Treball de Fi de Grau

GRAU D'ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques
Universitat de Barcelona

GENERADOR D'HORARIS DEL GRAU D'ENGINYERIA INFORMÀTICA DE LA UB

Mireia Martín Domínguez

Director: Àlex Pardo Fernández
Realitzat a: Departament de Matemàtica Aplicada i Anàlisi. UB
Barcelona, 27 de juny de 2015

Abstract

Pretty much all university students have to enroll in topics with a variety of class groups, all of them assigned to different days, hours and classrooms. This means they need to take care of what choices they make, if they want a timetable with no clashes between hours. Because of this, there are many existing tools that help with this task, with the capacity to automatically create timetables with minimum clashes from a set of topics, which are also able to show these results clearly and visually.

This project is born from the necessity to have well-organized timetables. The bachelor's degree in Computer Engineering from the University of Barcelona has no helpful tool for this task, so, as a student from this degree, I set out to make the following: a web application that can work with timetable data from all topics, calculate the best group combinations according to student criteria, and show the results visually, as well as manual group customization. This degree final project is about the analysis, design and implementation of this goal.

This web application is based on AngularJS, a JavaScript framework that is fully client-side. It also features front-end technologies like Bootstrap, LESS, CSS3 and HTML5.

Resum

Gairebé tots els estudiants universitaris s'han de matricular d'assignatures amb grups que tenen hores de classe diferents, i per això han de tenir cura amb les eleccions que fan, per tal d'aconseguir un horari ben organitzat i sense solapaments. Per això existeixen eines que ajuden amb aquesta tasca, amb la capacitat de formar horaris automàticament, amb el mínim nombre de solapaments, a partir d'un conjunt d'assignatures, i de mostrar el resultat de forma clara i visual.

El projecte neix d'aquesta necessitat de tenir un bon horari lectiu. El grau en Enginyeria Informàtica de la Universitat de Barcelona no té cap eina que ofereixi aquesta ajuda, de manera que, com a estudiant, m'he proposat el següent: una aplicació web, que treballi amb les dades horàries de cada assignatura, calculi les millors combinacions de grups segons els criteris de l'estudiant, i les mostri visualment, i a més permeti escollir grups manualment. Aquest treball final de grau consisteix en l'anàlisi, disseny i implementació d'aquest objectiu.

L'aplicació està programada en AngularJS, un framework de JavaScript d'execució completament client-side. També fa ús de tecnologies front-end com Bootstrap, LESS, CSS3 i HTML5.

Sumari

1. INTRODUCCIÓ 5
1.1 Context del projecte 5
1.2 Motivació 5
1.3 Estat de l'art 6
1.4 Estructura de la memòria 9

2. OBJECTIUS 10
2.1 Objectius generals del projecte 10
2.2 Objectius específics de l'aplicació 10

3. PLANIFICACIÓ 11
3.1 Planificació inicial 11
3.2 Planificació real 12

4. DESENVOLUPAMENT 14
4.1 Precedents 14
4.2 Disseny de l'aplicació 16
4.2.1 Casos d'ús 16
4.2.2 Maquetació 19
4.2.3 Lectura de les dades 21
4.2.4 Tecnologies emprades 22
4.2.5 Arquitectura de l'aplicació 25
4.3 Implementació del codi 27
4.3.1 Inicialització 27
4.3.2 Actualitzador de la URL 28
4.3.3 Tria d'assignatures 29
4.3.4 Tria manual de grups 33
4.3.5 Generador d'horaris 35
4.3.6 Visualitzador d'horaris 38
4.3.7 Format de les dades 42

5. PROVES I RESULTATS 49

6 CONCLUSIONS I TREBALL FUTUR..... 50
6.1 Conclusions 50
6.2 Possibles ampliacions i treballs futurs 50

7 ANNEX..... 52
7.1 Contingut de l'entrega 52
7.2 Instal·lació i execució de l'aplicació 53
7.3 Glossari de termes i conceptes..... 54

8 BIBLIOGRAFIA 55

1. INTRODUCCIÓ

1.1 Context del projecte

Per a facilitar l'aprenentatge i l'obtenció d'un títol acadèmic, la majoria d'universitats ofereixen una certa llibertat d'elecció amb les assignatures del grau que s'estigui cursant. En el cas de la Universitat de Barcelona, és possible que un estudiant esculli el nombre d'assignatures que vulgui cursar cada semestre, sempre que aquest nombre estigui entre un mínim i un màxim ja establerts, i que pugui triar l'ordre de les assignatures que matricula durant la carrera. També és freqüent que els ensenyaments universitaris tinguin una quantitat de crèdits optatius, que diversifica el nombre i tipus d'assignatures que es poden escollir. A més, per a facilitar l'assistència a classe i per a millorar la qualitat de l'ensenyament, la majoria d'assignatures tenen diversos grups de docència, que generalment es realitzen a hores i aules diferents, augmentant encara més les opcions per a l'alumne.

Aquesta llibertat d'elecció que té l'estudiant queda manifesta a l'hora de realitzar la matrícula. És en aquest moment que haurà d'escollir unes assignatures i uns grups concrets, i haurà d'assistir només a les classes a les que s'ha matriculat. Per tant, és important que tingui clar en tot moment els llocs, dies i hores en els que tindrà docència, i és desitjable que el seu horari lectiu estigui organitzat de la forma més convenient per a les seves necessitats.

1.2 Motivació

En el cas del Grau en Enginyeria Informàtica de la UB, l'única forma de veure totes les assignatures en conjunt és mirant unes combinacions predefinides, sense la llibertat de triar les assignatures que cadascú consideri més adequades, i on a més no surten totes les optatives que pot cursar. Per a fer-se horaris a mida, l'estudiant haurà de mirar les hores de cada grup de cada assignatura un per un, i calcular manualment quines combinacions de classes pot tenir. Si no ho fa, pot acabar amb un horari que tingui hores solapades, dificultant la seva assistència a classe i el seu rendiment, o amb el temps lliure mal organitzat. El procés de planificació, doncs, és important, i es podria millorar si es fessin aquests càlculs automàticament, personalitzats per a cada alumne.

Com a estudiant del Grau en Enginyeria Informàtica de la UB, penso que seria útil tenir disponible una aplicació que permeti generar horaris a mida, per a planificar les matrícules de la meva carrera, i aquest és el motiu principal que m'ha impulsat a fer aquest projecte. Una altra motivació que tenia era treballar amb tecnologies web, per a practicar una mica més les que ja coneixia i per a aprendre'n de noves.

1.3 Estat de l'art

La necessitat d'organitzar-se els horaris lectius d'una forma clara es pot apreciar fàcilment, i és per això que moltes universitats s'han vist en la mateixa situació que la UB. En aquest apartat s'explicaran algunes de les aplicacions ja existents i s'examinaran els seus punts forts i dèbils, per tal d'avaluar com hauria de ser una solució d'aquest estil per les assignatures del Grau en Enginyeria Informàtica de la UB.

Sense anar més lluny, a la Facultat d'Informàtica de Barcelona tenim una eina bàsica feta per la universitat. A la seva pàgina web surt una llista de totes les assignatures, de les quals podem triar tantes com vulguem. Un cop hem fet l'elecció, podem veure totes les hores lectives de tots els grups de les assignatures que hem escollit, o triar-ne alguns i mostrar l'horari només amb aquests. A les figures 1, 2 i 3 es pot veure l'aspecte que tenen l'elecció d'assignatures, grups i l'horari final.

Figura 1: Part de la pàgina d'elecció d'assignatures a la web de la FIB. "Triar subgrups" porta a la figura 2.

