2.- Motivació i Antecedents

L'elecció d'aquest projecte ha vingut donada per diferents elements:

- Existència d'una idea prèvia basada en l'observació de necessitats de l'entorn laboral, familiar i la possibilitat de resoldre un problema amb una complexitat important d'una manera eficient i elegant.
- Presència a l'oferta de projectes oferts pel centre, d'una forma genèrica (a falta de concretar algunes qüestions amb el/s tutor/s) del problema d'assignació d'horaris docents, en base a uns paràmetres bàsics.

Un cop tingut clar el títol i la base del projecte, s'ha hagut d'estudiar la problemàtica derivada del problema inicial, objectius a complir, elements a tenir en compte i possibles vies per solucionar-ho per poder afrontar l'obtenció de resultats en un temps raonable permetent i potenciant la facilitat d'ús de l'eina a desenvolupar.

3.- Objectius i observació de necessitats

L'objectiu principal d'aquest projecte és facilitar la tasca d'assignació de grups d'assignatures a professors, optimitzant el temps emprat en el mateix, basant-nos en un conjunt de restriccions, configurables lliurement per l'usuari final del programa.

Les restriccions que tindrem en compte a l'hora d'introduir les dades seran les següents:

- Interval de crèdits que pot impartir un professor (entre 0 i 50, encara que podria ser il·limitat).
- Capacitacions d'un professor davant d'una assignatura: si un professor pot impartir una assignatura o no.
- Compatibilitat horària: Possibilitat de tenir horaris disponibles diferenciats i personalitzats per cada professor de manera que un professor només pugui impartir un grup si està lliure en aquella franja horària.

Llavors, l'usuari haurà de poder:

- Donat un professor, assignar-li el seu horari de disponibilitat, diferent segons el dia o el quadrimestre.
- Donat un grup, assignar-li un horari, sigui en el primer, segon o ambdós quadrimestres.
- Assignar un nombre mínim i màxim de crèdits a impartir per part d'un professor.
- Assignar una llista de professors capacitats per impartir una assignatura.

Un cop introduïdes les dades, haurem de trobar la manera de calcular combinacions que compleixin amb les mateixes, utilitzant programari propi o de tercers.

Un altre objectiu a tenir en compte és la màxima compatibilitat, portabilitat i adaptabilitat del projecte, és a dir, que es pugui utilitzar en qualsevol sistema operatiu sense major problema i que es permeti adaptar o ampliar molt fàcilment les restriccions que s'hagin de tenir en compte. Això implicarà un disseny inicial més complex però que permetrà introduir futures modificacions segons necessitats d'una manera més ràpida i elegant.

Per un costat hem de tenir en compte les dades que haurem de guardar en una base de dades i les dades que simplement formaran part de l'execució del programa. D'aquesta manera, en un primer moment, veiem que necessitem tenir accessibles el llistat de professors, el llistat

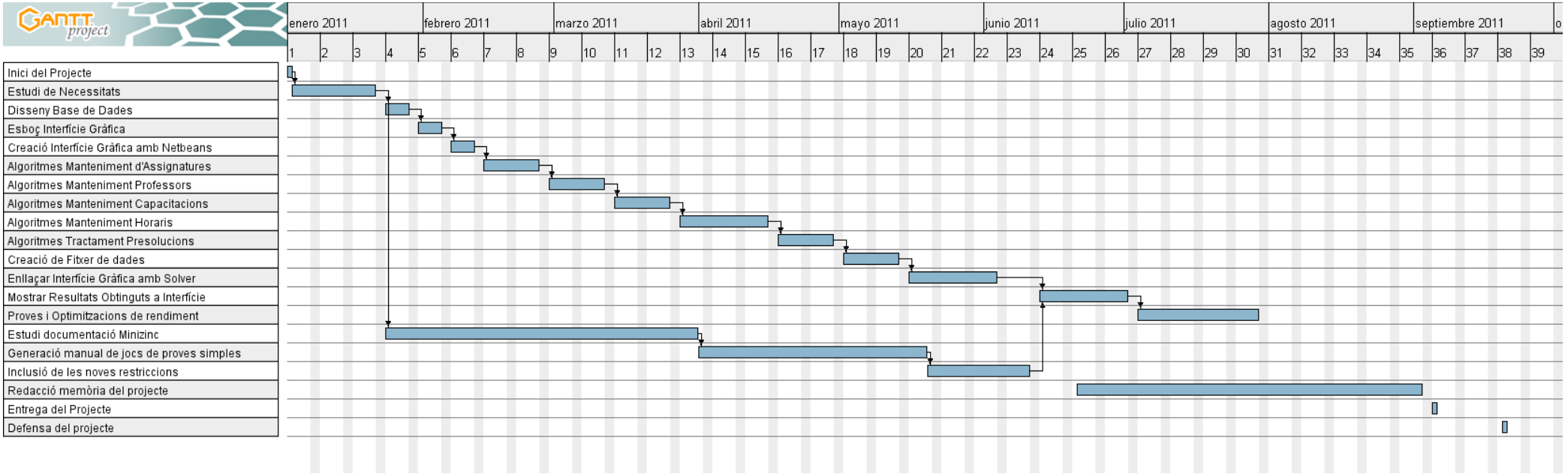
de grups, els horaris tant de professors com d'assignatures així com les capacitacions entre professors i grups. Hem de tenir en compte que aquest model de base de dades ha de ser fàcilment ampliable sense que això suposi un re-disseny complet de l'aplicació.

D'altra banda, necessitarem un resoledor que ens permeti, a partir de les dades d'entrada, trobar solucions a l'instancia del problema. S'haurà d'investigar diferents formes d'obtenir un resultat i avaluar-ne la idoneïtat i complexitat segons el volum de dades amb els que tinguem previst treballar.

A més a més, per complir amb els objectius anteriorment esmentats, a l'hora de triar eines a utilitzar, s'optarà per entorns multiplataforma tant per a la programació com per al sistema de base de dades.

4.- Fases i temporalització del projecte

El projecte està dissenyat per poder realitzar-se al llarg d'uns vuit mesos, i la distribució de tasques s'ha realitzat en base a aquesta temporalització. S'ha treballat sempre en dos flancs, per un costat l'interfície gràfica i per altra, la sintaxi i funcionament de diferents resoledors per poder fer les proves de rendiment adients. També s'ha hagut d'estudiar la connexió entre ambdós flancs per poder fer més atractiva i senzilla la utilització de l'aplicació de cara a l'usuari final.



5.- Estudi i Anàlisi

A l'hora d'afrontar el problema primer he hagut de determinar quines relacions hi havia entre les diferents variables que intervenen en algun moment. Un cop determinades aquestes relacions, s'ha hagut de dissenyar la base de dades, tenint en compte que possibles ampliacions futures no impliquessin un canvi important en la mateixa. A partir d'aquí hem de definir les diferents estructures de dades a utilitzar durant l'execució del programa, per poder manipular les dades d'una forma senzilla i el més eficient possible. Llavors s'ha de crear el model genèric de resolució del problema i el fitxer de dades relacionat, que serà generat de zero cada vegada que es modifiqui alguna dada. Amb el model genèric i el fitxer de dades, juntament amb un dels tres resoledors tinguts en compte, calcularem una possible solució a l'instància del problema. Amb la solució calculada interpretarem i mostrarem d'una manera simple i entenedora les assignacions que se li han adjudicat a cada professor.

5.1.- Perspectiva i decisions a l'hora d'afrontar el problema.

La primera decisió que havia de prendre era com interaccionarien les dades. Com que un dels objectius del projecte és la seva portabilitat, vaig optar per l'entorn MySQL, que disposa de connectors per la majoria de llenguatges de programació més habituals apart de facilitar-nos el creuament de dades mitjançant selects múltiples i joins.

Un cop decidit com treballarem amb les dades, faltava decidir quines dades era necessari guardar i quines es podien obtenir a partir de les primeres. A primer cop d'ull necessitava guardar dades sobre:

- Assignatures, característiques generals, tipologies derivades i grups derivats de les tipologies.
- Professors, dades generals, nom, cognoms
- Nombre de crèdits mínims i màxims d'un professor.
- Nombre de crèdits d'una assignatura, d'una tipologia i dels grups derivats de la mateixa.
- Capacitacions dels professors a l'hora de poder impartir una assignatura.
- Horaris, dels professors i de les assignatures.

Tal com s'explica més tard, ha estat necessari guardar informació sobre solucions preassignades (també anomenades presolucions).

Un cop creada l'estructura de base de dades adequada pel problema, calia decidir per un costat el llenguatge de programació a utilitzar en l'interfície gràfica i per altra, com o què faria les tasques de resolució un cop disposades totes les dades. En primer lloc, seguint amb el criteri de la portabilitat, vaig optar pel llenguatge Java, que proporcionava execució multi plataforma i al mateix temps facilitat a l'hora d'operar amb estructures de dades. En segon lloc, calia decidir quin tipus de programació volia seguir pel resoledor, fos intern o extern, i vaig estar sospesant entre tres opcions:

- Backtracking
- Algoritmes voraços
- Programació amb restriccions

En el següent apartat explicaré amb més deteniment els pros i contres de cadascun dels diferents algoritmes.

5.1.1.- Programació ad-hoc vs programació declarativa amb restriccions

Una de les decisions més importants a l'hora de triar el mètode de resolució del problema ha estat el tipus de programació que s'imposarà al resoledor. Teníem diverses opcions, començant per la disjuntiva entre programació imperativa i programació declarativa.

La programació imperativa es caracteritza per haver d'implementar tots els passos que necessita el resoledor per arribar a la solució final, actualitzant variables explícitament en cada pas si és necessari.

La programació declarativa amb restriccions es fonamenta en l'elaboració d'un model basat en restriccions que han de complir diferents variables per poder determinar la validesa d'un model i més tard utilitzar un resoledor per trobar la solució que compleixi les premisses del model.

Dins la programació declarativa amb restriccions hi ha diferents opcions, com el Minizinc o el Sictus Prolog.

Segons la tipologia de problema en el qual es basa aquest projecte, i seguint els consells dels directors del mateix, es va optar per la programació amb restriccions, pensant en la complexitat que comportaria un canvi en els requisits del problema, on amb una programació imperativa, podria suposar canvis importants. Dins la programació declarativa amb restriccions s'ha triat el Minizinc, bàsicament seguint criteris d'eficiència.

Agafant un exemple com pot ser la resolució d'un Sudoku, l'elecció d'un algoritme imperatiu (i dins aquest, diferents estratègies com el backtracking o algoritmes voraços) o declaratiu varia l'enfoc de la resolució del problema:

Problema inicial:
Tenim un tauler de 9x9 posicions amb x posicions ocupades i 81-x posicions lliures. A cada fila i cada columna del tauler hi han de ser presents els números de l'1 al 9 sense repetició dins la mateixa fila o columna. A més a més, dividirem el tauler de 9x9 en 9 taulers de 3x3 els quals hauran de contenir els nombres de l'1 al 9 sense repetició.

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Des d'un punt de vista imperatiu, el primer que hem de fer és calcular per cada casella desocupada, els possibles nombres que s'hi pot posar, per exemple, en la casella [0,2] hi podrien anar [1,2,4], a al casella [0,3] hi podrien anar [2,3], i així per totes les caselles buides.

Llavors, segons l'estratègia utilitzada a l'hora d'implementar l'algoritme imperatiu, s'afrontarà el problema d'una manera o d'una altra, si optem per utilitzar un mètode voraç, anirem emplenant els buits en base a algun criteri que triem i segons els possibles valors que pugui agafar aquella casella fins que completem el sudoku (obtenim una solució) o bé es deixen de complir les condicions definides inicialment (no es pot obtenir una solució per aquesta via), mentre que si optem per la via backtracking, tindrem l'opció de tirar enrere a cada pas per poder modificar el valor introduït a l'anterior buit per així intentar de trobar una solució satisfactòria.

En general, es pot tenir una visió del conjunt de solucions com les branques i fulles d'un arbre on, segons el mètode voraç comencem a la tija i acabem directament en un dels extrems podent trobar el que busquem o no, mentre que segons el mètode backtracking, anem recorrent les diferents parts de l'arbre fins que trobem el que busquem (o acabem les possibilitats).
Des del punt de vista declaratiu simplement hauríem de posar com a restriccions que tots els nombres de les files, columnes i quadrats 3x3 han de ser diferents entre ells i estar entre 1 i 9.

Exemples de resolucions de l'instància exemple del problema del sudoku segons la metodologia utilitzada:

Nota: Pel precàlcul de l'exemple he utilitzat una implementació pròpia de resolució del sudoku feta en c, i m'ha donat aquest resultat:

[x][y]=> candidats
[0][2]=> 1 2 4
[0][3]=> 2 6
[0][5]=> 2 4 6 8
[0][6]=> 1 4 8 9
[0][7]=> 1 2 4 9
[0][8]=> 2 8
[1][1]=> 2 4 7
[1][2]=> 2 4 7
[1][6]=> 3 4 7 8
[1][7]=> 2 3 4
[1][8]=> 2 4 7 8
[2][0]=> 1 2
[2][3]=> 2 3
[2][4]=> 3 4
[2][5]=> 2 4
[2][6]=> 1 3 4 5 7
[2][8]=> 2 4 7
[3][1]=> 1 2 5
[3][2]=> 1 2 5 9
[3][3]=> 5 7 9
[3][5]=> 1 4 7
[3][6]=> 4 5 7 9
[3][7]=> 2 4 5 9
[4][1]=> 2 5
[4][2]=> 2 5 6 9
[4][4]=> 5
[4][6]=> 5 7 9
[4][7]=> 2 5 9
[5][1]=> 1 5
[5][2]=> 1 3 5 9
[5][5]=> 1 4
[5][6]=> 4 5 8 9
[5][7]=> 4 5 9
[6][0]=> 1 3 9
[6][2]=> 1 3 4 5 7 9
[6][3]=> 3 5 7
[6][4]=> 3 5
[6][5]=> 7
[6][8]=> 4
[7][0]=> 2 3
[7][1]=> 2 7 8
[7][2]=> 2 3 7
[7][6]=> 3 6
[7][7]=> 3
[8][0]=> 1 2 3
[8][1]=> 1 2 4 5
[8][2]=> 1 2 3 4 5
[8][3]=> 2 3 5 6
[8][5]=> 2 6
[8][6]=> 1 3 4 6

Mètode Voraç:

1. Precàlcul de candidats per cada buit.
2. Omplir el primer buit que es troba amb un dels nombres candidats.
3. Actualitza la llista de candidats de la fila, columna i quadrat 3x3 eliminant si cal el nombre que acabem d'introduir.
4. Omplir el següent buit amb un dels nombres candidats.
5. Anar repetint els passos 3 i 4 fins que:
   - No quedin buits en el tauler (s'ha trobat una solució satisfactòria)
   - No hi hagi candidats (per aquest camí no hi ha solució).

Es podria optimitzar agafant sempre el buit amb menys candidats possibles.

Aquest mètode és molt ràpid però té uns quants problemes derivats:

- No ens assegura trobar solucions en cas que n'hi hagi, per la qual cosa hauríem d'anar repetint l'algoritme des de zero fins trobar una solució.
- No ens proporciona el llistat complet de solucions a menys que executem totes les possibles combinacions de candidats, per la qual cosa perdríem l'avantatge de la velocitat.

Exemple pseudocodi:

```
preCalculCandidats();
mentre (nombreBuits()>0 o casella.nombreCandidats()=0) fer
        buscaCasellaAmbMenysCandidats();
        anotaCandidatdeCasella(); // aquí es podria discutir quin candidat anotar en cas que n'hi hagués més d'un.
        actualitzaCandidatsFilesColumnesiQuadrat();
        decrementaBuits();
fmentre
si (nombreBuits()=0) fer
        escriureSolucio();
altrament
        escriureNoEsTrobaSolucio();
        //possible repetir el procés triant altres candidats, però des de l'arrel, no des de l'última branca com fa el backtracking.
fsi
```

Exemple evolució sudoku segons mètode voraç

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Primer candidat es tria primera opció ([0][2]=> 1 2 4), la qual cosa comporta que s'hagin de re-calcular els candidats de les posicions buides que resten.

5	3	1		7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Llavors toca continuar amb el següent, que té com a candidats ([0][3]=> 2 6)

5	3	1	2	7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Anar repetint el procés fins que ens trobem una posició sense candidats o bé omplim tot el sudoku (trobem una solució):

- [0][5]=> 4 6 8

5	3	1	2	7	4			
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Un cop aquí ens n'adonem que a la posició [2][5] ja no hi ha candidats, per tant tal com està actualment el sudoku no es pot resoldre.

Al ser un algoritme voraç, hauríem de donar aquest resultat per bo o bé, tornar a fer un altre intent des del començament utilitzant alguna altra estratègia (com per exemple utilitzar sempre el buit amb menys candidats possibles).

Un dels grans avantatges dels algoritmes voraços és la seva velocitat, però com a contrapartida, ens trobem que no ens asseguren solucions en cas que n'hi hagi (de fet si no en trobem cap no podem saber si realment n'hi ha alguna, mentre que si en trobem una, no podem determinar si és única, si n'hi ha més o més òptimes).

Mètode Backtracking:

1. Precàlcul de candidats per cada buit.
2. Anotar candidat en el primer buit que es troba.
3. Actualitzar llista de candidats de la fila, columna i quadrat 3x3 eliminant si cal el nombre que acabem d'introduir.
4. Crida recursiva tornant a punt 2
5. Si al final de les crides recursives es troba una solució satisfactòria, es tracta, en cas contrari, es desanota el candidat i es prova amb el següent. En cas d'acabar-se els candidats, es torna un pas més enrere a l'anterior crida recursiva.