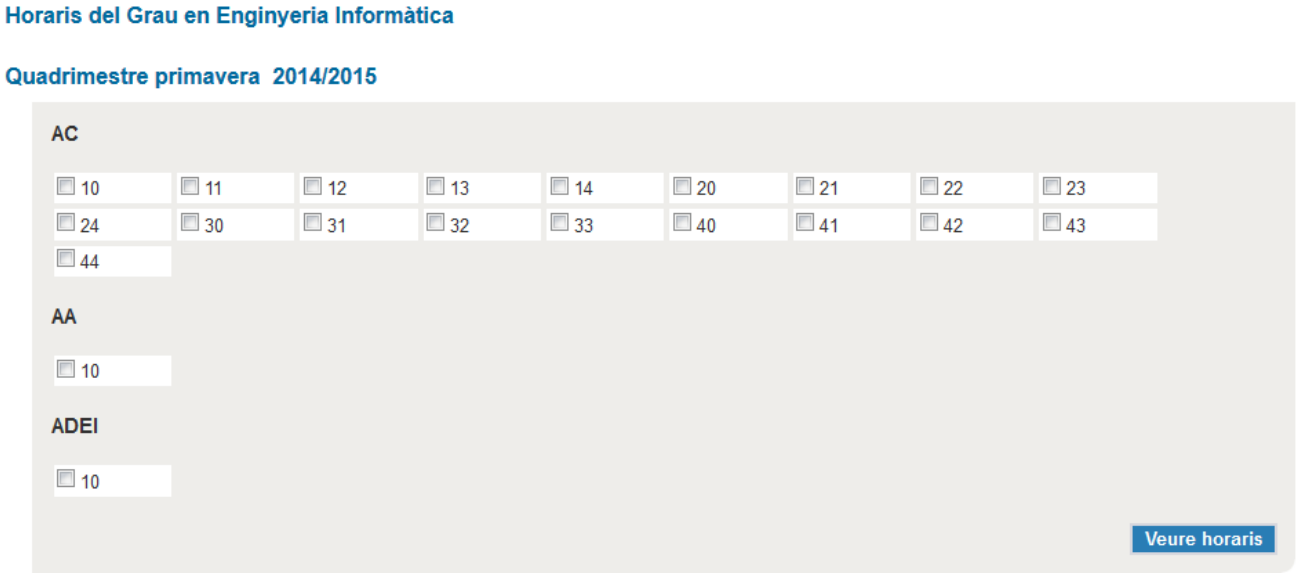

Figura 2: Elecció de grups d'assignatures a la web de la FIB. "Veure horaris" porta a la figura 3.

Horaris del Grau en Enginyeria Informàtica

Quadrimestre primavera 2014/2015

	dilluns	dimarts	dimecres	dijous	divendres
8:00-9:00					
9:00-10:00					
10:00-11:00					
11:00-12:00					
12:00-13:00	EXAMENS (-)				
13:00-14:00	EXAMENS (-)			EXAMENS (-)	
14:00-15:00	EXAMENS (-)			EXAMENS (-)	
15:00-16:00	EXAMENS (-)			EXAMENS (-)	
16:00-17:00	AA 10 (A5106)		AA 10 (A5106)	I adei 10 (A5S113)	
17:00-18:00	AA 10 (A5106)		p aa 10 (A5106)	I ac 42 (C6S303) I adei 10 (A5S113)	
18:00-19:00		ADEI 10 (A5104)		AC 40 (A6202)	
19:00-20:00		ADEI 10 (A5104)	p ac 40 (A6202)	AC 40 (A6202)	
20:00-21:00					

Figura 3: Visualitzador de l'horari triat a la web de la FIB. "Veure horaris" de les figures 1 i 2 arriben aquí.

Aquesta eina permet veure de forma clara el resultat final, així com possibles solapaments. A més, el fet que estigui disponible a una web la fa molt còmoda de fer servir. No obstant, la usabilitat és millorable: per a fer proves, cal anar endavant i enrere en el navegador, ja que la tria d'assignatures, la de grups i la visualització de l'horari no estan a la mateixa pàgina. Finalment, els grups encara cal triar-los manualment, sense cap funcionalitat que calculi automàticament quins són compatibles.

Per a la mateixa facultat podem trobar el *Gnank*. És una aplicació d'escriptori feta per estudiants, que connecta amb la web de la facultat per a obtenir les dades de les assignatures i mostrar-les en un menú, ordenades alfabèticament. Es poden filtrar per grups de matí o tarda, i es pot escollir grups concrets o tots els grups d'una assignatura, per a després visualitzar-los en un panell que hi ha al costat. A més, també es pot configurar el nombre de solapaments permesos, i cercar horaris que compleixin aquest requisit. A la figura 4 es pot veure l'aspecte d'aquesta aplicació.

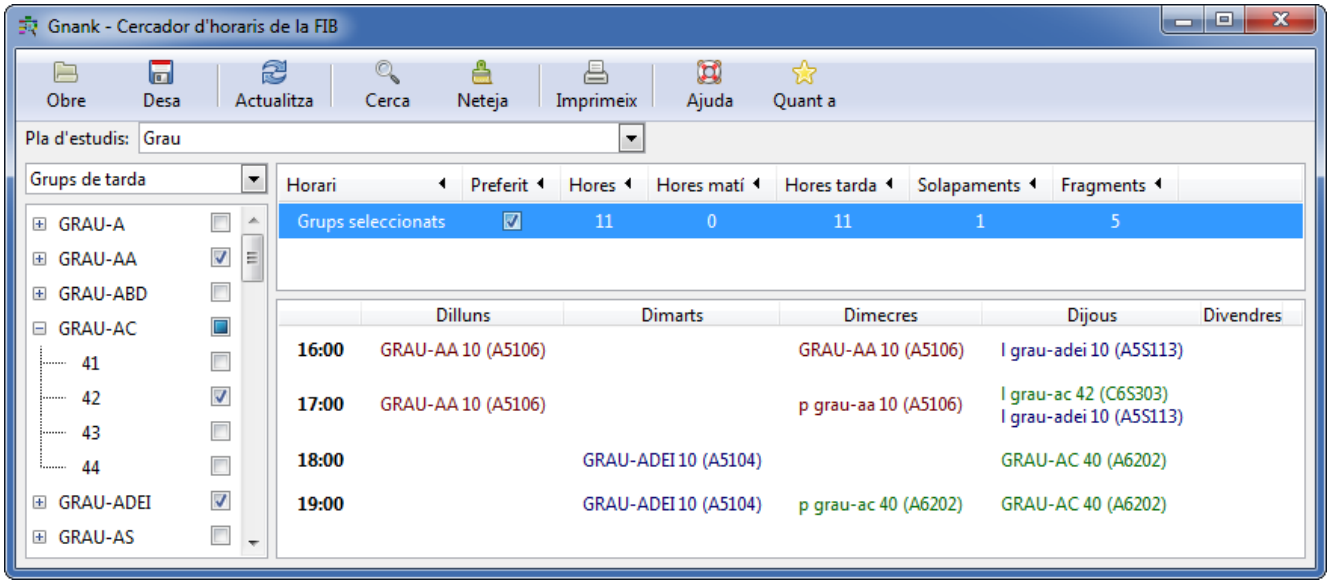

Figura 4: Aspecte de l'aplicació d'escriptori Gnank. Assignatures a l'esquerra, horari resultant a la dreta.

El fet que sigui una aplicació d'escriptori que només funciona en Windows i Linux la fa menys accessible que la web de la FIB. No obstant, té l'avantatge d'una interfície més pràctica: es pot anar triant assignatures i grups i veure els resultats immediatament, de manera que és més fàcil fer proves per a trobar una bona combinació. A més, té la funcionalitat (al botó de Cerca) de calcular horaris automàticament a partir de les assignatures: es pot escollir el nombre màxim de solapaments permesos, de forma que no calgui fer proves manuals per a trobar un horari que minimitzi aquest problema.