Aquest mètode és més lent que l'anterior però té diferents avantatges:

- Si existeix alguna solució la trobarem.
- Es pot trobar una solució concreta (la més òptima, la pitjor, la que compleixi certes condicions..)
- Es pot aconseguir el llistat complet de solucions fent mínims canvis a l'algoritme.
- Es poden aplicar optimitzacions per 'podar' les branques i així no haver de continuar per camins que no condueixin a solucions.

Exemple pseudocodi:

```
preCalculCandidats();
mentre (nombreBuits()>0) {
        anotaCandidat();
        cridaRecursiva(solucioParcial);
        desanotaCandidat();
}


cridaRecursiva(solucioParcial) {
        decrementaBuits();
        actualitzaCandidats();
        comprovaSolucio(); // si hi ha solució, mostrar-la, si no continuar anotant candidats, o “baixant nivells a l'arbre”
        si no hi ha solucio
                anotaCandidat();
                cridaRecursiva(solucioParcial);
                desanotaCandidat();
                incrementaBuits();
        fsi
}
```

Exemple evolució sudoku segons mètode backtracking

En la resolució pel mètode de backtracking (o tornada enrera), tot i que el desenvolupament inicial pot ser semblant al mètode voraç, hi ha un canvi molt important, que és el comportament davant un buit sense candidats; en comptes de donar per acabada l'execució, es torna enrere a la penúltima casella que havia deixat d'estar buida i s'hi el candidat que s'hi havia assignat per un altre de la seva llista de candidats. En cas de no tenir més candidats, repetiríem el procés anant a modificar una casella anterior.

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Primer candidat es tria primera opció ([0][2]=> 1 2 4), la qual cosa comporta que s'hagin de recalcular els candidats de les posicions buides que resten.

5	3	1		7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Llavors toca continuar amb el següent, que té com a candidats ([0][3]=> 2 6)

5	3	1	2	7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Anar repetint el procés fins que ens trobem una posició sense candidats o bé omplim tot el sudoku (trobem una solució):

- [0][5]=> 4 6 8

5	3	1	2	7	4			
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

En aquest punt, retrocediríem a [0][3] i en comptes de triar el 2, triaríem el 6:

5	3	1	6	7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Òbviament ara els candidats han variat, per la qual cosa pel buit [0][5] hi podria haver 2, 4 i

8.

Agafant el 2:

5	3	1	6	7	2			
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Llavors al buit de la posició [0][6] hi ha com a candidats 4, 8 i 9.

5	3	1	6	7	2	4		
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Anem repetint el procés fins el resultat desitjat, que pot ser:

- No es troba solució → Podem afirmar que l'instància del problema no té cap solució (en aquest cas ho podem fer amb total seguretat donat que, contràriament al mètode voraç, amb backtracking s'explora l'arbre sencer o fins a trobar una solució).
- Es troba una solució → Segons com implementem l'algoritme, podem indicar que, en el moment de trobar una solució no en busqui cap més, donant per acabada l'execució del mateix.
- Es volen buscar totes les solucions → Complementant l'anterior punt, si volem disposar de l'univers complet de solucions a l'instància del problema, hem de tenir en compte el fet de guardar cada solució trobada, tot continuant amb l'execució fins a l'exploració completa de l'arbre de solucions.

- De totes les solucions, l'òptima → Aquesta opció és una extensió de l'anterior, on es dóna un pes o una puntuació a cada solució trobada fins al final, on es retorna la que més s'escau als nostres interessos. En el cas concret del sudoku és difícil trobar una solució òptima no arbitrària més enllà de buscar patrons entre caselles o files i columnes (per exemple que hi hagi el mínim nombre de canvis de posició entre un quadrat 3x3 i un altre, o entre una fila i una altra). Tanmateix, en altres casos com la cerca de camins mínims/més curts i derivats, ens servirà per obtenir resultats el màxim d'eficients possibles. Utilitzant aquesta estratègia, també tindrem l'avantatge de poder descartar execucions en un moment donat si els valors que té en aquell moment no poden millorar el valor òptim, per la qual cosa en alguns casos podrem reduir de forma significativa el temps d'execució.

Mètode de programació amb restriccions declaratiu

1. Elaboració del model de restriccions.
2. Introducció de les dades a tractar.
3. Tractament de solucions.

La diferència principal d'aquest mètode és que no intervenim en el funcionament intern del mateix, simplement li diem les condicions que ha de complir la solució i les dades d'entrada. D'aquesta manera ens alliberem de la tasca d'implementar els algoritmes de cerca de manera que només hem de tenir en compte els requisits de la modelització del problema a resoldre, és a dir, en la correcta i completa especificació del mateix.

Els resoledors que interpreten i computen el model ja tenen implementats diferents algoritmes de backtracking i de manteniment de consistència dels dominis de les variables que intervenen en la solució i que permeten efectuar les operacions necessàries per intentar trobar solucions que compleixin el model.

Exemple pseudocodi:

```
restricció per cada fila, tots els nombres diferents i entre 1 i 9
restricció per cada columna, tots els nombres diferents i entre 1 i 9
restricció per cada quadre 3x3, tots els nombres diferents i entre 1 i 9
resol
```

No ens preocupem en cap moment de donar-li ordres específiques del que ha de fer a cada pas, si ha de modificar una variable, cridar a una funció, intercanviar dades, simplement li diem quines condicions ha de complir la solució i li indiquem que intenti resoldre-ho.

També s'ha de tenir en compte que es pot triar el criteri de resolució, si simplement es vol obtenir un resultat (o tots), si es vol maximitzar/minimitzar el valor d'una variable o si es vol una solució tal que el valor d'una variable sigui un en concret.

Observant els pros i contres dels diferents mètodes, i seguint el consell dels directors del projecte, vaig optar per l'opció de treballar amb la programació amb restriccions, per diferents motius, entre els quals hi havia:

- És una base molt potent a partir de la qual es vertebra la resolució d'un problema.
- No ens basem tant en el com si no el què volem obtenir.
- A partir d'una base genèrica generada des del programa, podem utilitzar diferents

resoledors per obtenir una solució, segons conveniència i sense haver de modificar el fitxer origen.

- Personalment no havia treballat mai amb programació amb restriccions, per la qual cosa m'era molt interessant explorar aquesta nova via o alternativa a l'hora d'afrontar la resolució d'uns certs perfils de problema.
- L'existència de projectes d'investigació a diferents universitats.

5.1.2.- Minizinc

Un cop decidida la via declarativa per resoldre les instàncies del problema, era el moment de triar quina base agafàvem a l'hora de crear el model a resoldre; a congruència dels directors del projecte, es va optar pel minizinc, donat que era una via on s'estava efectuant investigació en el seu departament.

La resolució de problemes combinacionals és una tasca amb una complexitat molt important, on s'ha de formular d'una manera molt precisa l'abast i tècniques de resolució dels mateixos, la qual cosa esdevé molt complicat i costós en temps. Per aconseguir eficiència s'ha d'explorar detingudament el problema i experimentar amb diferents tècniques a fi i efecte de determinar quina és més adient. A partir de l'estudi general dels resultats anteriors, es va anar enfocant la resolució en la divisió dels problemes en dos passos més simples. Per un costat tenim el model conceptual del problema, sense indicar en cap moment com resoldre'l i per altre, el resoledor que, a partir del model conceptual, aplicarà els algoritmes corresponents per obtenir solucions. El més important d'aquest disseny en dues fases és que a partir del mateix model conceptual genèric, podem utilitzar diferents resoledors sense haver de fer canvis, per la qual cosa el testeig de rendiment i idoneïtat és òptim. D'aquesta idea va sorgir un nou estàndard (o llenguatge) de modelatge, anomenat Zinc, preparat per complir les premisses de genericitat de model anteriorment esmentades. Existeixen altres estàndards de modelatge com poden ser AMPL, Localizer, OPL i altres que incloguin ESRA i ESSENCE. Fins que va aparèixer Zinc, cada estàndard de modelatge estava basat o optimitzat en tècniques específiques segons el perfil de problema que s'afrontava; teníem alguns preparats per cerques com per exemple Localizer; altres com per exemple AMPL treballava amb MIP (model·latge matemàtic) o ESSENCE i ESRA, que eren especialment per programació amb constants (CP). Llavors l'especificació del Zinc ens proporciona les eines per poder desenvolupar models a alt nivell per poder utilitzar-los amb el resoledor que desitgem. Per aquests models disposarem d'estructures de dades, llistes, funcions i predicats definits per l'usuari a partir dels quals construirem les restriccions necessàries per a l'obtenció de les solucions buscades.

En general, el Zinc ens proporciona:

- Suport per a diferents tipus de problemes (satisfer condicions, optimització, preferència - restriccions toves-).
- Notació i sintaxi matemàtica (iteracions, sobrecàrrega, llistes, conjunts, matrius).
- Limitacions per domini de la variable o estructura de dades.
- Separació de les dades del model.

- Alt nivell d'encapsulació i estructura de dades (conjunts, matrius, tuples, registres, enumeracions).
- Extensibilitat (l'usuari pot definir funcions i predicats propis).
- Fiabilitat (comprovació de tipus i d'instàncies, control de condicions -asserts-).
- Possibilitat d'utilitzar diferents mètodes de resolució/resoledors per un mateix model.
- Semàntica molt simple.

A partir del Zinc, es crea el MiniZinc, que proporciona unes eines bàsiques de modelatge sent més simple d'implementar. Bàsicament es tracta d'un llenguatge de nivell mitjà de primer ordre, és a dir, que utilitza predicats i funcions l'argument de les quals són constants o variables d'individu. Respecte al Zinc, les capacitats que es perden són:

- Algunes restriccions automàtiques
- Restriccions toves
- Arrays relacionals (amb índex no enters)
- Funcions definides per l'usuari
- Algunes comprovacions prèvies (assertions)

Exemple de fitxer Zinc [http://www.hakank.org/minizinc/send_more_money2.zinc]:

```
%
% Send more money problem in Zzinc.
%
% Send More Money
%    SEND
%  + MORE
%  ------
%   MONEY
%
% This Zinc model was written by Hakan Kjellerstrand.
% See also my MiniZinc page: http://www.hakank.org/minizinc
%

include "globals.zinc";
enum digits = {S,E,N,D,M,O,R,Y};
array[digits] of var 0..9: x;

constraint alldifferent(x) /\ 1000*x[S] + 100*x[E] + 10*x[N] + x[D]  +  1000*x[M] + 100*x[O] + 10*x[R]
+ x[E]  = 10000*x[M] + 1000*x[O] + 100*x[N] + 10*x[E] + x[Y]  /\ x[S] > 0 /\ x[M] > 0;

solve satisfy;
```

```
output [ "x: ", show(x), "\n",  "\n", "     ", show(x[S]), show(x[E]), show(x[N]), show(x[D]),"\n", " +  ", show(x[M]), show(x[O]), show(x[R]), show(x[E]),"\n"," = ",  show(x[M]), show(x[O]), show(x[N]), show(x[E]), show(x[Y]), "\n"];
```

Versió MiniZinc [http://www.hakank.org/minizinc/send_more_money.mzn]:

```
%
% Send more money problem. in Minizinc.
%
% This Minizinc program was written by Hakan Kjellerstrand and is commented in
% Constraint Programming: Minizinc, Gecode/flatzinc och ECLiPSe/minizinc
% http://www.hakank.org/webblogg/archives/001209.html
%
%
% Send More Money
%    SEND
%  + MORE
%  ------
%   MONEY
%
% Implemented in a very straightforward way.
%
% See also my MiniZinc page: http://www.hakank.org/minizinc
%

include "globals.mzn";
var 0..9: S;
var 0..9: E;
var 0..9: N;
var 0..9: D;
var 0..9: M;
var 0..9: O;
var 0..9: R;
var 0..9: Y;

array[1..8] of var int : fd =   [S,E,N,D,M,O,R,Y];

constraint  all_different(fd) /\ 1000*S + 100*E + 10*N + D  +  1000*M + 100*O + 10*R + E  =
10000*M + 1000*O + 100*N + 10*E + Y  /\  S > 0 /\  M > 0;

solve satisfy;

output [ %    show(fd), "\n",     "S:", show(S), " E:", show(E), " N:", show(N), " D:", show(D),      " M:", show(M), " O:", show(O), " R:", show(R), " Y:", show(Y),     "\n\n",    "     ", show(S), show(E), show(N), show(D),"\n",    " +  ", show(M), show(O), show(R), show(E),"\n",   " = ",  show(M), show(O), show(N), show(E), show(Y), "\n" ];
```

Un cop tenim el problema especificat, la qual cosa es realitza en dues parts:

- model: és la part principal de l'especificació del problema, i ens indica l'estructura del mateix, restriccions que haurà de complir i estratègia de resolució a seguir.
- dades: contindrà les dades d'entrada de l'instància del problema.

A partir del model en minizinc el que fem és traduir el model i dades d'entrada a un fitxer flatzinc, el qual és un altre llenguatge d'especificació de problemes amb la particularitat que no està lligat a un resoledor si no que podem triar-ne un qualsevol (és independent del resoledor). Un cop tenim el fitxer del flatzinc, és el moment de computar-lo amb el resoledor que ens interessi per obtenir algun resultat (amb solució o sense).

En aquest projecte s'ha treballat amb tres resoledors, encara que està obert a l'inclusió d'altres opcions a fi d'avaluar comportaments i eficiència. Aquests resoledors són:

- G12
- Gecode
- fzn2smt + yices2

En l'apartat de mètodes i resolucions aplicades s'explicarà amb més profunditat les característiques de cadascun d'ells.

5.1.3.- Dades i Relacions a tenir en compte

A l'hora de dissenyar les relacions entre les diferents dades i estructures de dades hem d'anar molt en compte perquè el sistema ens permeti aconseguir la coherència, integritat i filtrat necessari pel correcte funcionament del programa.

Com hem vist en un punt anterior, haurem de guardar informació sobre:

- Assignatures, característiques generals, tipologies derivades i grups derivats de les tipologies.
- Professors, dades generals, nom, cognoms
- Capacitacions dels professors a l'hora de poder impartir una assignatura.
- Horaris, dels professors i de les assignatures.

Primer de tot haurem de tenir en compte que assignatures, tipologies i grups han d'estar íntimament relacionats, si no seguint una estructura jerarquitzada per saber en tot moment de qui depèn cada entitat.

A més a més, les assignatures/tipologies/grups han de tenir un horari concret, el qual s'ha de buscar la manera que sigui fàcilment configurable i extremadament flexible. També hem de tenir en compte que cada assignatura/tipologia/grup tindrà un nombre concret de crèdits.

Amb els professors, hem de tenir en compte per un costat les restriccions de nombre de crèdits (un professor pot fer $x_1 < x < x_2$ crèdits) i per altre, tal com passava amb les assignatures/tipologies/grups, s'ha de tenir cura en dissenyar un sistema flexible d'assignació horària (en aquest cas de disponibilitat).

A partir dels professors i les assignatures/tipologies/grups ens adonem que entre ells també han d'estar relacionats, per controlar en tot moment si un professor en concret té la preparació o perfil necessari per exercir la docència en una assignatura/tipologia/grup.

Aquí afegiríem un nou concepte, que és el de les presolucions o solucions parcials del problema, que ens hauria de permetre fixar certes solucions abans d'enviar l'instancia del problema al resoledor.

5.1.4.- Restriccions contemplades

Un cop dissenyades les dades a tenir en compte, serà molt important decidir quines restriccions aplicarem a la resolució del problema.

Primer de tot, la més obvia serà que una assignatura/tipologia/grup només ho pot donar un professor i no només això, si no que ho ha de donar un professor, no podem deixar assignatures/tipologies/grups 'orfes', no està contemplat ni el cas de tenir dos o més professors a classe ni el de no tenir-ne cap.

També hem de tenir en compte el nombre de crèdits que ha i que pot exercir un professor, i la solució ha de moure's en aquest interval, per exemple si un professor ha de donar entre 10 i 20 crèdits, la solució no pot ser ni 4 ni 23, però si podria ser 16 per exemple.

Una altra restricció a tenir en compte és la capacitat o perfil d'un professor per poder assignar-li docència d'una assignatura/tipologia/grup en concret, per exemple si algú està especialitzat en àlgebra, però no en gestió dels processos de producció d'una empresa, que podem evitar que li sigui assignada aquesta última assignatura/tipologia/grup.

Llavors hem de tenir en compte que hi haurà assignatures/tipologies/grups els horaris dels quals seran incompatibles amb els del professor, és a dir, si una assignatura/tipologia/grup té classe el divendres de 15h a 17h i un professor els divendres no està disponible, llavors en cap cas li podrà ser assignada aquesta assignatura/tipologia/grup.

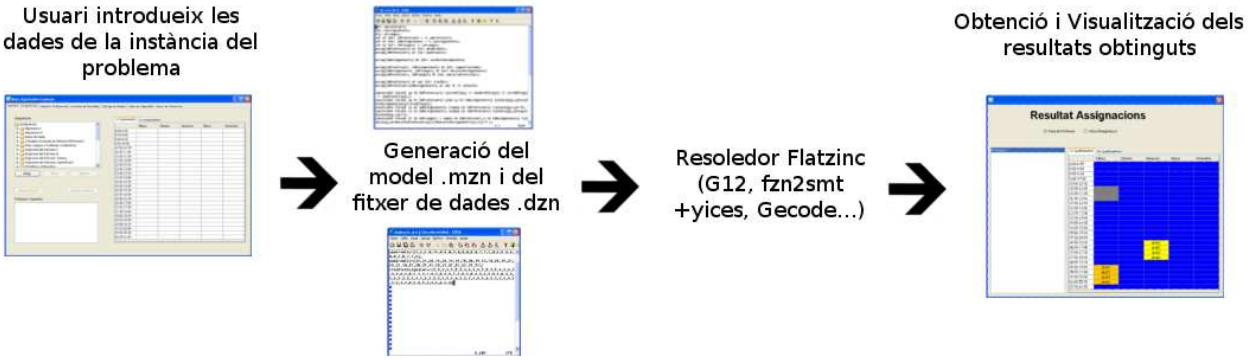
Evidentment, un professor no podrà tenir assignades dues assignatures/tipologies/grup que comparteixin part o tot l'horari.

A partir de les anteriors consideracions i guiats per un possible usuari final, s'han tingut en compte les següents restriccions, comentades a l'apartat de la creació del model de resolució del problema:

- Obligar que la solució assigni un nombre de crèdits a cada professor que estigui dins l'interval definit.
- Obligar que cada grup estigui assignat a un professor, evitant grups sense professor i grups amb més d'un professor assignat.
- Evitar l'assignació d'un grup a un professor que no pugui realitzar-ne docència per causa de capacitació o disponibilitat horària.
- Evitar que hi hagi grups assignats a un professor que es solapin entre ells.

Apart de les restriccions implementades, se n'han contemplat altres de cara a necessitats i ampliacions futures, tal com es comenta a l'apartat de “Reflexions i possibles millores”, on s'haurien de fer canvis en el model i passar a definir l'assignació per optimització en comptes de per decisió, és a dir, d'entre les diferents solucions presents, es triaria la que més s'adeqüi a la definida en el model.

5.1.5.- Estructura conceptual del funcionament del programa



L'estructura del programa serà la següent:

Primer de tot l'usuari ha d'introduir les dades corresponents a assignatures/tipologies/grups, horaris, capacitacions, professors i limitacions de crèdits. Llavors el programa generarà un model i un fitxer amb les dades que hem introduït en format interpretable pel conversor mzn2fzn (de format minizinc a format flatzinc), i gràcies a aquest conversor, obtindrem un fitxer fzn. Un cop generat aquest fitxer, l'utilitzarem amb el resoledor flatzinc que ens interessi, donat que el codi és genèric i és compatible amb tots ells. Per últim el resoledor ens tornarà el resultat del càlcul de la instància del problema segons les dades d'entrada introduïdes per l'usuari, moment en el qual el programa capturarà aquest resultat i l'interpretarà, transformant-lo en un format visual fàcilment interpretable per l'usuari final.

Exemple de model .mzn

```
	int: nprofessors;
	int: nassignatures;
	int: nfranges;
	set of int: idProfessors = 1..nprofessors;
	set of int: idAssignatures = 1..nassignatures;
	set of int: idFranges= 1..nfranges;
	array[idProfessors] of int: minCredits;
	array[idProfessors] of int: maxCredits;
	array[idAssignatures] of string: nomAssignatures;
	array[idAssignatures] of string: nomGrups;
	array[idAssignatures] of int: tipologies;

	array[idAssignatures] of int: creditsAssignatura;
	array[idProfessors, idAssignatures] of int: capacitacions;
	array[idAssignatures, idFranges] of int: horarisAssignatures;
	array[idProfessors, idFranges] of int: horarisProfessors;
	array[idProfessors] of var int: credits;
	array[idProfessors,idAssignatures] of var 0..1: solucio;
	array[idProfessors,idAssignatures] of 0..1: incompatiblesProf;
	array[idAssignatures,idAssignatures] of 0..1: incompatibles;

	constraint forall (p in idProfessors) ((credits[p] >= minCredits[p]) /\ (credits[p] <=
	maxCredits[p]));

	constraint forall (p in idProfessors) (sum (a in idAssignatures)
	(solucio[p,a]*creditsAssignatura[a])=credits[p]);

	constraint forall (a in idAssignatures) (sum(p in idProfessors) (solucio[p,a])=1);

	constraint forall (p in idProfessors, a in idAssignatures) ( if (incompatiblesProf[p,a]=1) then
	solucio[p,a]=0 else true endif );

	constraint forall (a in idAssignatures, p in idProfessors) (if (incompatiblesProf[p,a]=1) then
	(forall (b in idAssignatures) (solucio[p,a]+solucio[p,b]<2)) else true endif);

	constraint forall (a in idAssignatures, b in idAssignatures) (if (incompatibles[a,b]=1 /\ a!=b)
	then (forall (p in idProfessors) ((solucio[p,a]==1)->(solucio[p,b]==0))) else true endif);

solve :: int_search(credits, first_fail, indomain_min, complete) satisfy;

output [ show(solucio[p,a]) ++ " " ++ "\n" | p in idProfessors, a in idAssignatures ] ++ [
show(credits[p]) ++ " " ++ "\n" | p in idProfessors ];
```

Exemple de fitxer de dades .dzn

```
nprofessors=1;
nassignatures=3;
nfranges=260;
minCredits=[0];
maxCredits=[10];
creditsAssignatura=[1,3,3];
capacitacions=[|1,1,1|];
nomGrups=["|grup1","|grup2","|grup3",];
nomAssignatures=["|Assignatura1","|Assignatura1","|Assignatura1",];
tipologies=[0,1,3,];
incompatibles=[|1,0,0|0,1,0|0,0,1|];
incompatiblesProf=[|0,0,0|];
horarisAssignatures=[|0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0|0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0|0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0|];
horarisProfessors=[|1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1|];
```

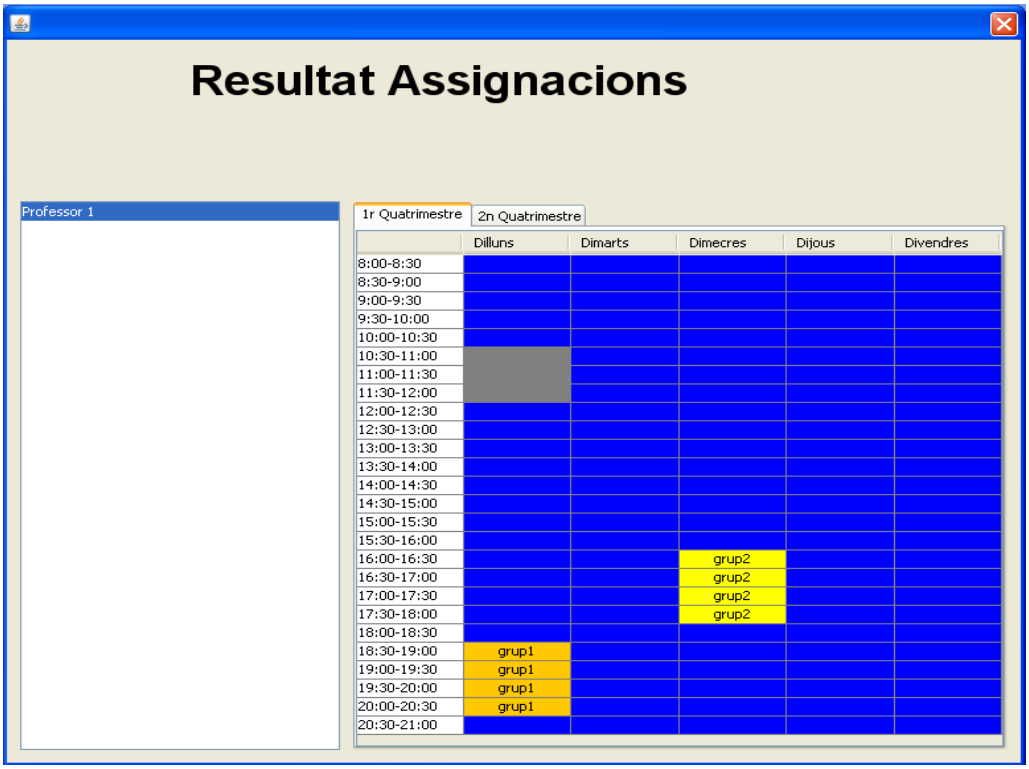
Fitxer fzn obtingut a partir del model mzn i del fitxer de dades dzn

```
array [1..3] of int: capacitacions = [1, 1, 1];
array [1..3] of int: creditsAssignatura = [1, 3, 3];
array [1..780] of int: horarisAssignatures = [0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];
array [1..260] of int: horarisProfessors = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1];
array [1..9] of int: incompatibles = [1, 0, 0, 0, 1, 0, 0, 0, 1];
array [1..3] of int: incompatiblesProf = [0, 0, 0];
array [1..1] of int: maxCredits = [10];
array [1..1] of int: minCredits = [0];
array [1..3] of int: tipologies = [0, 1, 3];
array [1..1] of var int: credits :: output_array([1..1]);
array [1..3] of var 0..1: solucio :: output_array([1..1, 1..3]);
constraint int_le(0, credits[1]);
constraint int_le(credits[1], 10);
constraint int_lin_eq([1], [solucio[1]], 1);
constraint int_lin_eq([1], [solucio[2]], 1);
constraint int_lin_eq([1], [solucio[3]], 1);
constraint int_lin_eq([-1, 1, 3, 3], [credits[1], solucio[1], solucio[2], solucio[3]], 0);
solve  :: int_search(credits, first_fail, indomain_min, complete) satisfy;
```

Exemple de resultat obtingut del resoledor a partir del fitxer fzn

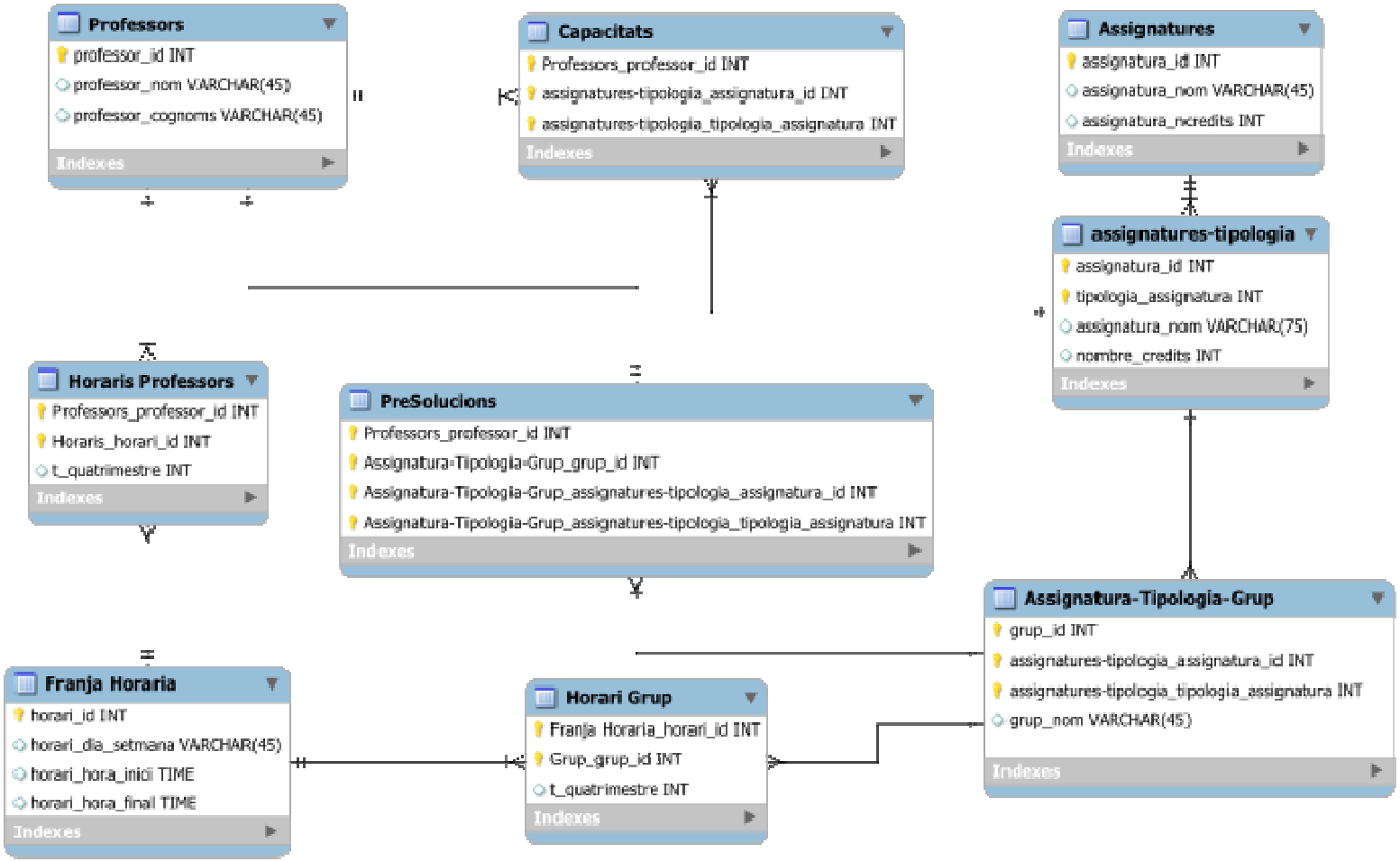
```
credits = array1d(1..1, [7]);
solucio = array2d(1..1, 1..3, [1, 1, 1]);
----------
```

Exemple gràfic de la interpretació dels resultats obtinguts:



6.- Disseny i implementació

6.1.- Model Relacional de la Base de Dades



6.2.- Estructuració i tractament de les dades

Hem de diferenciar molt bé els diferents tipus de dades amb els que treballarem, per un costat els que es guardaran directament a la base de dades i per altre, les estructures de dades aconseguides i filtrades provinents de la base de dades que s'utilitzaran en temps d'execució.

6.2.1.- Informació guardada a la Base de Dades

Tal com hem explicat en l'apartat de les dades a tenir en compte, a la base de dades hi guardarem informació sobre:

- Professors
- Assignatures/Tipologies/Grups
- Capacitacions
- Horaris
- Presolucions

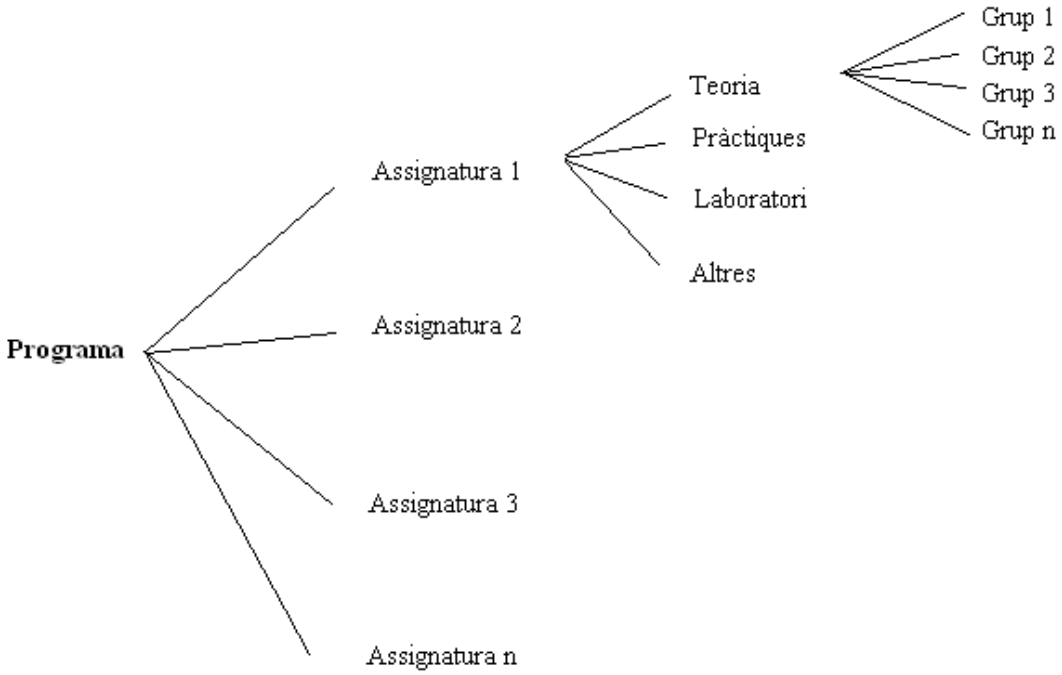
Hem de ser especialment estrictes a l'hora de detectar possibles necessitats futures a l'hora d'ampliar l'abast de l'aplicació, per la qual cosa he intentat fer-ho el més genèric possible de manera que sense canvis o amb els mínims canvis possibles, es pugui adaptar a nous casos.

A partir de les consideracions anteriors, el disseny de la base de dades per l'aplicació ha quedat així:

6.2.2.- Raonaments derivats de l'estructuració i funcionament de la base de dades

- **Separar assignatures, assignatures-tipologies, assignatures-tipologies-grups.**

S'ha fet així per evitar la duplicitat de les dades (per exemple el nom de les assignatures en totes les tipologies i grups). Segueix una estructura jeràrquica que consisteix en que hi ha *n* assignatures, cada assignatura té (fins a) quatre tipologies i cada tipologia té *m* grups.