Existeixen moltes més aplicacions pensades per ajudar a planificar horaris. Una cerca a Google mostra moltes universitats del món que tenen eines similars, adaptades als seus ensenyaments. Els trets comuns, però, solen ser els mateixos o similars: tenen les dades de les assignatures, permeten fer una selecció entre elles i els seus grups (manualment, i de vegades automàticament), i mostren els resultats de forma visual. Per tant, la nostra aplicació, adaptada a la UB, també haurà de tenir aquests trets, tant en funcionalitats com en aspecte, i possiblement tenir altres millores. S'explicaran en més detall més endavant, a l'apartat d'objectius.

1.4 Estructura de la memòria

En aquest apartat s'explicarà el contingut d'aquest document, que detallarà el procés de desenvolupament d'aquest Treball Final de Grau.

El capítol 1 ha començat amb el context i la motivació que han portat a escollir aquest TFG, deixant clara quina necessitat es vol adreçar, i en l'estat de l'art s'han estudiat algunes solucions que existeixen actualment per a cobrir-la. Amb aquesta introducció s'ha establert la base per al capítol 2, que definirà els objectius que es volen assolir en la realització d'aquest projecte. Després, en el capítol 3 s'exposarà la planificació per aconseguir aquests propòsits, mostrant el progrés setmana a setmana.

El capítol 4 és el més llarg de tot del document, ja que detalla tot el disseny i desenvolupament del projecte. Primer s'ensenya la base que s'ha fet servir, sobre la qual s'ha construït l'aplicació final, i els apartats següents expliquen exactament com aquesta versió ha estat construïda. Es detalla quin comportament ha de tenir, com s'ha dissenyat l'aspecte visual i la lectura de dades, quines tecnologies fa servir i com està organitzada la seva arquitectura. Finalment, la implementació va pas per pas per cada funcionalitat de l'aplicació, explicant què fa i com ho fa, quins mecanismes del llenguatge fan servir i com es comuniquen entre sí. Conté exemples de codi quan calen, per ajudar a ilustrar el seu funcionament.

El capítol 5 explica el mètode general que s'ha fet servir per a provar l'aplicació, i l'entorn sota el qual s'han dut a terme les proves. També compara els resultats finals del projecte amb els objectius que s'esperaven a l'inici, per avaluar l'èxit del projecte.

En la mateixa línia, el capítol 6 exposa les conclusions a les que s'ha arribat després de la realització del Treball Final de Grau, així com possibles millores i ampliacions que podrien dur-se a terme.

Finalment, aquest document compta amb un annex, on es mostra l'estructura de fitxers de l'aplicació, com s'instal·la i com s'executa. També té un glossari de termes tècnics que s'han fet servir en aquest document. Després de l'annex es llista la bibliografia que s'ha fet servir en la realització d'aquest projecte.

2. OBJECTIUS

Un cop hem vist les causes de la necessitat de tenir horaris a mida, i hem vist com són algunes aplicacions que la cobreixen, podem determinar els objectius que ha de complir el nostre projecte.

2.1 Objectius generals del projecte

Aquest generador d'horaris pretén ajudar a realitzar la matrícula i la inscripció als grups de pràctiques del Grau d'Enginyeria Informàtica de la UB. Per tant, l'objectiu principal és una aplicació que faciliti la vida de l'estudiant, per tal que pugui escollir les combinacions d'assignatures que vulgui, i que obtingui l'horari amb les hores lectives que millor s'adapti a les seves necessitats.

A més, es vol que l'aplicació es pugui integrar en la web de la facultat en un futur, i que sigui fàcilment accessible, ja que s'ha de poder accedir tant en el moment de matrícula com en la planificació anterior. Per tant, serà una aplicació web, ja que un navegador i una connexió a Internet són suficients per accedir-hi, i per facilitar la seva possible integració en la web de la UB. Això permetrà també treballar amb noves tecnologies per al desenvolupament web, que és una de les motivacions del projecte.

2.2 Objectius específics de l'aplicació

Ja hem vist que aquest tipus d'aplicacions tenen una sèrie d'aspectes comuns. Tenint això en compte, en general hauria de tenir les següents funcionalitats:

- Obtenir les dades actualitzades dels horaris de cada assignatura i cada grup
- Oferir totes les assignatures i grups dels quals l'estudiant es pot matricular, i permetre fer una selecció qualsevol entre aquests
- Calcular eficientment les combinacions possibles i escollir-ne les millors, segons els criteris de l'estudiant.
- Representar de forma clara i concisa aquestes combinacions, per a veure a simple vista quins possibles problemes poden sorgir amb cada horari
- Guardar els horaris escollits, per a posterior consulta

A més a més, seria desitjable si el programa apliqués les restriccions de matrícula necessàries de cada estudiant: indicar si les assignatures escollides sumen un nombre inferior al mínim de crèdits, o superior al màxim, si no s'han escollit assignatures pendents per superar, etcètera. En època de matrícula, seria útil si també pogués obtenir les places restants de cada grup en temps real, de forma que si s'han esgotat en un grup concret ho indiqui i generi els horaris tenint en compte aquesta limitació.

3. PLANIFICACIÓ

Aquest projecte va començar amb un endarreriment d'aproximadament un mes en l'inici del treball. Això deixa un període de més o menys 5 mesos per a realitzar tota la feina, tant la seva implementació com la memòria que documenta el seu procés. Tenint això en compte, s'ha fixat la següent planificació par tal d'assolir els objectius fixats.

3.1 Planificació inicial

Inicialment es va proposar programar l'aplicació des de zero, i anar afegint funcionalitats incrementalment. D'altra banda, es va intentar el contacte amb l'administració de la UB per a tenir accés a les dades de les assignatures, ja que altrament la seva obtenció seria amb un mètode més laboriós i menys fiable.

La feina es va planificar amb reunions setmanals amb el tutor. Les primeres setmanes començaven així:

Setmana 1

- Disseny preliminar de l'aspecte visual de la web
- Elecció de llenguatges pel projecte: la base seria en PHP (ja que fer-lo servir milloraria la integració de l'aplicació amb WordPress, per a la web de la UB) i per a fer la aplicació més dinàmica es faria servir JavaScript
- Instal·lació i estudi de WordPress, per facilitar integrar l'aplicació amb la UB

Setmana 2

- Creació de dades d'assignatures de prova per a poder treballar
- Creació d'una pàgina web inicial en PHP, capaç de carregar el contingut de les dades de prova i de mostrar-les en forma de taula horària

Setmana 3

- Estudi de tecnologies basades en JavaScript asíncron: AJAX
- Addició de la funcionalitat d'afegir i eliminar assignatures amb JavaScript

Aquesta planificació no va durar gaire, per dos motius:

- La càrrega de treball de programar el projecte des de zero és bastant alta, i es disposava de menys temps de l'habitual per fer el projecte. A més, no teníem garanties de poder obtenir les dades de les assignatures directament de la UB, cosa que obligaria a fer servir un altre mètode més laboriós que fes aquesta tasca (augmentant més la càrrega de treball), i gairebé impossibilitava obtenir el format de les dades dels estudiants, per a calcular horaris tenint en compte les seves restriccions de matrícula, obligant a deixar de banda aquest objectiu.

- Algunes de les aplicacions que ja existeixen per a generar horaris són molt completes, i moltes són de codi lliure, de manera que es poden modificar. Excepte en la part d'obtenir les dades de la UB, també són molt semblants al que s'espera de la nostra aplicació, tant en funcionalitats com en aspecte.

Per tant, a partir de la setmana 2 vam decidir abandonar el treball que s'havia fet i agafar una de les aplicacions ja existents com a base. Si bé això augmentava el treball d'investigació, ja que caldria estudiar el codi i les tecnologies que fes servir el nou projecte, a la llarga reduiria el temps necessari de programar i provar el codi, cosa que sempre tarda més temps del que es planifica (i en el cas d'aquest projecte hi ha menys temps). A més, aprendre noves tecnologies web és part dels objectius del projecte, de forma que el treball afegit entra perfectament dins dels nostres plans.

3.2 Planificació real

Un cop decidida la base sobre la que treballar, la planificació va canviar de construir el projecte des de zero a analitzar i entendre en profunditat el funcionament de la nova aplicació, per a finalment adaptar-la a les característiques i necessitats del Grau en Enginyeria Informàtica de la UB. Així doncs, el repartiment de la feina va quedar així:

Setmanes 1 i 2

- Estudi de la tecnologia del projecte i el seu funcionament
- No hi ha notícies de la UB per a accedir a la base de dades de les assignatures, de forma que s'estudia un altre mètode (que serà més laboriós): web scraping, que consisteix en l'obtenció automàtica de dades a partir de pàgines web, i és menys fiable que obtenir-les des de la base de dades directament. Es miren llibreries i frameworks que puguin ajudar amb aquesta tasca, com Scrapy o Beautiful Soup
- Es deixa de banda la integració de l'aplicació amb WordPress, el sistema de gestió de continguts on es traslladarà la web de la UB, ja que els llenguatges que el projecte fa servir no són similars i no hi ha prou temps per adaptar-lo

Setmana 3

- Instal·lació de l'aplicació base, fer-la funcionar localment
- Adaptació de l'entorn de desenvolupament al nou projecte
- Estudi del codi i la seva tecnologia, primàriament del front-end
- Primeres modificacions de l'aplicació base: modificació del disseny

Setmana 4

- Estudi del codi i la seva tecnologia, aprenentatge de més llenguatges
- Modificació del front-end, reestructuració de l'aspecte de la pàgina
- Hi ha notícies de la UB i l'accés a la base de dades serà possible: es deixa de banda el web scraping, i la programació de l'obtenció de dades queda a l'espera de que es rebi l'accés
- Disseny del format de dades de les assignatures i els seus grups

Setmanes 5 i 6

- Estudi del codi i la seva tecnologia, aprenentatge de més eines que el projecte fa servir
- Modificació del front-end, millora de la usabilitat amb l'eina Bootstrap

Setmana 7

- Estudi del codi i la seva tecnologia, primàriament del back-end
- Finalització de les modificacions al front-end, aspecte final de la pàgina
- Encara no s'ha rebut accés a les dades de la UB, de manera que es creen manualment els fitxers de les assignatures i els seus grups
- Estructuració de la memòria

Setmana 8

- Lectura de les dades de les assignatures des de l'aplicació
- Escriptura de la memòria

Setmanes 9, 10 i 11

- Integració total de les dades de les assignatures a l'aplicació
- Afegida una funcionalitat que mostra un petit requadre (*tooltip*) quan es passa el cursor per sobre de les assignatures de l'horari, per tal de donar més informació i claredat visual
- Escriptura de la memòria

Setmana 12

- Reunió amb els encarregats de l'accés al GR@D, la base de dades amb les assignatures de la UB: es posa en marxa l'obtenció de les dades, tot i que ja no dóna temps a integrar-les a l'aplicació
- Afegits filtres per a la tria d'assignatures, per a millorar la usabilitat
- Escriptura de la memòria

Setmana 13

- Arreglats errors al codi trobats a última hora
- Finalització i revisió de la memòria
- Entrega del projecte

4. DESENVOLUPAMENT

4.1 Precedents

L'aplicació base que es va triar per a fer el projecte es diu UniBuddy Timetable Planner. És una web d'una sola pàgina, feta per Tobias Wooldridge, que pot generar horaris per a tres universitats australianes. Va ser escollida per les seves funcionalitats:

- Dóna molta llibertat per a l'elecció d'assignatures i permet afegir-les amb comoditat, per facilitar la creació de l'horari resultant.
- La generació d'horaris és molt exhaustiva: ofereix moltes opcions que permeten un grau de personalització molt elevat, i mostra els resultats d'una forma molt clara i intuïtiva.
- L'horari final es mostra amb claredat, amb cura de l'aspecte visual.
- Permet escollir grups manualment per a cada assignatura.
- A la URL de la web es guarda la informació de l'horari, de forma que això permet guardar-lo per a posterior consulta.
- Permet traslladar l'horari escollit a Google Calendar.

A les figures 5 i 6 es veu el seu aspecte:

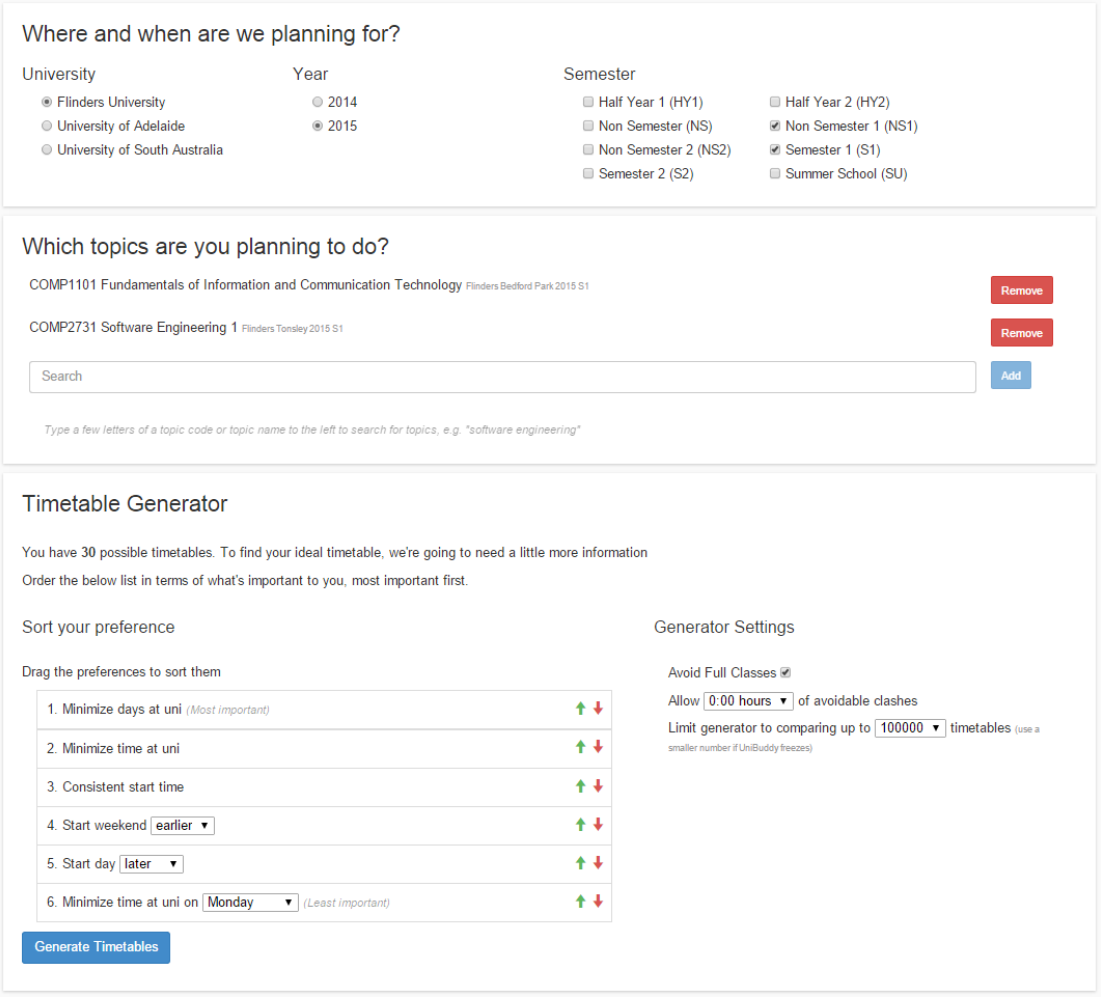

Figura 5: UniBuddy Timetable Planner (I). Surten la tria d'universitat, d'assignatures i la generació d'horaris.