- **No utilitzar un ID únic per cada tipologia o grup.**

  S'ha fet així per facilitar les cerques derivades de la selecció de dades per assignatura, tipologia i/o grup. Mitjançant índex s'ha optimitzat la velocitat de les mateixes. A més a més, d'aquesta manera evitem actualitzacions en cascada a l'hora d'actualitzar dades de l'assignatura.

  Es podria avaluar afegir l'ID únic per simplificar algunes operacions de cerca, però en tot cas seria informació redundant.

- **Afegir les quatre tipologies possibles un cop s'afegeix una assignatura, i filtrar segons el nombre de crèdits.**

  Si hi hagués un nombre variable de tipologies, hagués optat per l'addició dinàmica de les mateixes, però com que el nombre estava definit, he preferit inserir-les primer i actualitzar-les amb el nombre de crèdits quan calgui.

  Aquest és un dels punts a millorar en una futura modificació del programa, flexibilitzar el

nom i nombre de tipologies.

- **La taula t_capacitacions no té en compte els grups si no les tipologies, per què ?**
  Segons es va acordar amb els tutors, les capacitacions corresponen a les tipologies, no als grups, els quals 'hereten' de les tipologies la capacitació corresponent. Les úniques dades que es guarden és l'assignatura, la tipologia i el professor. D'aquesta manera podem filtrar per assignatura, tipologia o professor sense haver de dependre de cap altra taula (a menys que necessitem els noms de professors o assignatures, on es relacionarà directament amb la taula corresponent).
- **Com s'actualitza el nombre de crèdits d'una assignatura ?**
  El nombre de crèdits d'una assignatura s'actualitza en el moment que es modifica el nombre de crèdits de les tipologia derivades d'aquella assignatura, és a dir, si l'assignatura X té 3 crèdits de teoria, 5 de pràctiques i 2 d'altres, l'assignatura tindrà 10 crèdits en total.
- **El nombre de crèdits d'un grup, d'on s'extreu ?**
  El nombre de crèdits d'un grup sempre serà el mateix que la seva tipologia; d'aquesta manera només caldrà modificar la tipologia en comptes d'haver d'actualitzar en cascada els crèdits de cada grup derivat de la mateixa.
- **La taula t_franja_horaria manté una estructura de separar dies, hores d'inici i hores de fi, i s'utilitza una ID única, contravenint el raonament de les assignatures-tipologies-grups, per què ?**
  Per un costat s'ha preparat la taula de manera que es pugui crear un horari personalitzat per cada dia de la setmana a nivell de segon, és a dir, en casos concrets podríem discriminar des d'un segon a 86400 segons (un dia).
  Llavors s'utilitza l'ID única per poder identificar fàcilment les franges horàries en cas que no estiguin ordenades a la base de dades.
- **La taula t_horaris_grups, com funciona ?**
  Donat que la taula t_franja_horaria s'estructura a partir de l'ID, unim aquesta dada amb assignatura – tipologia – grup (per poder filtrar d'una manera simple i eficient i pel disseny explicat anteriorment). Es fa de manera dinàmica, és a dir, si s'afegeix una franja horària a un grup, s'insereix a la taula, i s'elimina en cas contrari. És la mateixa estructura que segueix la taula dels horaris dels professors.
- **Per què ens serveix la taula de presolucions ?**
  Ens serveix per assignar d'una manera extremadament simple solucions ja calculades o

prefixades, en la forma assignatura-tipologia-grup i professor. D'aquesta manera podem saber en qualsevol moment quines assignacions són predefinides i quines calculades.

6.2.3- Disseny interfície d'usuari

A l'hora de dissenyar l'interficie que utilitzarà la persona responsable de l'execució del programa s'han estudiat diferents alternatives a l'hora d'estructurar els diferents apartats i dades presents en el problema. En tot moment s'ha optat per l'agilitat i simplificació de les accions habituals de manteniment de les dades, així com les de configuració.

En un primer moment es va identificar com a essencial el manteniment d'assignatures/tipologies/grups i el de professors com una de les parts més importants de l'interacció amb l'aplicació, per la qual cosa es va posar especial èmfasi en la manera que es mostraven dites dades.

Per un costat, tenint les assignatures, tipologies i grups com una estructura jeràrquica on, començant per les assignatures, les tipologies hereten les propietats de les assignatures i els grups hereten les propietats dels grups, tal com explico en el següent apartat d'estructures de dades utilitzades en l'elaboració del programa. Una forma entenedora i flexible de mostrar aquesta informació és la de crear un arbre de tres nivells, tal com es veu a la imatge següent:

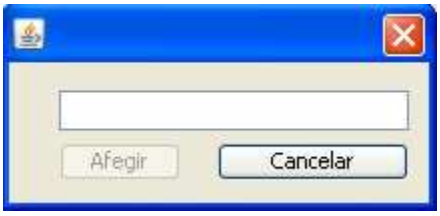
A dalt de tot tenim l'arrel "Assignatures", de la qual en surten cinc branques (Matemàtiques, Física I, TOGE, MTP i Ciencies Ye). Llavors, dins de la primera assignatura (Matemàtiques), veiem que apareixen dues tipologies (Pràctiques i Altres). S'observa un icona diferent a Altres respecte a Pràctiques, donat que Altres encara no té cap grup creat. Com a últim, hi ha el Grup A, com a Pràctica de l'assignatura Matemàtiques. Per poder discriminar entre Assignatures, Tipologies i Grups, vaig utilitzar els objectes Assignatura, AssignaturaTipologia i AssignaturaTipologiaGrup, respectivament, com explico d'una forma més detallada en el següent apartat d'estructures de dades.

Una part molt important apart de la visualització és com s'afegeixen, modifiquen i s'eliminen

assignatures, tipologies i grups. Després d'estudiar diferents opcions, com utilitzar un menú que sortís al fer clic amb el botó dret sobre algun dels objectes, vaig optar per fer-ho més genèric, tal com es veu a la imatge, amb tres botons ("Afegir", "Editar" i "Eliminar"), que agafen un comportament (o queden deshabilitats) segons quin sigui l'objecte seleccionat. Llavors, segons l'objecte tenim els comportaments següents:

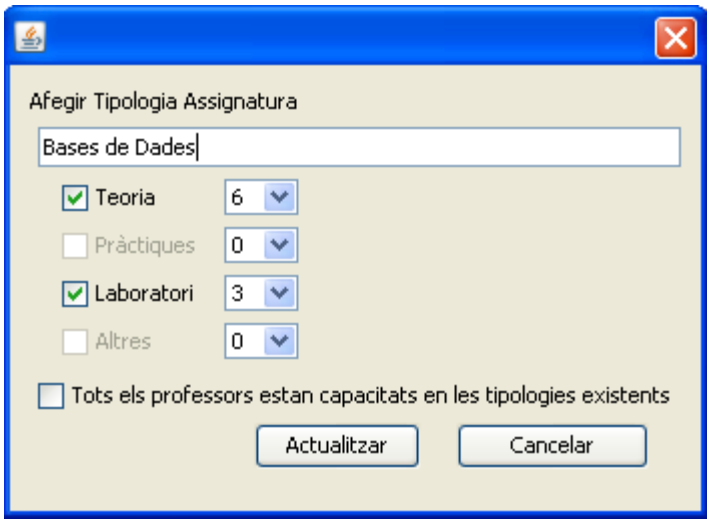
Objecte\Botó	**Afegir**	**Editar**	**Eliminar**
Arrel (on hi diu "Assignatures")	Afegir Assignatura	Deshabilitat	Deshabilitat
Assignatura	Modificar Nom Assignatura, nombre de crèdits d'una tipologia	Modificar Nom Assignatura, nombre de crèdits d'una tipologia	Eliminar Assignatura (i dades relacionades per mantenir la coherència a la base de dades).
AssignaturaTipologia	Afegir Grup	Modificar Nom Assignatura, nombre de crèdits d'una tipologia	Eliminar Tipologia i informació relacionada.
AssignaturaTipologiaGrup	Afegir Grup		

- Exemple d'afegir Assignatura (clic en Afegir tenint seleccionada l'arrel):



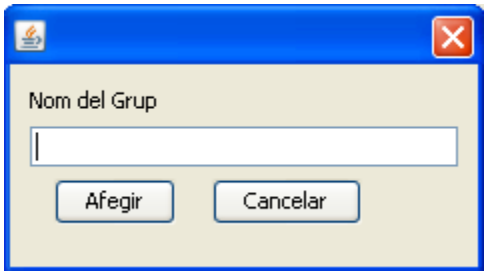
Només cal que posem el nom de l'assignatura en qüestió, donat que no cal més informació per crear una assignatura nova.

- Exemple del que apareix en clicar "Afegir" o "Editar" tenint seleccionada una Assignatura o "Editar" tenint seleccionada una AssignaturaTipologia:

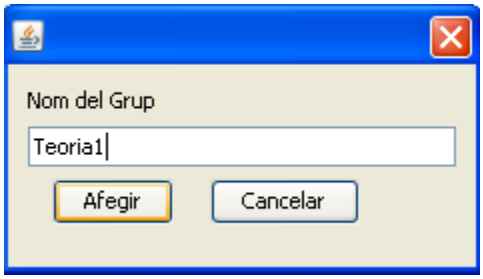


Tal com veiem, es pot modificar el nom de la mateixa així com el nombre de crèdits de cada tipologia. Més avall veiem una opció per seleccionar que és "Tots els professors estan capacitats en les tipologies existents", que serà explicada en l'apartat del manteniment de les capacitacions.

– Exemple del que apareix en clicar "Afegir" tenint seleccionada una AssignaturaTipologia:

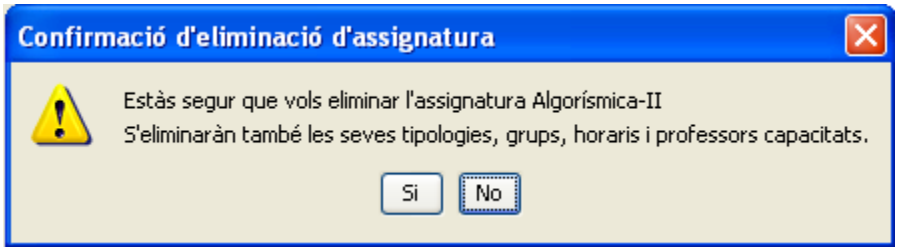


– Exemple del que apareix en clicar "Editar" tenint seleccionada una AssignaturaTipologiaGrup:

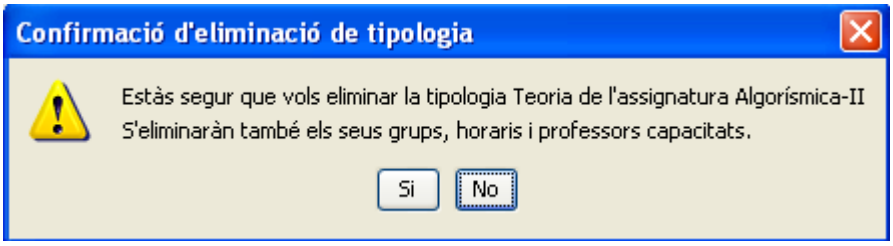


A l'hora d'eliminar una assignatura, assignaturaTipologia o assignaturaTipologiaGrup s'avisa a l'usuari que aquesta acció comportarà també l'eliminació de les dades relacionades:

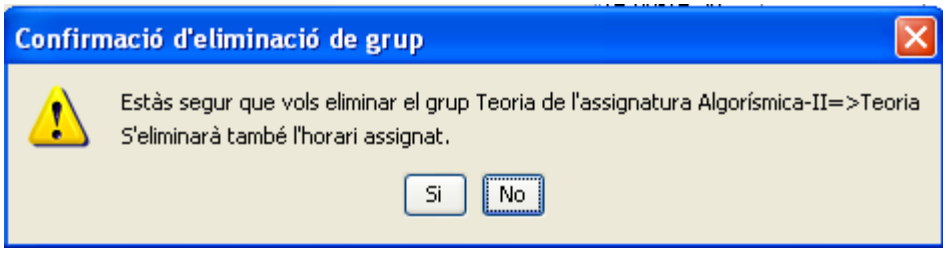
– Assignatura.



– AssignaturaTipologia

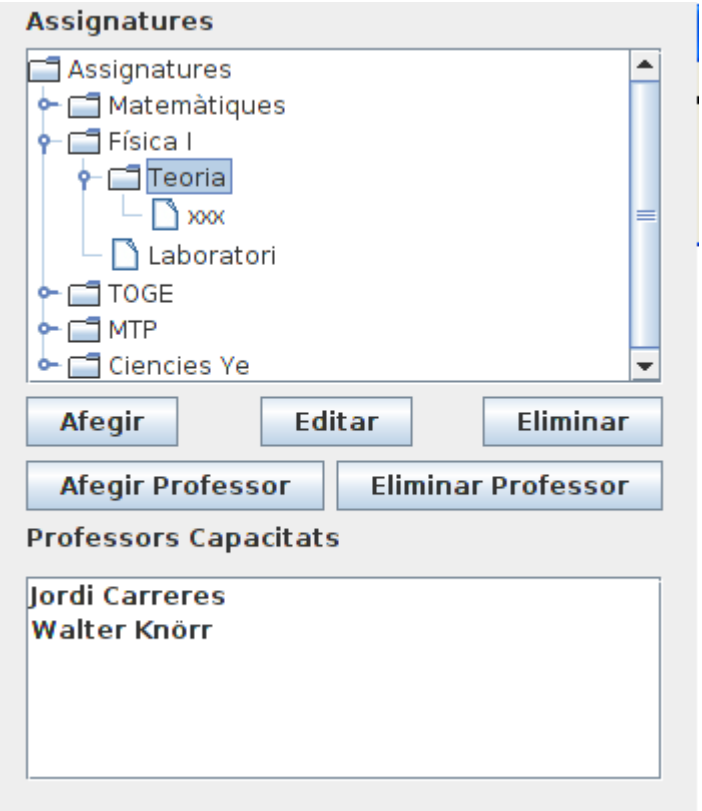


– AssignaturaTipologiaGrup

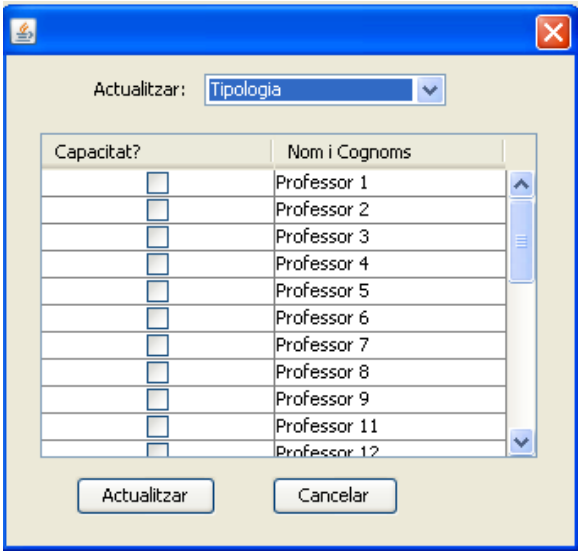


Apart del manteniment de les assignatures, tipologies i grups, en el mateix apartat vaig pensar que calia agrupar les dades que estaven íntimament relacionades, com són els professors capacitats i els horaris de l'assignatura, tipologia o grup.

Pels professors, al no definir una especial jerarquia, vaig optar per utilitzar una llista estàndard per mostrar-los, de manera que en el moment que es selecciona una assignaturaTipologia, apareixen els professors capacitats de la mateixa, o en el cas d'un grup, els professors capacitats de la tipologia d'aquell grup.



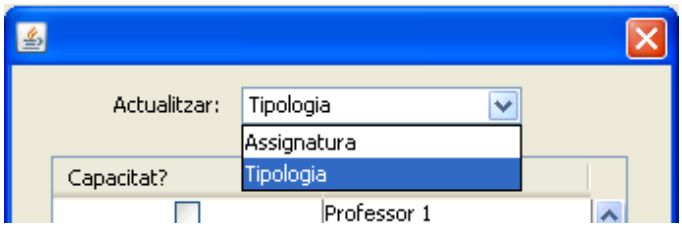
Per fer que un professor pugui donar una assignatura (hi estigui capacitat), simplement hem de fer clic al botó d'"Afegir Professor" tenint marcada una AssignaturaTipologia, i llavors li apareixerà aquesta pantalla:



En aquesta taula ens apareix per un costat un quadre on es pot marcar si aquell professor està capacitat per la tipologia seleccionada i a l'altre, el nom del professor perquè el podem identificar. Cal esmentar que en aquesta llista només apareixen els professors que encara no estan capacitats per

la tipologia.

Si es mira amb més deteniment la pantalla, es veu a la part superior un desplegable amb diferents opcions:

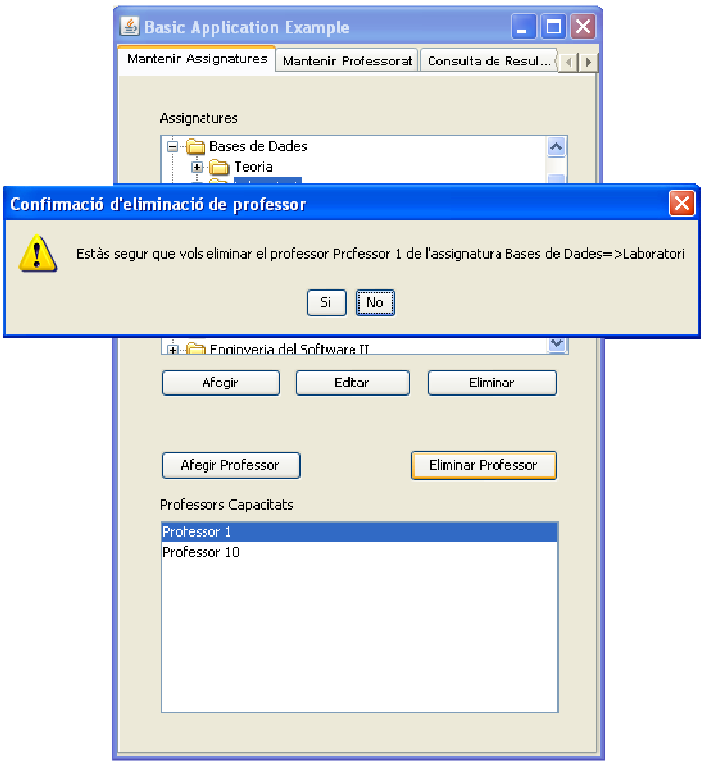


Aquest desplegable és important, ja que, de cara a l'usuari s'ha habilitat de manera que pugui capacitar un o més professors per una assignatura completa (en totes les tipologies que tingui) simplement triant "Assignatura" en comptes de tipologia, per fer més ràpida l'entrada de dades.

És a dir, a l'hora d'entrar els professors capacitats tenim diferents opcions:

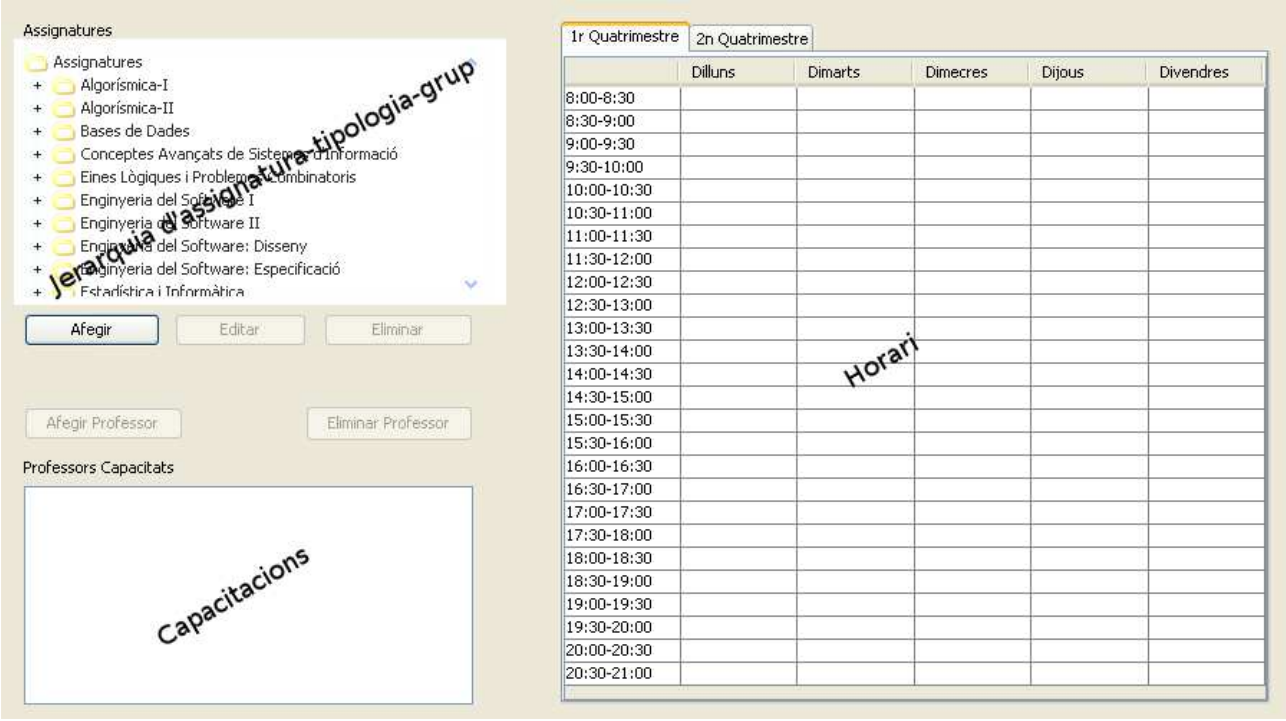
- Si afegim capacitacions tenint seleccionada una Assignatura, els professors marcats els capacitarà per a totes les tipologies disponibles de la mateixa.
- Si afegim capacitacions tenint seleccionada una AssignaturaTipologia o AssignaturaTipologiaGrup, podrem marcar si volem capacitar els professors marcats només per aquella tipologia (opció "Tipologia") o bé per tota l'assignatura (opció "Assignatura").

Per eliminar la capacitació d'un professor en una determinada tipologia d'una assignatura, hem de seleccionar primer dita tipologia, fet que farà que ens aparegui la llista de professors actualment capacitats en la mateixa, seleccionar el professor que volem retirar-li la capacitació i fer clic al botó d'"Eliminar Professor", llavors ens demanarà confirmació per acabar d'executar l'acció, tal com s'observa a la imatge de la dreta.

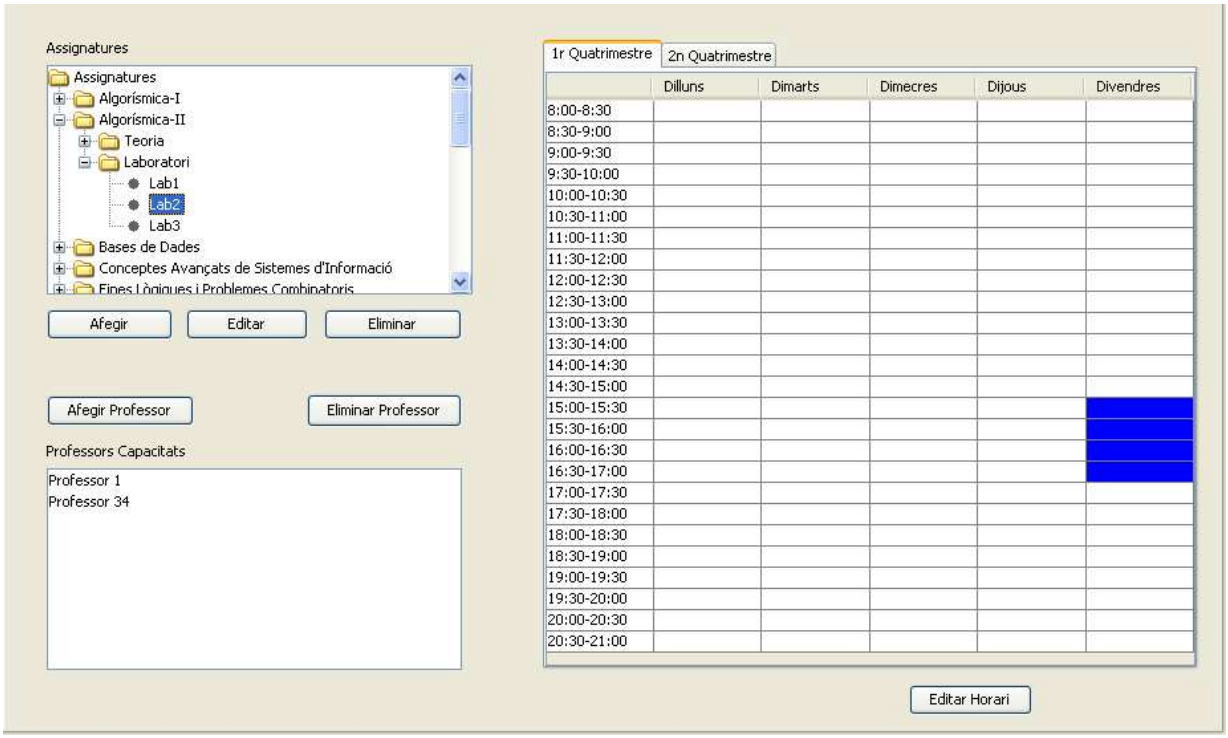


A més a més de la jerarquia d'assignatures-tipologia-grups i de les capacitacions, vaig trobar

necessària la presència de la relació d'horaris del grup/tipologia/assignatura, de manera que la pantalla de treball de l'usuari queda distribuïda així:



L'idea és tenir tota la informació en una mateixa pantalla, de manera que es puguin mantenir tots els conceptes relacionats amb assignatura-tipologia-grup des d'un punt comú i, a més a més, que la consulta sigui el més senzilla possible. D'aquesta manera, si fem clic en un grup, veurem (i podrem editar) el seu horari, així com els professors capacitats per la docència del mateix, tal com



podem veure a la captura següent:

Si volem modificar les hores on tindrà docència el grup seleccionat, hem de fer clic al botó "Editar Horari", i el quadre de l'horari canviarà d'aspecte perquè podem seleccionar les franges horàries que ens interessin:

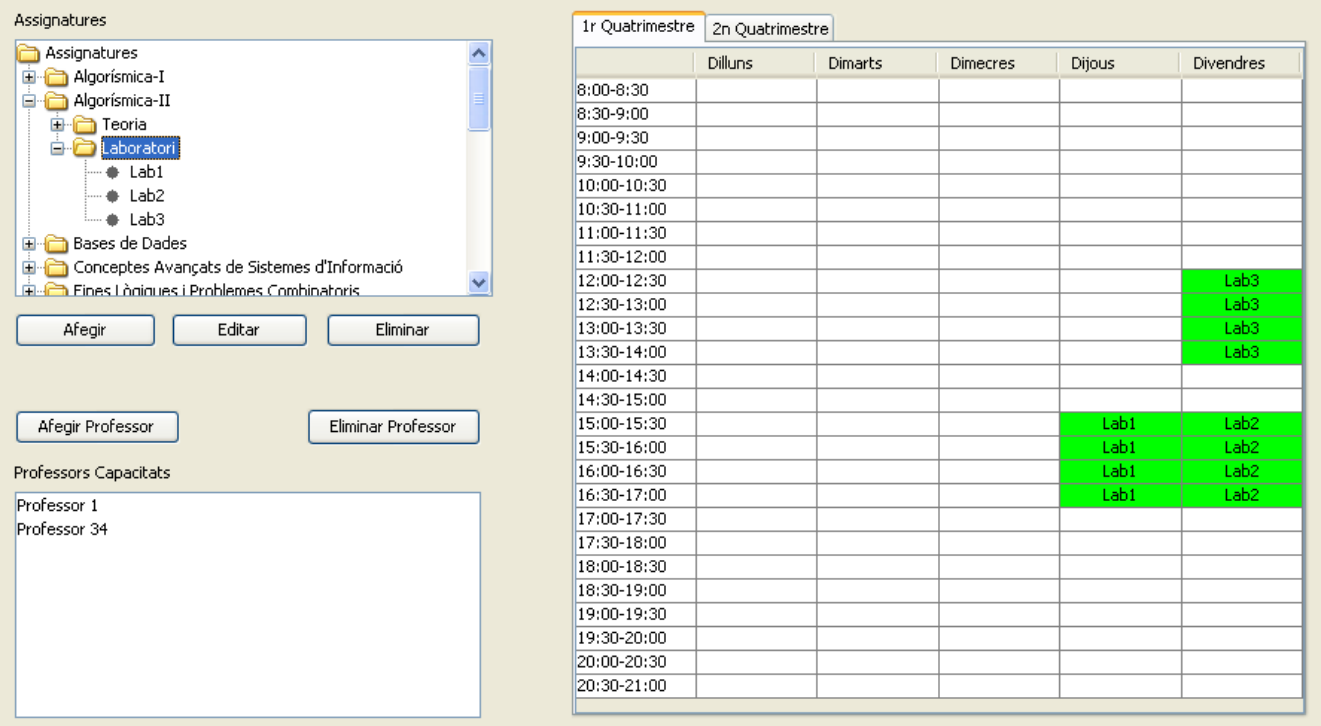
Com es veu, les quatre franges que anteriorment apareixien acolorides en blau, es troben marcades. Per actualitzar (afegir o eliminar franges) simplement hem de marcar o desmarcar el quadre que ens interessi. Un cop fet això, per confirmar els canvis, hem de fer clic a "Actualitzar". En cas que no volguéssim modificar l'horari, hauríem de fer clic a "Cancelar".

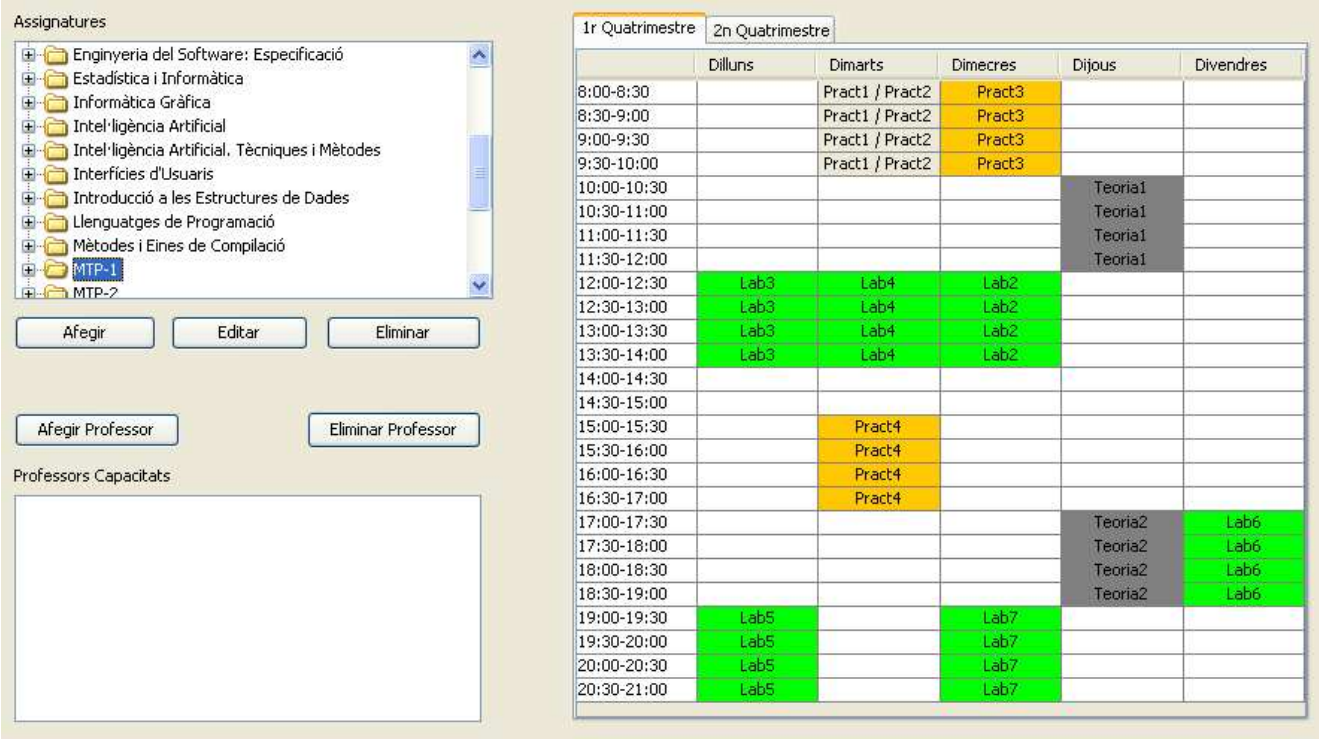
També es pot apreciar com és possible triar el quadrimestre a la part superior, on apareixen dues pestanyes. L'aplicació està pensada i dissenyada perquè sigui possible una ampliació de cara a poder posar-hi els intervals de temps (quadrimestres, mesos, etc) que sigui necessari, de cara a possibles necessitats futures.

Una altra cosa que pot ser interessant a l'hora de visualitzar horaris és poder veure la distribució dels grups en una tipologia, o directament la distribució de tota l'assignatura, amb les

tipologies i grups corresponents. Simplement fent clic a sobre d'una tipologia que tingui grups derivats (si no no mostra res al no haver-hi horaris relacionats) o sobre una assignatura que tingui tipologies, ens mostrarà la relació d'horaris dels diferents grups derivats, en un color característic de cada tipologia (per diferenciar fàcilment), tal com es veu a les dues imatges de la pàgina següent.

Vista dels Horaris triant una tipologia

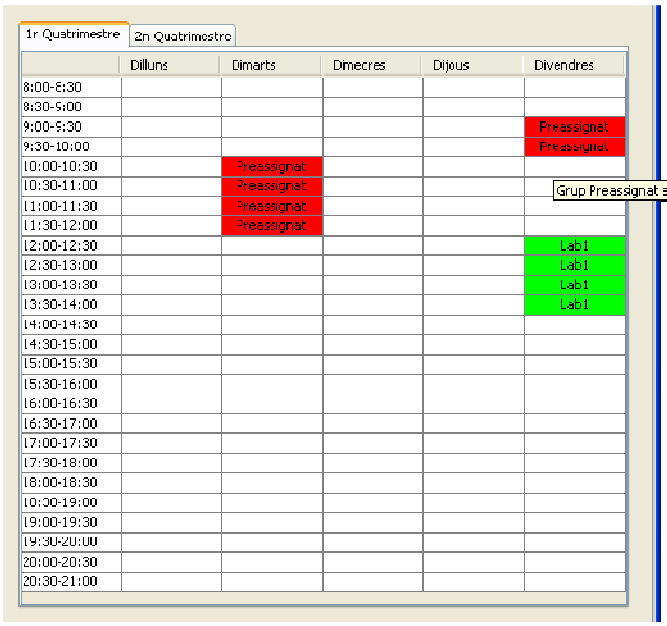




Vista dels Horaris triant una assignatura

Si hi ha coincidències en els horaris dels diferents grups, surt en un color diferent i es mostren els noms dels grups separats per una barra "/".