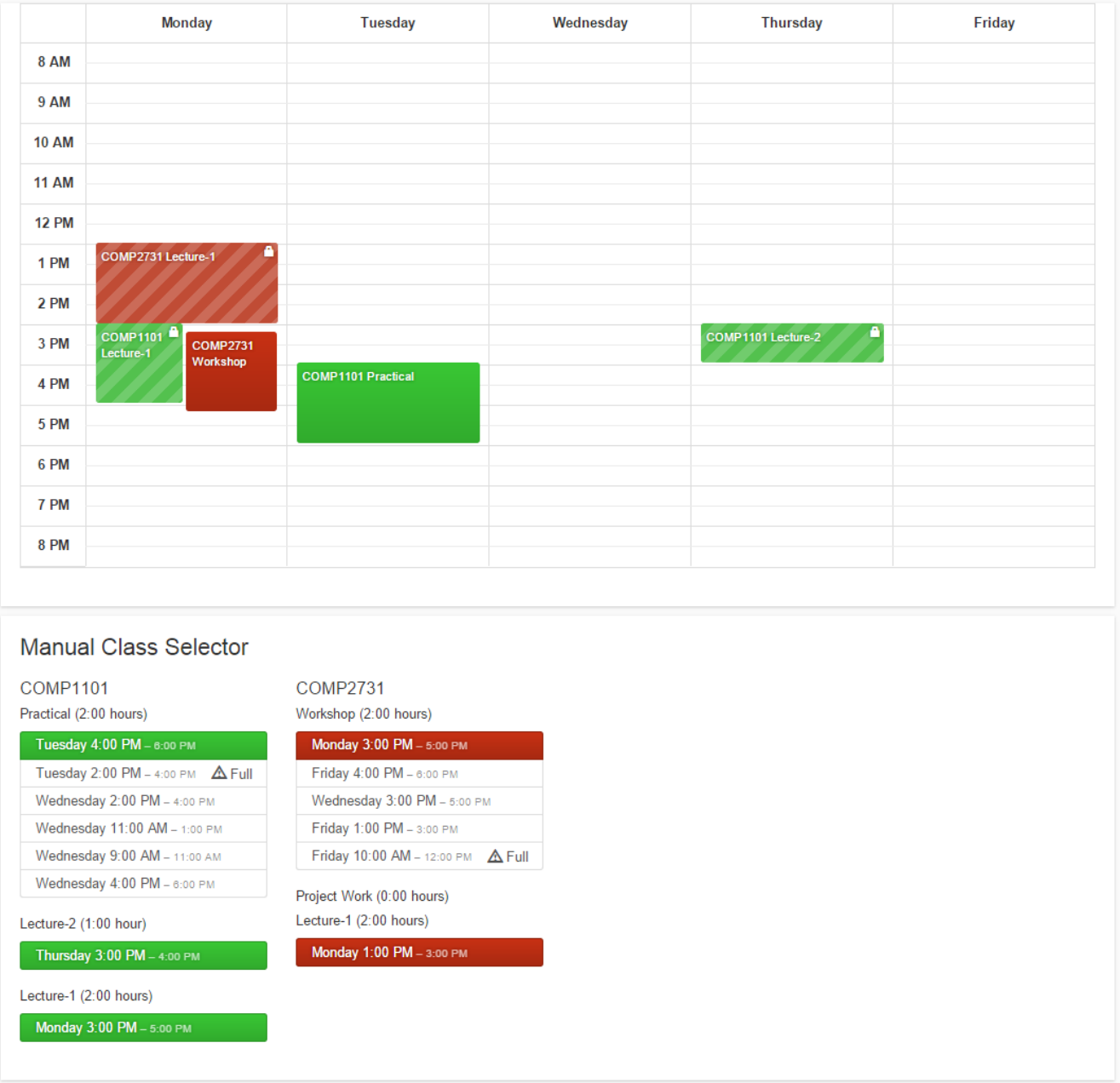

Figura 6: UniBuddy Timetable Planner (II). Surten el visualitzador d'horaris i la tria manual de grups.

Malauradament, la funcionalitat de Google Calendar està en beta i no funciona gaire bé, de forma que no s'ha incorporat al nostre projecte per no donar temps a corregir-la. Tot i així, com es pot veure, és una aplicació molt completa. Assoleix gairebé totes les funcionalitats indicades als objectius específics de l'aplicació, i és de codi lliure, de forma que es pot modificar i adaptar-la a les nostres necessitats.

Les principals modificacions que s'han fet a l'aplicació són una remodelació del disseny visual, i la incorporació de les dades de les assignatures de la UB al seu funcionament, amb les conseqüents modificacions al codi. En els següents apartats es farà un anàlisi de les decisions rere aquestes dues millores, així com del funcionament, l'arquitectura i la implementació detallada de l'aplicació final.

4.2 Disseny de l'aplicació

Aquí es descriuran les decisions que s'han pres per a construir el projecte.

4.2.1 Casos d'ús

Els casos d'ús pretenen explicar d'una forma senzilla les accions que ha de poder fer l'usuari que fa servir un sistema (en aquest cas, l'estudiant que fa servir la nostra aplicació) que té una finalitat concreta o més d'una. És molt útil per a deixar clar quines funcionalitats ha de tenir.

Un generador d'horaris té una única finalitat, que es la creació d'un horari, i tots els esforços del projecte estan centrats en arribar a aquest fi, amb possibles refinaments i millores dels resultats. Per tant, només té un possible cas d'ús. A la figura 7 es pot veure la seva representació, en el llenguatge UML.

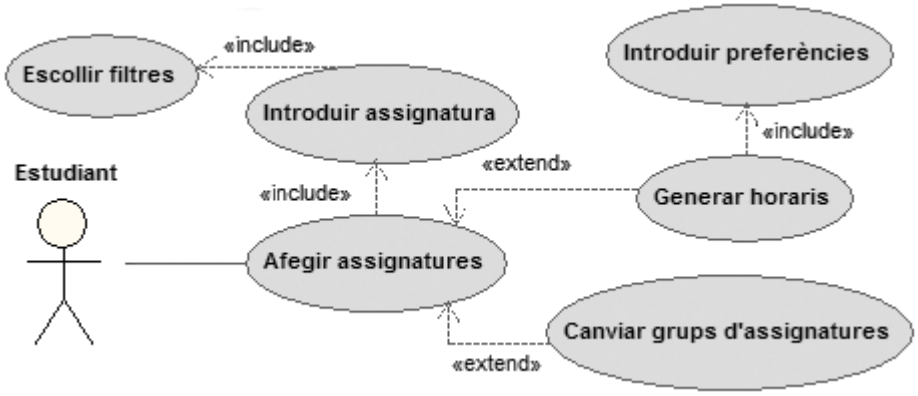

Figura 7: Cas d'ús de l'aplicació: es crea l'horari quan s'afegeixen assignatures, les altres opcions el refinen.

La seva especificació completa seria la següent:

1 Descripció

Aquest cas d'ús descriu com l'estudiant fa servir la nostra aplicació per a obtenir un horari que li sigui convenient.

2 Actor principal

Estudiant del grau en Enginyeria Informàtica de la UB

3 Precondicions

L'estudiant sap de quines assignatures es vol matricular.
El sistema conté les dades de totes les assignatures.