El fet de poder preassignar solucions (presolucions) implica que sigui interessant la seva visualització al mateix temps que ens mostra els diferents grups. Aquestes presolucions es mostren de color vermell, i en el moment que hi passem per sobre ens indica el nom del grup i el professor assignat, tal com podem veure a la captura següent:

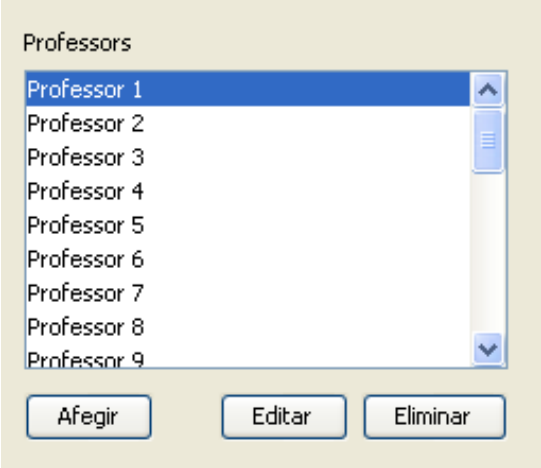


Cal destacar que les presolucions tant apareixen si seleccionem una assignaturaTipologiaGrup, assignaturaTipologia o assignatura.

Un cop explicada la pantalla del manteniment d'assignatures-tipologies-grups amb les seves capacitacions i els seus horaris, una altra necessitat que hi havia era la del manteniment de professors.

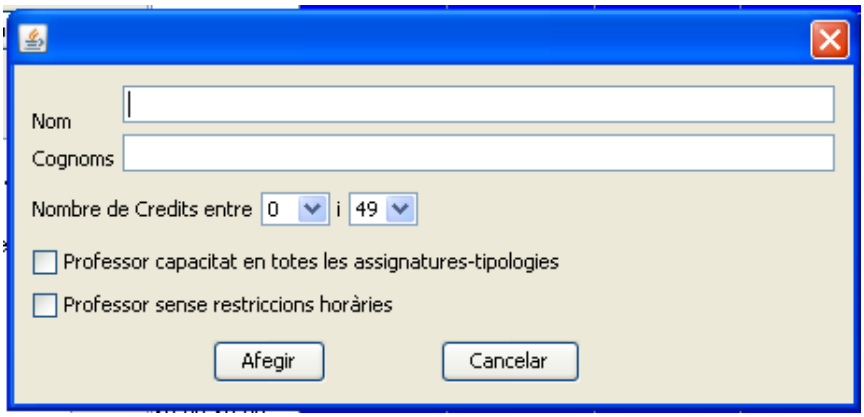
S'ha intentat mantenir una estructura semblant a l'anterior apartat per coherència i per facilitar la tasca a l'usuari que utilitzarà l'aplicació.

Com que no s'ha definit cap tipus de jerarquia a l'hora d'organitzar els professors, s'ha optat per mostrar-los com a llista de noms i cognoms:



Tal com apareixia a l'apartat d'assignatures-tipologies-grups, hi ha els tres botons que ens permetran fer el manteniment dels professors.

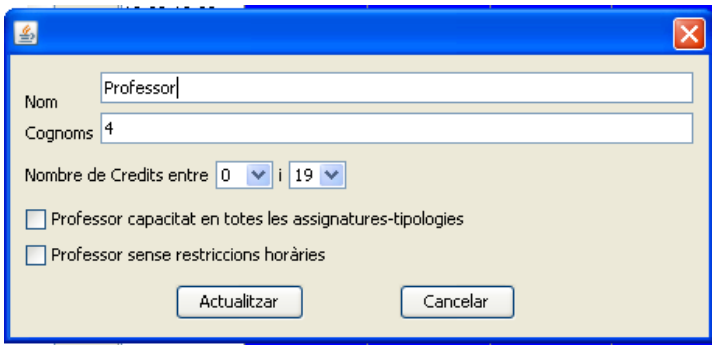
Si volem afegir un professor, fem clic a "Afegir" i ens apareixerà la pantalla següent:



Hem d'introduir el seu nom, cognoms i seleccionar l'interval de crèdits que li poden ser assignats. Llavors, per estalviar passos en casos on un professor pugui donar qualsevol assignatura-tipologia, es pot marcar el primer quadre. L'últim quadre ens serveis per fer l'assignació de total disponibilitat docent a un professor, en el cas que no tingui cap tipus de limitació horària al respecte.

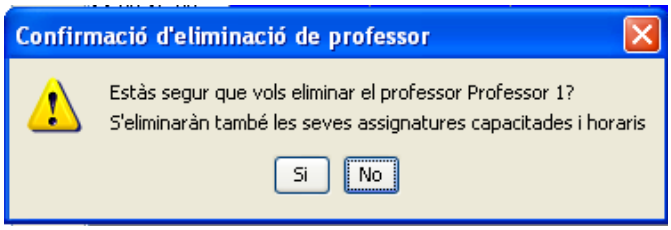
El botó per editar un professor ens mostra la mateixa pantalla que el d'afegir, però amb les

dades de nom, cognoms i interval de crèdits amb les dades corresponents carregades, tal com podem veure a la pàgina següent.

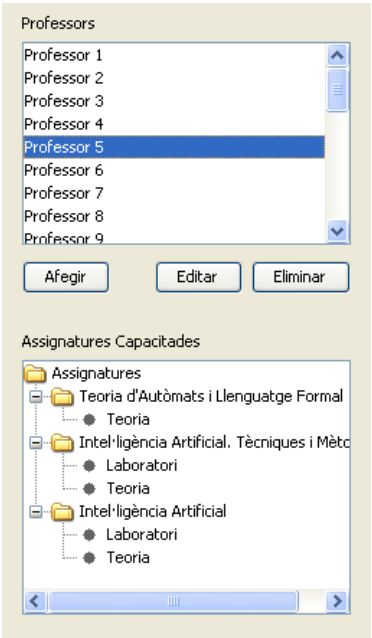


Aquí podem tornar a marcar les capacitacions i la no restricció horària si ens cal.

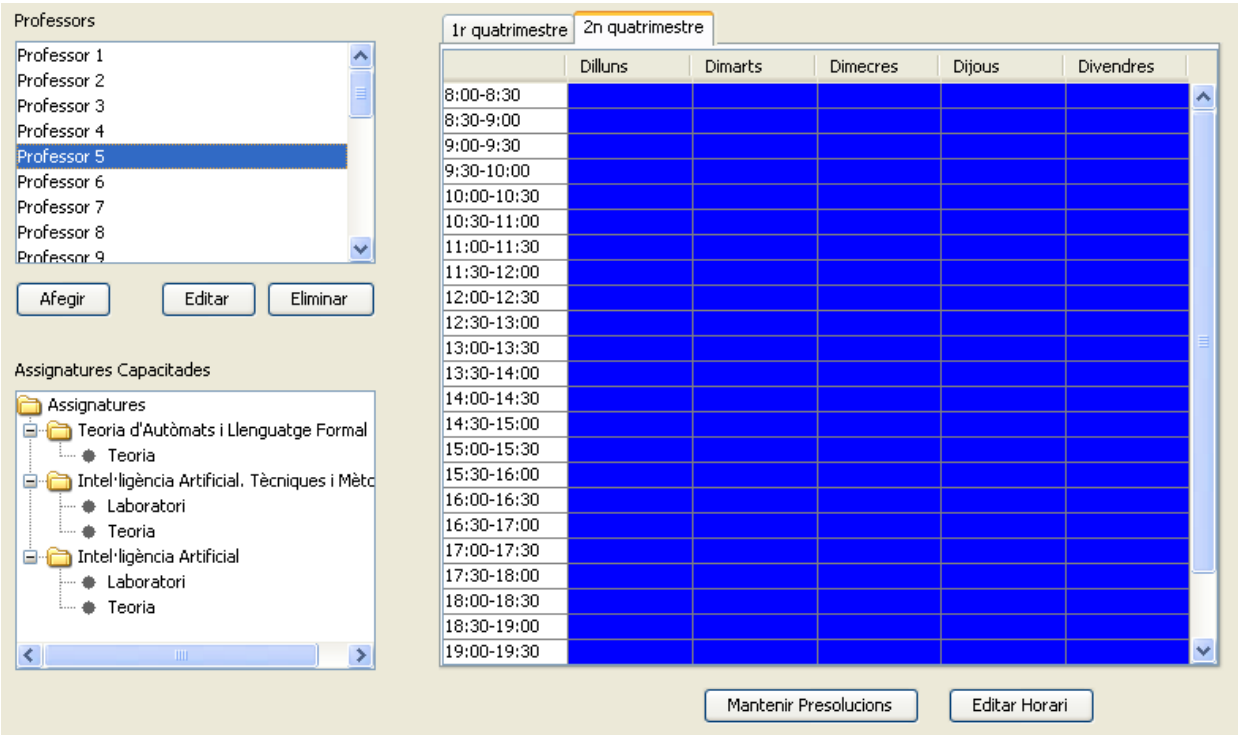
Si triem l'opció “Eliminar”, ens surt la confirmació avisant-nos que, apart dels professors, també s'eliminaran els horaris i capacitacions relacionades.



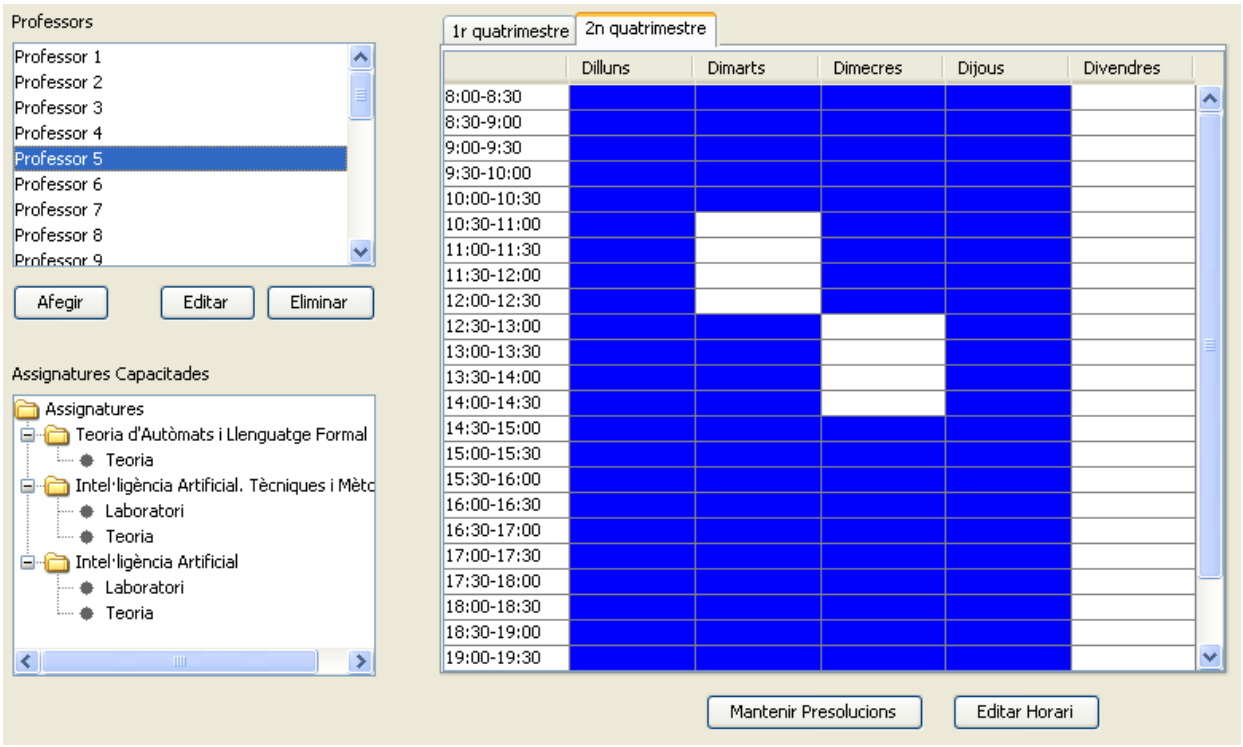
Tal com passava amb la pantalla inicial, al fer clic sobre un professor, ens apareixen dades relacionades amb el mateix, com és el cas d'assignatures capacitades, en aquest cas només com a visualització, sense poder ser editades. Remarcar que es mostra amb la jerarquia corresponent, per una interpretació més visual i senzilla.



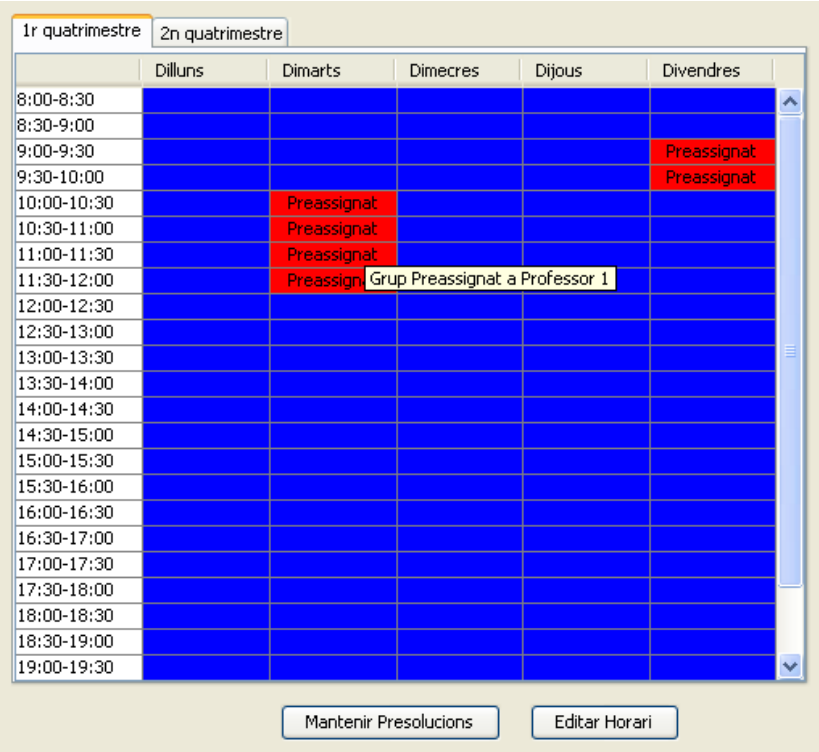
Apart de les capacitacions, cada professor també té un quadre de disponibilitat horària propi, que determinarà la possibilitat d'assignar-li grups en unes certes franges horàries.



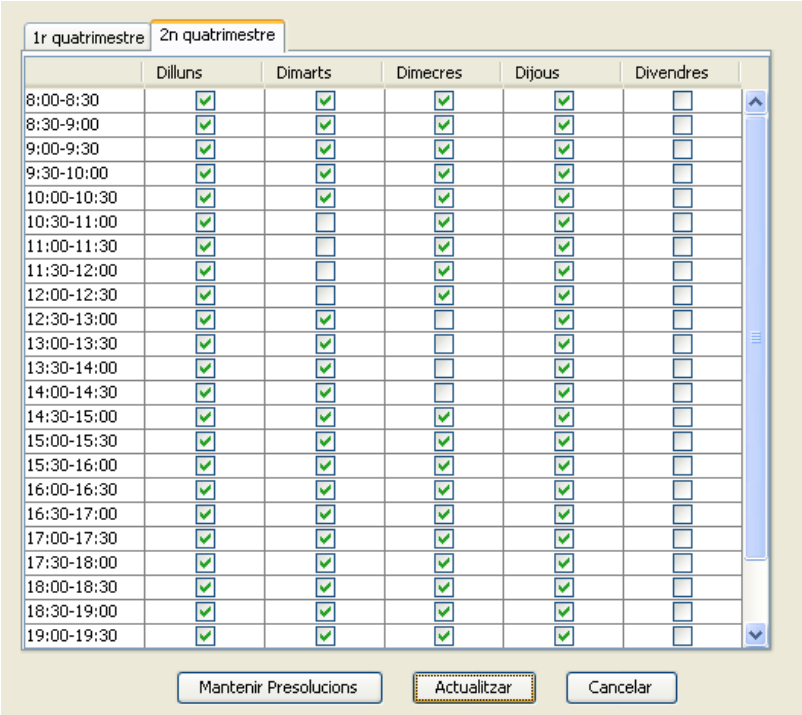
En aquest cas el professor té disponibilitat total en el segon quatrimestre. Si la disponibilitat fos menor, apareixerien marcades en blau les franges disponibles i en blanc les no disponibles, tal com podem veure a la captura següent:



Si existeix alguna preassignació, tal com passava a la primera pantalla, també ens apareix marcada en vermell:



El mecanisme de modificació d'horaris de disponibilitat dels professors és exactament igual al de l'apartat de grups, es fa clic a "Editar Horari" i a la graella apareixen els quadrats on definirem si el professor està disponible o no:



6.2.4.- Estructures de dades utilitzades en temps d'execució

S'ha utilitzat la programació per capes, per separar correctament la part de tractament de les dades respecte la de presentació, concretament s'ha seguit el patró MVC (model-vista-controlador), separant l'accés a base de dades (model), de la capa de control (controlador) i de la interfície d'usuari (vista). Ens ha servit per poder tenir un creixement sostenible de l'aplicació sense duplicitat de codi ni excessius problemes a l'hora de mantenir el codi.