4 Fluxe principal d'events

1. El cas d'ús comença quan l'estudiant ha carregat l'aplicació al seu navegador.
2. Al menú d'escollir assignatures, l'estudiant escull un semestre i un curs d'entre els que s'ofereixen, per a filtrar quines assignatures ha de carregar l'aplicació.

3. L'aplicació carrega les assignatures del grau d'Enginyeria Informàtica de la UB que compleixin les característiques que l'usuari ha marcat i les mostra en pantalla.
4. L'estudiant escriu el nom o abreviació, parcialment o completa, d'una assignatura al quadre de text.
5. L'aplicació mostra la llista d'assignatures que coincideixen amb el nom o abreviació que l'estudiant ha introduit, a partir de la llista d'assignatures que ha carregat al punt 3.
6. L'estudiant escull una assignatura de la llista i apreta el botó d'Afegir.
7. L'aplicació carrega la informació de l'assignatura escollida, l'afegeix a la llista de les assignatures escollides, la mostra al visualitzador d'horaris (amb informació addicional si es passa el cursor per sobre) i actualitza la direcció URL de la web amb les dades de l'assignatura i grups. Si no hi havia assignatures carregades abans, també habilita els menús d'escollir preferències i grups.
8. L'estudiant pot repetir els passos 2, 4, i 6 diverses vegades, fins a arribar al màxim de crèdits permesos.
9. L'estudiant escull una assignatura de la llista d'assignatures escollides i apreta el botó d'Eliminar.
10. L'aplicació elimina la informació de l'assignatura escollida, la retira de la llista de les assignatures escollides, la treu dell visualitzador d'horaris i actualitza la direcció URL de la web amb les dades de l'assignatura i grups que queden. Si no hi ha més assignatures, deshabilita els menús d'escollir preferències i grups.
11. L'estudiant pot repetir el pas 9 diverses vegades, fins a eliminar totes les assignatures escollides.
12. Al menú d'escollir preferències, l'estudiant ordena la llista de preferències, decideix si vol evitar grups plens i solapaments evitables o no, limita el nombre de comparacions que l'aplicació pot fer, i apreta el botó de Generar horaris.
13. L'aplicació compara els grups entre sí, crea diverses possibilitats que compleixen els requisits de l'usuari, i les ordena segons les preferències que l'usuari ha escollit.
14. L'aplicació mostra la llista de possibles horaris.
15. L'estudiant escull una de les opcions i apreta el botó de Mostra.
16. L'aplicació deixa de mostrar al visualitzador d'horaris la informació dels grups que hi havia abans, mostra la informació dels grups del possible horari (amb informació addicional si es passa el cursor per sobre), i actualitza les dades dels nous grups de les assignatures a la direcció URL de la web.
17. Al menú d'escollir grups, l'estudiant obre la llista de grups d'una de les assignatures que ha afegit, i canvia el grup actual per un altre de la llista.
18. L'aplicació deixa de mostrar al visualitzador d'horaris la informació del grup anterior, mostra la informació del nou grup que ha triat l'usuari (amb informació addicional si es passa el cursor per sobre), i actualitza les dades del nou grup triat de l'assignatura a la direcció URL de la web.

19. L'estudiant guarda la direcció URL de la web per a consultar-la en una altra ocasió.

5 Fluxes alternatius

5.1 Assignatures no trobades

Si al pas 3 no hi ha assignatures a l'aplicació que compleixin els requisits de l'estudiant:

1. Es mostra el missatge "No s'han trobat assignatures que compleixin els filtres".

2. Es torna al pas 2.

5.2 Assignatura no trobada

Si al pas 5 no hi ha assignatures que coincideixin amb el que l'estudiant ha escrit:

1. Es mostra el missatge "No s'han trobat resultats".

2. Es torna al pas 4.

5.3 Carregar assignatures des de l'URL

Si al pas 1 hi ha assignatures carregades a la URL de la pàgina:

1. L'aplicació carrega la informació de les assignatures a la URL, l'afegeix a les assignatures escollides, la mostra al visualitzador d'horaris (amb informació addicional si es passa el cursor per sobre) i habilita els menús d'escollir preferències i grups.

2. Es torna al pas 2.

5.4 No eliminació d'assignatures

El pas 9 és opcional si l'estudiant no vol eliminar assignatures.

5.5 No generació d'horaris

Els passos 12, 13, 14, 15 i 16 són opcionals, tot i que en un ús típic es realitzaran.

5.6 No selecció de grups

Els passos 17 i 18 són opcionals, tot i que en un ús típic es realitzaran.

5.7 Sortir sense guardar els horaris

El pas 19 és opcional si l'estudiant no vol conservar l'horari creat.

6 Postcondicions

S'ha creat un horari que segueix els criteris de l'estudiant.

4.2.2 Maquetació

La maquetació és la composició d'una pàgina web, que determina l'estructura, comportament i atributs visuals dels elements que hi han d'aparèixer. Una bona maquetació és molt important, ja que, si està ben feta, pot millorar la usabilitat de qualsevol aplicació, i que aquesta sigui més fàcil i ràpida de fer servir per als usuaris.

Sabent això, considerem l'ús típic de l'aplicació. És important que estigui visible en tot moment l'horari final, ja que és on es mostra el resultat de les accions de l'estudiant. També és important que l'acció que s'estigui fent es mostri a prop del visualitzador, per a que els canvis produïts saltin a la vista. Tenint-ho tot en compte, la composició s'ha organitzat en una sola pàgina de dues columnes, com es veu a la figura 8.

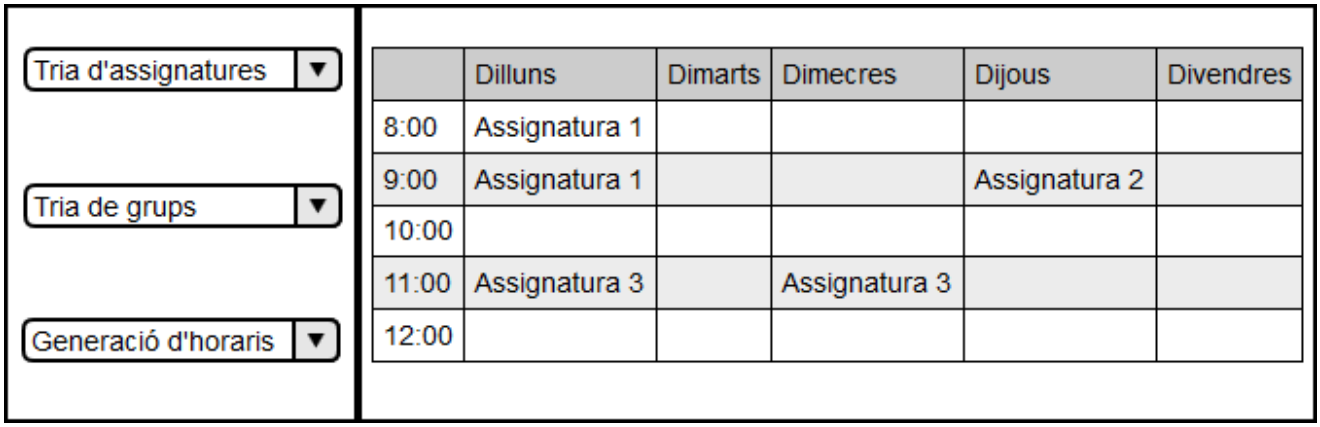

Figura 8: Representació simplificada dels apartats visuals bàsics de l'aplicació.

A la columna de l'esquerra estan totes les accions que pot realitzar l'estudiant, agrupades en tres apartats segons el seu paper: afegir assignatures, triar grups de les assignatures i indicar preferències per generar horaris. Si bé la generació d'horaris és un pas que es faria abans que la tria de grups en la majoria dels casos, és d'esperar que l'estudiant passi més temps en l'ultim, ja que és una tria manual i no automàtica, i per això s'ha col·locat més a dalt. A més, no es faran servir totes les opcions a la vegada, de manera que s'han de poder amagar les que no estiguin actives, per a millorar l'experiència d'usuari. D'altra banda, a la columna de la dreta està el visualitzador d'horaris, sempre visible, per a veure immediatament els resultats de les eleccions de l'estudiant.