S'ha intentat fer el codi el màxim genèric i reutilitzable possible, per diferents raons:

– Més fàcil interpretació i manteniment.

– Agrupació de codi en procediments semblats.

– Eliminació de duplicitats.

Un exemple és el mètode explicat més endavant i anomenat seleccionarPersonalitzat, on se li passa un Object com a paràmetre, i segons el seu tipus real es realitzen unes operacions o altres i finalment es retorna una taula de dues dimensions d'Objects (genèrica).

A l'hora d'estructurar les dades, s'ha optat per la utilització d'objectes, per homogeneïtzar el treball amb les dades. Així, he creat els següents objectes:

- Assignatura
  - Guarda informació sobre la id i el nom de l'assignatura.
  - Té les operacions més bàsiques (diferents constructors, get/set id i nom) i la funció toString(), que retorna el nom de l'Assignatura.
- AssignaturaTipologia
  - Hereta variables de l'objecte Assignatura, afegint-hi tipologia i el nombre de crèdits de la mateixa.
  - Així com l'assignatura, té, apart de les heretades, les funcions d'obtenir (get) i d'assignar (get) de les dues noves variables afegides.
  - També implementa les funcions getNomTipologia() i toString(), que ens tradueix la tipologia (treballat sempre com a enter) a la cadena de caràcters equivalent.
- AssignaturaTipologiaGrup
  - Hereta variables de l'objecte AssignaturaTipologia, afegint-hi l'id del grup i el nom del mateix.
  - Implementa Comparable, per la qual cosa hi trobem les funcions de comparació compareTo (Object x), equals(Object x) i hashCode(), necessàries pel correcte funcionament d'alguns algoritmes utilitzats.

- FranjaHoraria
  - Aquest objecte ens ajuda a construir l'horari d'un professor o una assignaturaTipologiaGrup, i es compon de cinc variables:
    - id : Identificador principal i únic.
    - dia : Dia de la setmana de l'objecte, valors compresos entre 0 i 6.
    - hora_inici : inici de la franja horària en segons.
    - hora_final : final de la franja horària en segons.
    - marcat : variable booleana utilitzada per a la visualització (ús intern, no té interacció amb la base de dades).

D'aquesta manera, podem tenir objectes que ens defineixen les diferents franges horàries que podem necessitar a nivell de segon, és a dir, podem tenir un objecte franjaHoraria que comprengui l'interval entre dilluns a les 09:00 i dilluns a les 11:00.

Una limitació que té és el fet de no poder tenir objectes franjaHoraria que englobin dies diferents. Per simplificar i eliminar càlculs es va separar per dies, però de cara a actualitzacions es pot tenir en compte, donat que el canvi és relativament senzill:

> En comptes de tenir 7 dies de 86400 segons, es tindria una setmana de 7*86400 segons = 604800 segons. Llavors per poder calcular el dia d'inici i final, es podria simplement calculant el %86400, que ens donaria un valor entre 0 i 6. Tanmateix, aquesta reflexió podria ser extrapolable a mesos o a tot l'any si fos necessari.

- Professor
  - Aquest objecte està composat per una id, informació personal (nom i cognoms) i interval mínim i màxim de crèdits que se li poden assignar.
  - Les seves operacions són les bàsiques de manipulació d'enters i cadenes de caràcters.
  - S'implementen les funcions hashCode() i equals(Object obj) pel correcte funcionament d'alguns algoritmes.
- PreSolucio
  - Està composat per un professor, per una assignaturaTipologiaGrup i per una franjaHoraria.
  - Ens servirà a l'hora de construir els horaris, per saber en tot moment quina assignaturaTipologiaGrup hi ha en una franjaHoraria i quin professor ha estat

assignat a la mateixa.

- Disposa de les operacions bàsiques per canviar les propietats dels diferents objectes.

6.2.5.- Creació del model de resolució del problema

Per a la resolució de les diferents instàncies del problema es poden crear des d'un de general fins a multitud de models específics segons el perfil de centre docent a tractar. Donat que s'ha creat l'aplicació fàcilment parametritzable, es poden tenir diferents models fàcilment intercanviables per l'estudi del rendiment de cadascun, sense perdre de vista el fet que haurien de seguir complint les restriccions imposades inicialment.

Basant-nos en el llenguatge del minizinc, s'ha creat un model genèric de resolució que ens permetrà obtenir resultats (en cas que n'hi haguessin) a partir de tants fitxers de dades d'entrada com es desitgi, tot utilitzant diferents resoledors (d'entrada l'aplicació està preparada per tres resoledors - G12, Gecode i fzn2smt+yices2- però és ampliable d'una forma fàcil i ràpida). El model explicat el podem veure tot seguit:

```
int: nprofessors;
int: nassignatures;
int: nfranges;
set of int: idProfessors = 1..nprofessors;
set of int: idAssignatures = 1..nassignatures;
set of int: idFranges= 1..nfranges;
array[idProfessors] of int: minCredits;
array[idProfessors] of int: maxCredits;
array[idAssignatures] of string: nomAssignatures;
array[idAssignatures] of string: nomGrups;
array[idAssignatures] of int: tipologies;
array[idAssignatures] of int: creditsAssignatura;

array[idProfessors, idAssignatures] of int: capacitacions;
array[idAssignatures, idFranges] of int: horarisAssignatures;
array[idProfessors, idFranges] of int: horarisProfessors;
array[idProfessors] of var int: credits;
array[idProfessors,idAssignatures] of var 0..1: solucio;
array[idProfessors,idAssignatures] of 0..1: incompatiblesProf;
array[idAssignatures,idAssignatures] of 0..1: incompatibles;

constraint forall (p in idProfessors) ((credits[p] >= minCredits[p]) /\ (credits[p] <= maxCredits[p]));
constraint forall (p in idProfessors) (sum (a in idAssignatures)
(solucio[p,a]*creditsAssignatura[a])=credits[p]);
constraint forall (a in idAssignatures) (sum(p in idProfessors) (solucio[p,a])=1);
constraint forall (p in idProfessors, a in idAssignatures) ( if (incompatiblesProf[p,a]=1) then
solucio[p,a]=0 else true endif );
constraint forall (a in idAssignatures, p in idProfessors) (if (incompatiblesProf[p,a]=1) then (forall
(b in idAssignatures) (solucio[p,a]+solucio[p,b]<2)) else true endif);
constraint forall (a in idAssignatures, b in idAssignatures) (if (incompatibles[a,b]=1 /\ a!=b) then
(forall (p in idProfessors) ((solucio[p,a]==1)->(solucio[p,b]==0))) else true endif);

solve :: int_search(credits, first_fail, indomain_min, complete) satisfy;
```

```
output [ show(solucio[p,a]) ++ " " ++ "\n" | p in idProfessors, a in idAssignatures ] ++ [
show(credits[p]) ++ " " ++ "\n" | p in idProfessors ];
```

Si anem analitzant el model per blocs tenim:

Declaració de variables:

```
int: nprofessors;
int: nassignatures;
int: nfranges;
set of int: idProfessors = 1..nprofessors;
set of int: idAssignatures = 1..nassignatures;
set of int: idFranges= 1..nfranges;

array[idProfessors] of int: minCredits;
array[idProfessors] of int: maxCredits;
array[idAssignatures] of string: nomAssignatures;
array[idAssignatures] of string: nomGrups;
array[idAssignatures] of int: tipologies;

array[idAssignatures] of int: creditsAssignatura;
array[idProfessors, idAssignatures] of int: capacitacions;
array[idAssignatures, idFranges] of int: horarisAssignatures;
array[idProfessors, idFranges] of int: horarisProfessors;
array[idProfessors] of var int: credits;
array[idProfessors,idAssignatures] of var 0..1: solucio;
array[idProfessors,idAssignatures] of 0..1: incompatiblesProf;
array[idAssignatures,idAssignatures] of 0..1: incompatibles;
```

int: nprofessors; Ens diu el nombre total de professors que hi ha.

int: nassignatures; Determina el nombre de grups que hi ha.

int: nfranges; És el nombre de franges horàries podem trobar, és a dir, el nombre de blocs de temps potencialment presents a la solució.

set of int: idProfessors = 1..nprofessors; Llista de professors, utilitzat per fer recorreguts a taules. Tindrà una llargada igual a nprofessors (definit anteriorment).

set of int: idAssignatures = 1..nassignatures; Llistat de grups, per recórrer estructures de dades. Tindrà una llargada igual a nassignatures (definit anteriorment).

set of int: idFranges= 1..nfranges; Ens servirà per recòrrer vectors i taules de dades. Tindrà una llargada igual a nfranges (definit anteriorment).

array[idProfessors] of int: minCredits; Té una llargada igual a idProfessors (el nombre total de professors) i ens indicarà el nombre mínim de crèdits que haurà de tenir assignats cadascú perquè la solució sigui considerada satisfactòria.

array[idProfessors] of int: maxCredits; Té una llargada igual a idProfessors (el nombre total de professors) i ens indicarà, complementant l'estructura de dades minCredits, el nombre màxim de crèdits que haurà de tenir assignats cadascú perquè la solució sigui considerada satisfactòria.

array[idAssignatures] of string: nomAssignatures; Conté el nom de les diferents assignatures, és freqüent que es trobin repeticions donat que una assignatura pot tenir més d'una tipologia, i aquesta més d'un grup. S'utilitza a l'hora de recuperar la solució.

array[idAssignatures] of string: nomGrups; Exactament igual que l'array nomAssignatures, però amb el nom del grup en comptes del nom de l'assignatura. Aquí no és tant probable que hi hagi repeticions però també n'hi pot haver.

array[idAssignatures] of int: tipologies; Serveix per tenir les dades sobre les tipologies relacionades amb nomAssignatures i nomGrups, per poder identificar cada assignaturaTipologiaGrup a l'hora de recuperar les dades de la solució.

array[idAssignatures] of int: creditsAssignatura; Ens indica el nombre de crèdits de cada grup, per a poder fer els càlculs de restriccions en el moment de buscar solucions.

array[idProfessors, idAssignatures] of int: capacitacions; És una taula que relaciona professors amb grups i ens indica si aquell professor pot exercir docència sobre aquell grup. Es tracta amb 0's i 1's, agafant el zero com a no capacitat i l'1 com a capacitat.

array[idAssignatures, idFranges] of int: horarisAssignatures; Aquesta és una altra taula que relaciona grups amb franges horàries, de manera que ens proporciona informació sobre si un grup té classe en una franja horària en concret o no. Es segueix el mateix conveni, un 0 significa que no hi ha classe mentre que un 1 significa que si que n'hi ha.

array[idProfessors, idFranges] of int: horarisProfessors; Ens indica la disponibilitat o no d'un professor en una certa franja horària. També s'estructura com una taula de 0's (no disponible) i d'1's (disponible).

array[idProfessors,idAssignatures] of 0..1: incompatiblesProf; Taula que relaciona professors i grups de

manera que si un professor i un grup són incompatibles (per horari, per capacitacions...) apareix marcat amb un 1, mentre que si són compatibles, hi haurà un 0.

array[idAssignatures,idAssignatures] of 0..1: incompatibles; Aquesta taula ens mostra si hi ha grups amb horaris 'solapats', és a dir, amb franges horàries comunes, indicant-ho amb un 1. Si no coincideixen amb cap franja horària, 0.

array[idProfessors] of var int: credits; Càlcul del nombre de crèdits assignats a cada professor

segons la solució calculada.

array[idProfessors,idAssignatures] of var 0..1: solucio; En aquesta taula és on es guardarà la solució calculada pel resoledor flatzinc, es tracta d'una taula que relaciona professors i grups on hi haurà un 1 per columna obligatoriament, que correspondrà al professor que li sigui assignat aquell grup.

Restriccions:

```
constraint forall (p in idProfessors) ((credits[p] >= minCredits[p]) /\ (credits[p] <= maxCredits[p]));
```

Aquesta restricció impedeix que un professor tingui assignada una quantitat de crèdits fora de l'interval donat per l'usuari, és a dir, si un professor pot donar entre 10 i 15 crèdits, segons aquesta restricció, no podrà haver-hi una solució ni amb 8 ni amb 16, per exemple.

```
constraint forall (p in idProfessors) (sum (a in idAssignatures) (solucio[p,a]*creditsAssignatura[a])=credits[p]);
```

En aquesta restricció el que fem és actualitzar la taula de crèdits de professors segons la solució calculada, és a dir, sumem el nombre de crèdits dels grups assignats al professor i ho posem a la casella corresponent de la taula de crèdits.

```
constraint forall (a in idAssignatures) (sum(p in idProfessors) (solucio[p,a])=1);
```

Amb aquesta restricció obliguem que a cada columna de la taula de solucions hi hagi un i només un 1, és a dir, que un grup només pugui i, al mateix temps, hagi de ser assignat a un professor.

```
constraint forall (p in idProfessors, a in idAssignatures) ( if (incompatiblesProf[p,a]=1) then solucio[p,a]=0 else true endif );
```

D'acord amb aquesta restricció, si un professor és incompatible amb un grup (sigui per horari o per capacitació) automàticament tindrà un 0 a la casella corresponent de la taula solucio. Ens serveix per evitar que li pugui ser assignat un grup a un professor que no pugui fer-ne docència, sigui per indisponibilitat horària o per no capacitació.

```
constraint forall (a in idAssignatures, b in idAssignatures) (if (incompatibles[a,b]=1 /\ a!=b) then (forall (p in idProfessors) ((solucio[p,a]==1)->(solucio[p,b]==0))) else true endif);
```

En aquesta restricció el que obliguem és que si dos grups són incompatibles entre si i un professor té assignat un dels grups (1), automàticament no tindrà assignat l'altre (0). D'aquesta manera evitarem que un professor pugui tenir dos grups assignats els horaris dels quals es solapin

en algun moment.

Mètode de resolució:

```
solve :: int_search(credits, first_fail, indomain_min, complete) satisfy;
```

En aquest apartat donem indicacions al resoledor a l'hora de prendre decisions mentre calcula una solució de l'instància del problema.

Per un costat li indiquem que és un int_search perquè treballarem amb variables del tipus **int**, que es trobaran al vector **credits.** El fet de indicar-li que a l'hora de provar una variable utilitzi l'estratègia first_fail vol dir que sempre triarà primer la variable que tingui el domini més petit. La instrucció **indomain_min** farà que dels valors del domini, sempre s'agafi primer el més petit i **complete** força a fer una cerca completa dins el domini.

D'aquesta manera ens assegurem que l'estratègia que segueix el resoledor és la de triar primer sempre el valor més petit del domini més petit.

En aquest punt s'ha de comentar que l'estratègia de cerca només es seguirà en cas d'utilitzar resoledors de constraint programming natius com G12 o Gecode, mentre que amb el fzn2smt+yices2 no té en compte els criteris de cerca, donat que el resoledor d'SMT (yices2) no en fa cas.

Mostra de resultats:

```
output [ show(solucio[p,a]) ++ " " ++ "\n" | p in idProfessors, a in idAssignatures ] ++ [
show(credits[p]) ++ " " ++ "\n" | p in idProfessors ];
```

Aquí li indiquem al resoledor com i quines dades ens ha de retornar, per poder interpretar-les després des de l'interfície gràfica.

6.2.6.- Creació de fitxer de dades pel programa

Apart del model de resolució, on definim variables, restriccions, estratègies i format de sortida, també hem de crear un altre fitxer per completar l'instància del problema. En aquest fitxer hi seran presents les dades corresponents a les variables declarades en el model, és a dir:

```
int: nprofessors;
int: nassignatures;
int: nfranges;
array[idProfessors] of int: minCredits;
array[idProfessors] of int: maxCredits;
array[idAssignatures] of string: nomAssignatures;
array[idAssignatures] of string: nomGrups;
array[idAssignatures] of int: tipologies;
array[idAssignatures] of int: creditsAssignatura;
array[idProfessors, idAssignatures] of int: capacitacions;
array[idAssignatures, idFranges] of int: horarisAssignatures;
array[idProfessors, idFranges] of int: horarisProfessors;
array[idProfessors,idAssignatures] of 0..1: incompatiblesProf;
array[idAssignatures,idAssignatures] of 0..1: incompatibles;
```

Tal com es pot observar, no s'inclouen les variables de tipus var, donat que són les que es calcularan en temps d'execució.