Amb aquesta organització en ment, l'aspecte final que ha quedat a la pàgina és el que es pot veure a les figures 9 i 10.

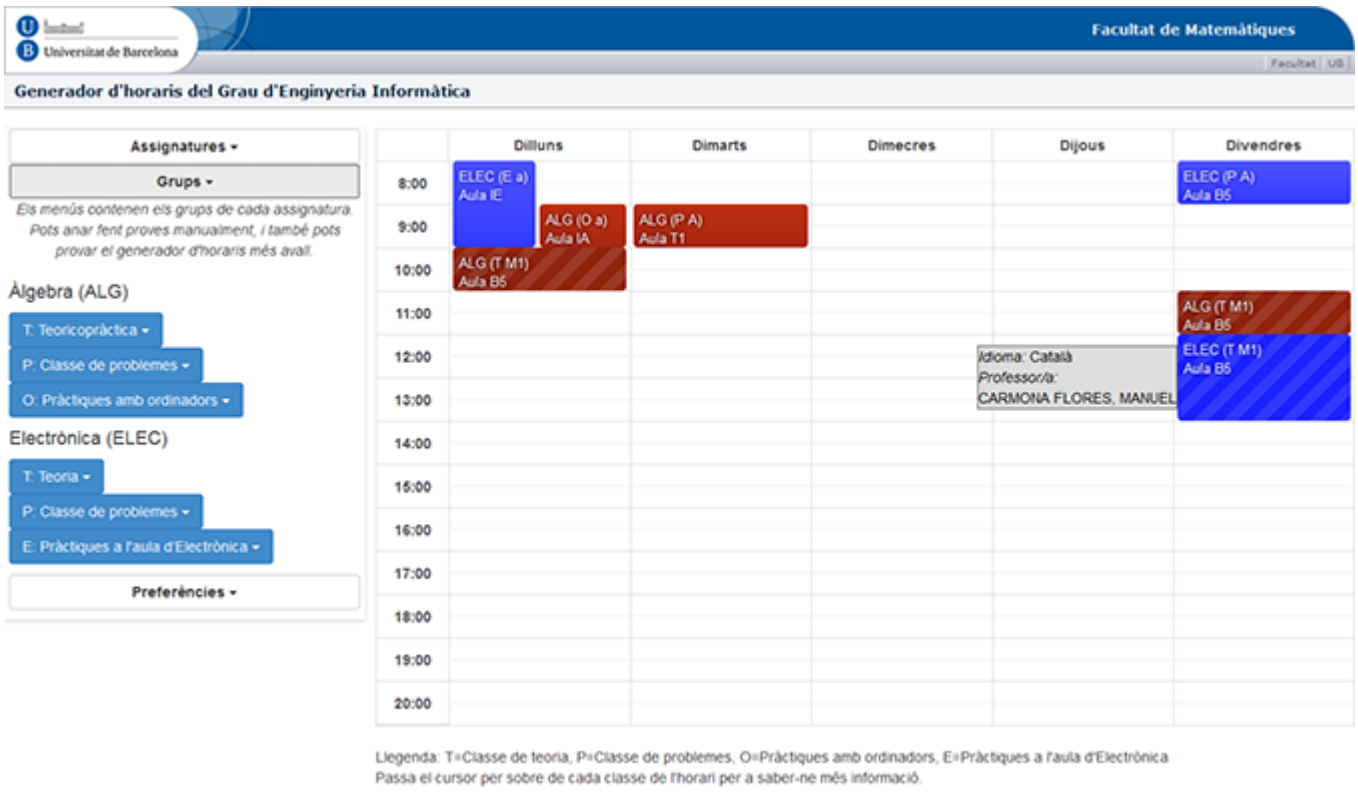

Figura 9: Aspecte final de l'aplicació. El menú que canvia els grups està desplegat, els altres estan plegats. També es mostra la informació addicional que surt al passar el cursor per sobre d'una classe de l'horari.

Figura 10: Els diferents menús, amb les accions d'usuari, desplegats. Es poden observar alguns elements de millora d'usabilitat: per exemple, els grups estan en menús desplegables, per ocupar menys espai.

Finalment, cal dir que aquest projecte s'ha pensat per a ajudar amb la matrícula de la web de la UB, que de moment no està dissenyada amb mòbils i tablets en ment. Per tant, la maquetació d'aquesta aplicació està pensada per a navegadors d'escriptori.

4.2.3 Lectura de les dades

És evident que una aplicació que crea horaris treballant amb dades d'assignatures i grups ha de tenir un accés directe i ràpid a aquestes dades. Per a facilitar aquesta tasca, la seva lectura està pensada en dos passos:

1. S'obtenen les dades de les assignatures de la UB i aquestes es guarden en fitxers, preparats per a ser llegits amb facilitat.
2. La nostra aplicació llegeix els fitxers i extreu les dades, per a processar-les i mostrar-les en pantalla, a disposició de l'estudiant.

Inicialment podria semblar que no calen aquests fitxers intermedis, i que s'hauria de poder llegir les dades directament des del punt de sortida. No obstant, a la llarga és millor aquest procediment, si sabem el cost de cada pas.

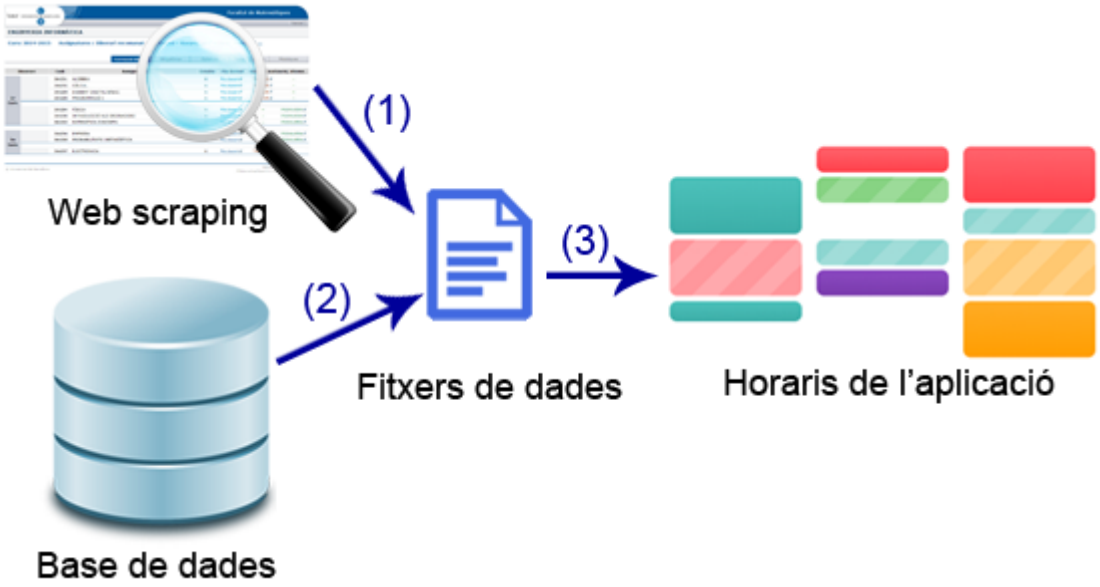

Figura 11: Esquema dels passos possibles per a que l'aplicació llegeixi les dades

Com es veu a la figura 11, tenim els mètodes 1 i 2 per al primer pas, i el mètode 3 per al segon pas. Són els següents:

1. Podem obtenir les dades amb el mètode de web scraping, llegint-les de la pàgina web amb els horaris. Aquest mètode és complicat de programar, ja que les pàgines dels horaris no estan dissenyades per a ser rastrejades amb facilitat, i a més un canvi a la web pot fer que la lectura deixi de funcionar i s'hagi de programar de nou. Funciona si no hi ha alternatives, però no és molt fiable.
2. També podem obtenir les dades connectant amb la BBDD de les assignatures directament. Aquest mètode és millor que el web scraping, però la base de dades no està pensada per a que sigui compatible amb la nostra aplicació, de forma que, si es volen fer moltes consultes, és inferior a tenir fitxers de lectura preparats específicament per a aquest fi.