Un exemple de fitxer de dades és aquest:

```
nprofessors=29;
nassignatures=90;
nfranges=260;
minCredits=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];
maxCredits=[31,31,28,19,34,19,19,19,28,19,13,10,28,19,31,31,25,10,31,28,31,31,19,31,37
,37,37,31,31];
creditsAssignatura=[3,3,3,3,3,3,3,3,6,3,3,3,3,3,5,6,3,3,3,3,3,3,3,3,2,1,1,1,3,3,3,3,3,2,2,3,3,3
,3,3,3,3,3,3,6,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,5,2,3,3,3,3,3,3,3,3,3,
6,3,3,3];
capacitacions=[|0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0|1,1,1,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0.....|]
nomGrups=["Teoria1","Lab1","Lab2","Lab3","Teoria","Lab1","Lab2","Lab3","Teoria1","L
ab1","Lab2","Teoria1","Lab1","Lab2","Teoria1","Teoria1","Lab1","Lab1","Teoria1","Lab1
","Lab2","Teoria1","Lab1","Lab2","Teoria1","Pract1","Lab1","Lab2","Teoria1","Lab1","Te
oria1","Lab1","Teoria1","Lab1","Lab2","Teoria1","Lab1","Teoria1","Lab1","Lab2","Lab3"
,"Lab4","Teoria1","Lab1","Teoria1","Lab1","Teoria1","Teoria2","Pract1","Pract2","Pract3"
,"Pract4","Lab1","Lab2","Lab3","Lab4","Lab5","Lab6","Lab7","Teoria1","Teoria2","Lab1
","Lab2","Lab3","Lab4","Lab5","Lab6","Lab7","Teoria1","Lab1","Lab2","Lab3","Teoria1"
,"Lab1","Lab2","Teoria1","Lab1","Teoria1","Lab1","Teoria1","Lab1","Teoria1","Lab1","L
ab2","Teoria1","Lab1","Teoria1","Lab1","Lab2","Lab3"];
```

```
nomAssignatures=["Algorísmica-I","Algorísmica-I","Algorísmica-I","Algorísmica-
I","Algorísmica-II","Algorísmica-II","Algorísmica-II","Algorísmica-II","Bases de
Dades","Bases de Dades","Bases de Dades","Conceptes Avançats de Sistemes
d'Informació","Conceptes Avançats de Sistemes d'Informació","Conceptes Avançats de
Sistemes d'Informació","Eines Lògiques i Problemes Combinatoris","Enginyeria del
Software I","Enginyeria del Software I","Enginyeria del Software II","Enginyeria del
Software: Disseny","Enginyeria del Software: Disseny","Enginyeria del Software:
Disseny"..... "Teoria d'Autàmats i Llenguatge Formal"];
tipologies=[0,2,2,2,0,2,2,2,0,2,2,0,2,2,0,0,2,2,0,2,2,0,2,2,0,1,2,2,0,2,0,2,0,2,2,0,2,0,2,2,2,2,
0,2,0,2,0,0,1,1,1,1,2,2,2,2,2,2,2,0,0,2,2,2,2,2,2,2,0,2,2,2,0,2,2,0,2,0,2,0,2,0,2,2,0,2,0,2,2,2,];
incompatibles=[|1,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0
,0|0,1,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0....0,0,0,0,0,0,0,0,1,0,0,0,0,0,1,0|0,0,0,0,0,0,0,0,0,0,0,0,0
,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,
0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1|];
incompatiblesProf=[|1,1,1,1,1,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1
,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1|0,0,0,0,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,............,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0|0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0|];
horarisAssignatures=[|0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0..............0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0|];
horarisProfessors=[|1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1..........1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1|];
```

*S'ha eliminat part del fitxer donada la seva extensió.

Resoledors utilitzats

6.2.7.- Resoledor G12

Desenvolupat pel NICTA (Institut Australià de Recerca, la institució més gran dedicada a la recerca en Tecnologies de la Informació a Austràlia).

El projecte de desenvolupament d'una plataforma per la programació amb constants va sorgir per afrontar problemes combinatoris d'optimització molt complexos, derivats de la indústria. Aquesta nova manera de processar la informació ha esdevingut una revolució i ha permès a organitzacions de totes mides capturar i treballar amb les dades que disposen i tenir-les en compte a l'hora de prendre decisions. La recerca es divideix en quatre parts:

– Construcció de programes amb modelitzacions més potents.
– Crear nous mètodes de cerca i resolució més eficients.
– Llenguatge més ric per modelitzar problemes.
– Entorn més potent de resolució de models.

El sistema està basat i utilitza la programació amb restriccions, que permet crear fàcilment models per resoldre problemes de manera eficient. D'aquesta manera el temps de desenvolupament i de càlcul es poden reduir dràsticament.

URL: http://www.nicta.com.au/research/projects/constraint_programming_platform

6.2.8.- Resoledor Fzn2smt+Yices2

És una eina molt potent, desenvolupada pel departament d'Informàtica i Matemàtica Aplicada de la Universitat de Girona, que s'encarrega de compilar un fitxer del format Flatzinc al format SMT-LIB (estàndard sorgit d'una iniciativa internacional destinada a la recerca de mètodes per comprovar la satisfacibilitat de certs tipus de formules de primer ordre, en general les que no tenen quantificadors i tenen àtoms de certes teories com la d'aritmètica entera). Va ser dissenyada amb la idea de provar la tecnologia SMT fora dels camps de la verificació, els quals eren el seu objectiu principal. És compatible amb totes les estructures de dades i constraints del Flatzinc, i ell mateix determina el mètode més adequat en el moment de fer la traducció del format Flatzinc al SMT. Un cop feta la traducció, és necessari que interactuï amb un resoledor SMT, on nosaltres hem utilitzat el Yices 2, desenvolupat pel Laboratori de Ciències de la Computació del SRI (http://www.sri.com/).

En el nostre cas les fórmules de primer ordre restringida utilitzen predicats d'aritmètica lineal, per la qual cosa en serà molt útil per a la resolució d'instàncies del nostre problema.

URL: http://ima.udg.edu/Recerca/ESLIP/fzn2smt/index.html

6.2.9.- Resoledor Gecode

És un conjunt d'eines a partir de les quals es desenvolupen sistemes i aplicacions basades en restriccions. Conté un resoledor lliure, gratuït, eficient i amb capacitat per multithreading (diferents execucions concurrents).

URL: http://www.gecode.org/

7.- Eines utilitzades

Pel desenvolupament del projecte s'han utilitzat diferents eines, que detallem tot seguit:

- Java: Es valora per ser multiplataforma i per la seva potència a l'hora de treballar amb objectes i estructures de dades complexes, a més de la possibilitat de crear interfícies gràfiques adequades per l'usuari final.
- MySQL: S'utilitza aquest sistema gestor de base de dades per les interaccions amb l'interfície gràfica i s'ha triat per la seva llicència lliure, per ser present a diferents sistemes operatius i per la facilitat que proporciona a l'hora de crear bases de dades
- Netbeans: S'ha utilitzat per la programació de l'interfície gràfica i pel desenvolupament de l'aplicació en general.
- Vim: S'utilitza per editar el model genèric del minizinc.

8.- Proves i Resultats

A l'hora de fer tests del correcte funcionament i del rendiment de l'aplicació es va començar treballant amb una instància consistent en 10 professors i 30 grups d'assignatures, amb els seus corresponents horaris, disponibilitats i capacitacions. Amb el model inicial existien problemes de rendiment utilitzant els resoledors g12 i gecode (trigaven hores), mentre que amb l'eina fzn2smt+yices2 el temps d'execució era acceptable (minuts).

A partir d'aquí es va treballar intensament en l'optimització primer i, més tard, amb el precàlcul d'incompatibilitats que ens va permetre simplificar els recorreguts a variables dins l'execució del resoledor, aconseguint una millora molt important del rendiment (al voltant d'un 800% més ràpid de mitjana).

Llavors es va agafar un cas més real a partir dels horaris presents a la pàgina web de la Universitat de Girona, del departament d'Informàtica i Matemàtica Aplicada (IMA), compost per noranta grups i utilitzant una trentena de professors ficticis. En aquest cas concret, el fet d'optimitzar les restriccions i precalcular les incompatibilitats va provocar que el resoledor tardés uns 35 segons per trobar una solució a trigar-ne al voltant de 4, utilitzant com a base el resoledor fzn2smt+yices2 (desenvolupat pel departament d'Informàtica i Matemàtica Aplicada de l'UdG i el més ràpid dels tres segons els tests).

Aquest últim test es componia d'una trentena de professors i noranta grups.

Resum de diferents proves:

Resoledor	Nombre Professors	Nombre Grups	Temps requerit
G12	2	10	<1 seg
fzn2smt+yices2	2	10	<1 seg
Gecode	2	10	<1 seg
G12	10	35	1 minut
fzn2smt+yices2	10	35	<1 seg
Gecode	10	35	30 seg
G12	29	90	>1 hora
fzn2smt+yices2	29	90	5 seg
Gecode	29	90	> 1 hora

9.- Dificultats trobades

Durant el modelatge i desenvolupament de l'aplicació s'han trobat diferents dificultats tant de disseny com d'implementació, que s'han resolt de diferents maneres, ajustant-se sempre als principis de genericitat, procurant solucions elegants i fàcils de mantenir.

Primer de tot va aparèixer el fet d'haver de treballar amb franges horàries, la qual cosa va originar una reflexió sobre com afrontar-les. Hi havia diferents alternatives, per un costat creant objectes d'interval, és a dir, si teníem una assignatura dilluns de 09:00 a 11:00, es creava un objecte amb aquesta informació. Per evitar repeticions, s'hauria d'haver intentat reutilitzar objectes, és a dir, grups coincidents compartien objectes. Aquest últim fet portava problemes a l'hora de mantenir els horaris, donat que si modifiquéssim un objecte, quedarien afectats tots els grups relacionats amb aquest objecte. Llavors es va optar per una solució més simple, es defineixen un grup de franges horàries i cada grup en té unes quantes assignades. D'aquesta manera s'obté una flexibilitat màxima, a nivell de segon i pràcticament sense límits.

Seguint amb la part de modelatge, un altre problema va ser el de diferenciar assignatura, assignatura-tipologia i assignatura-tipologia-grup, que clarament segueix una jerarquia i teníem el problema de redundància i dades duplicades, per la qual cosa es va decidir segmentar-ho en tres taules diferents, aprofitant per fer més simples les relacions amb les capacitacions i els horaris.

Un altre problema va ser el fet de trobar un disseny simple i al mateix temps molt potent de cara a la interacció amb l'usuari que, després de diferents esbossos conceptuals, va quedar definit tal com es pot veure actualment, unint per un costat assignatures-tipologies-grups en una base jeràrquica de fàcil interpretació amb els horaris corresponents i els professors capacitats. Apart, a la part dels professors, visualització jeràrquica de les assignatures que tenen capacitades i dels horaris de disponibilitat. S'ha intentat fer-ho de la manera més gràfica i fàcil d'interpretar possible, tractant el tema amb usuaris no tècnics de diferents sectors (educació, periodisme, altre perfil no habituat a l'ús d'ordinadors..) on, després d'explicar-los una mica l'aplicació, podien fer suggerències al respecte, a partir de les quals es va acabar definint el disseny final.

A l'hora d'enllaçar el programa amb els diferents resoledors, va sorgir més d'un problema derivat de l'obtenció de les dades de sortida, especialment en el cas del fzn2smt que, al ser un programa fet en java que s'executava des de la nostra aplicació, no s'acabava d'aconseguir el resultat en un format interpretable. En aquest cas el que es va fer va ser demanar el codi font de l'eina i, a partir d'aquest, crear un objecte fzn2smt amb la funció executar(), la qual escriu en un fitxer el resultat de sortida, des d'on es llegirà el contingut i s'interpretarà per construir la taula d'horaris de

cada professor.

En l'apartat del càlcul de solucions, hi va haver alguns problemes a l'hora de tractar restriccions, com la de disponibilitat horària dels professors o la de no solapament d'assignatures-tipologia-grups en un mateix professor. Els diferents models que vam anar provant eren massa costosos en temps i recursos de memòria, la qual cosa era un handicap molt important a l'hora de treballar amb models reals donat que, a partir de cert volum de dades, els càlculs s'eternitzaven. Es va treballar intensament en aquest sentit, optimitzant el model, eliminant redundàncies i, en última instància, que ha suposat la millora més important, precalculant algunes dades. El fet d'afegir el precàlcul dels grups incompatibles entre si i amb professors ha fet augmentar el rendiment al voltant d'un 800% (segons casos) respecte al model optimitzat.

En la visualització de resultats, davant els problemes d'interpretació i solapaments que sorgien des del punt de vista de les assignatures (sovint hi trobàvem grups solapats o no s'entenia massa bé el resultat per part d'usuaris finals amb els quals he fet proves), es va decidir fer només la part de vista per professor, és a dir, un cop es tria el professor, es mostren les assignacions docents derivades de la solució calculada.

De cara a la visualització “off-line” de les dades, es van tenir en compte diverses opcions, fitxers csv, pdf's, html.. i es va decidir donar l'opció de crear un fitxer pdf amb l'horari assignat a un professor en un cert quadrimestre utilitzant la llibreria iText, és a dir, del que es visualitza cada moment a la pantalla a l'hora de consultar resultats. Es va concretar aquest punt en base a que el pdf és un format molt estandaritzat i apte per a la visualització i impressió.

10.- Reflexions i possibles millores

Arribats al final del projecte, és moment de veure el còmput global de fins on s'ha arribat i cap a on es podria encaminar per acabar d'absorbir les possibles necessitats futures de l'usuari final.

Per un costat hem aconseguit donar una funcionalitat bàsica que cobreix les necessitats que pugui tenir un centre docent a l'hora d'assignar professors a l'horari ja definit, tenint en compte intervals de crèdits, disponibilitats i capacitacions dels professors i els horaris dels grups. Apart de trobar una solució que satisfà les diferents restriccions buscades, hem aconseguit fer-ho en un temps raonable gràcies a les diferents optimitzacions i precàlculs que s'han dut a terme.

A partir d'aquí, hi ha diferents restriccions que es poden afegir en una possible segona part del projecte, passant de ser una resolució de cara a trobar una sola solució a ser una resolució enfocada a trobar la millor solució segons els valors que agafin certes variables. Serien restriccions 'toves' o de preferència, en comptes de dures o d'obligació.

Algunes d'aquestes restriccions podrien ser per exemple:

- Agrupar la docència d'un professor en el menor nombre de dies possible.
- Procurar que es vagin repetint les assignacions d'altres anys, és a dir, si un professor té docència al grup Lab1, que l'any que ve en torni a tenir si és possible.
- Evitar que dos professors coincideixin en una assignatura.
- A partir d'una solució, intentar trobar-ne una altra limitant el nombre de canvis d'assignació, és a dir, que entre les dues solucions no hi hagi més canvis que els que nosaltres determinem.

A més a més, es pot estudiar la inclusió de nous resoledors com SCIP, JaCoP o altres que estiguin basats en l'estàndard del minizinc/flatzinc, per poder comparar temps d'execució i utilitzar un o altre segons les necessitats.

11.- Experiència personal amb el projecte

Aquest projecte m'ha servit per aprendre una manera nova de resoldre els problemes basada en les restriccions, en comptes de centrar-me amb la programació imperativa que havia utilitzat fins ara. És una opció molt potent i, sobretot, molt flexible a l'hora de definir models per a resoldre problemes molt diferents d'una manera ràpida i optimitzada. A més a més, l'avantatge de poder parametritzar les dades d'entrada fa que es puguin utilitzar models genèrics a partir dels quals es poden resoldre tantes instàncies del problema com desitgem.

A més a més ha servit per conèixer el desenvolupament d'un projecte universitari des del principi, començant per la planificació, presa de decisions, implementació, proves i per últim, redacció de la memòria del mateix.

El fet d'haver de superar certes dificultats m'ha fet aprendre altres punts de vista a l'hora d'afrontar certs problemes, apart de la possibilitat que m'ha brindat aquest projecte de conèixer noves eines sobre les quals tenir una base on desenvolupar solucions ha estat una experiència molt enriquidora i útil.

Per últim, haver de planificar i ajustar el temps de desenvolupament organitzant la feina m'ha servit de cara a possibles projectes, per poder tenir una base a partir de la qual temporalitzar bé tasques a realitzar durant un període mitjà o llarg de temps.

En general puc valorar com a molt positiva l'experiència i els coneixements assolits.

12.- Conclusions

Aquest projecte ha estat fruit d'un problema real sobre el que s'ha intentat trobar una solució elegant, utilitzant diferents eines de manera que la resolució fos el més eficient possible. A partir dels requisits inicials s'ha construït una interfície gràfica senzilla però al mateix temps potent des d'on l'usuari pot introduir les dades necessàries per a la resolució de la seva instància del problema.

Un cop assentades les bases del problema, s'ha decidit utilitzar un mètode declaratiu amb restriccions per poder disposar de models genèrics sense preocupar-nos dels algoritmes de resolució, donat que serà el resoledor que utilitzem el que s'encarregarà del càlcul.

En tot moment s'ha intentat mantenir el disseny simple de cara a l'usuari final per així fer l'aplicació més intuïtiva i que la corba d'aprenentatge sigui lo menys pronunciada possible.

A més a més s'ha tingut en compte la possible adaptació i ampliació del problema per a possibles actualitzacions futures, procurant que siguin el menys traumàtiques i costoses en temps possibles.

13.- Bibliografia

Manual minizinc: http://www.hakank.org/minizinc/

Nicholas Nethercote, Peter J. Stuckey, Ralph Becket, Sebastian Brand, Gregory J. Duck, Guido Tack: MiniZinc: Towards a Standard CP Modelling Language. CP 2007: 529-543

Kim Marriott, Nicholas Nethercote, Reza Rafeh, Peter J. Stuckey, Maria Garcia de la Banda, Mark Wallace: The Design of the Zinc Modelling Language. Constraints 13(3): 229-267 (2008)

Dubtes relacionats implementació Java: http://stackoverflow.com

API Java: http://download.oracle.com/javase/6/docs/api/index-files/index-1.html