3. Un cop obtingudes les dades, els fitxers on es guarden ja estan pensats per a que siguin de fàcil lectura, de manera que per a l'aplicació és trivial llegir-los tantes vegades com calgui.

Per tant, els mètodes 1 i 2 son costosos, i el mètode 3 és ràpid i fiable. Si sabem que les dades de les assignatures es mantenen constants durant el curs acadèmic, el que surt més a compte és obtenir les dades una vegada a l'inici del curs per crear els fitxers de dades, i que l'aplicació llegeixi dels fitxers cada vegada que ho necessiti.

Donades les dificultats del projecte, els fitxers amb les dades que l'aplicació llegeix actualment han estat creats manualment. Per a provar l'aplicació és suficient, però a llarg termini surt més a compte fer servir un dels dos mètodes descrits anteriorment.

4.2.4 Tecnologies emprades

Aquest projecte fa servir moltes tecnologies i eines d'última generació, de manera que en aquest apartat s'explicarà breument quines son i en què consisteixen. Val la pena recordar que hi ha un glossari a les últimes pàgines de la memòria, on es descriuen diversos termes tècnics.

AngularJS

AngularJS és un framework de JavaScript, de codi obert i mantingut per Google. Tot el seu codi i llibreries es troben a la part del client, de manera que els resultats usuals de comunicar-se amb un servidor, que solen ser carregar nou contingut o actualitzar parts de la pàgina, es poden obtenir sense la necessitat d'aquesta interacció.

La característica més notable d'AngularJS és coneix amb el nom de *data binding* bidireccional: és una manera de mantenir sincronitzades algunes de les dades internes del model, del back-end de l'aplicació, amb la vista que es mostra en pantalla, de forma que, si es produeix un canvi en una de les dues parts, aquest es veu reflectit en l'altra part automàticament. Això permet reduir considerablement la complexitat del codi.

Tal i com està dissenyat, AngularJS va especialment bé per a aplicacions web d'una sola pàgina, ja que el *data binding* permet actualitzar amb facilitat només parts concretes, i sense haver de connectar amb el servidor. Com s'ha explicat a l'apartat de maquetació, la nostra aplicació és una sola pàgina amb totes les funcionalitats a prop entre sí, de forma que AngularJS és una bona elecció pel projecte. Al voltant del 76% de l'aplicació està escrita en AngularJS, i per tant és la base de la majoria del codi.

HTML5 + AngularJS

HTML és el llenguatge de marques que es fa servir per estructurar i presentar el contingut de totes les pàgines web del món. La versió 5 té noves funcionalitats i permet construir aplicacions web més complexes, de forma que és bona idea fer servir l'última versió al nostre projecte. A més, en el cas de la nostra aplicació, el llenguatge HTML5 estarà ampliat amb marques i atributs personalitzats: és una altra de les característiques d'AngularJS, les anomenades directrius. Al voltant del 15% del codi de l'aplicació està escrit en HTML5, incloent les extensions d'AngularJS.

CSS3 + LESS

CSS és un llenguatge de fulls d'estil que es fa servir per descriure l'aspecte visual dels elements HTML d'una pàgina. La versió 3 té noves funcionalitats i permet un millor control sobre l'estil final, de forma que és bona idea fer servir l'última versió al nostre projecte. A més, està acompanyat per LESS, un preprocessador de CSS que li afegeix noves funcionalitats. LESS permet l'ús de variables i funcions de forma semblant a la d'un llenguatge de programació, i es compila per a que el funcionament final sigui igual al del CSS, cosa que dóna una gran flexibilitat a l'hora de configurar el disseny d'una pàgina. Al voltant del 8% del codi de l'aplicació està escrit en CSS i LESS.

Bootstrap

Bootstrap és un framework per al front-end, i també fa servir LESS. És un conjunt d'eines amb plantilles de disseny, tipografies, formularis, botons i menús, entre d'altres, amb unes característiques que donen facilitats per a la maquetació de pàgines web. També està pensat per a ser *responsive*, i que s'adapti a la mida de la pantalla on es mostra, amb diferents configuracions per a escriptori, tablet i mòbil. Aquest projecte fa servir els seus elements per a millorar la usabilitat de l'aplicació, tot i que les facilitats per a adaptar-se a diferents mides no s'han utilitzat; com s'ha explicat a l'apartat de maquetació, l'aplicació prioritza optimitzar l'experiència en navegadors d'escriptori.

JSON

JSON és un estàndard de fitxers per a guardar dades de forma simple. És lleuger i s'organitza en una estructura jeràrquica, i donat que fa servir notació d'objectes de JavaScript, la seva lectura és trivial per a un framework de JavaScript com AngularJS. És el format que faran servir els fitxers on es guardin les dades de les assignatures del grau en Enginyeria Informàtica de la UB, adaptades per a la nostra aplicació.

Aquests són els llenguatges i frameworks en què s'ha programat el codi de l'aplicació. No obstant, el nostre projecte també té més eines de suport.

Node.js: Bower, Grunt, Karma

Node.js és un entorn de programació server-side basat en JavaScript asíncron. Està dissenyat per a que sigui no bloquejant, de forma que pot atendre un gran nombre de peticions. Aquest projecte no fa servir codi de Node.js per a la aplicació en sí, però sí que fa servir el seu gestor de paquets NPM per facilitar les tasques relacionades amb el desenvolupament. Així doncs:

- **Bower** descarrega els paquets amb les llibreries necessàries per al codi (AngularJS i Bootstrap, principalment), i gestiona les seves dependències.
- **Grunt** automatitza una sèrie de tasques per a preparar el codi per a la seva execució, i s'encarrega d'accions com ara examinar el codi buscant errors, compilar el codi LESS en CSS estàndard, i comprimir el CSS i el JavaScript eliminants caràcters no necessaris (com ara espais i salts de línia) en un procés que es diu minificació. També té un mode d'execució contínua fent servir la tasca ***watch***, que cada vegada que canvia un fitxer automàticament aplica els processos de preparació descrits abans.
- **Karma** és un servidor web que es connecta a un navegador i executa proves unitàries de parts del codi d'AngularJS, indicant de forma clara si passen o fallen. Fa servir PhantomJS, un navegador sense interfície gràfica, per a executar els seus tests. En aquest projecte és part de la rutina d'execució de Grunt.

AngularJS Eclipse + Nodeclipse

Eclipse és un entorn de desenvolupament integrat (IDE) de software, amb diverses funcionalitats com autocompletar codi o integració amb control de versions. Per a aquest projecte l'hem fet servir amb el plugin d'AngularJS Eclipse, per ajudar amb el desenvolupament del codi, i Nodeclipse, per a la configuració. Altres IDEs adequats són Sublime Text o Brackets, que també tenen plugins per a facilitar la programació en els llenguatges d'aquest projecte, però s'ha escollit Eclipse per familiaritat.

Github

Github és un repositori de codi que molts usuaris fan servir per a portar el control de versions del seu software, sota el sistema de Git. Com que la majoria del codi és públic, permet la participació col·laborativa entre programadors, així com la possibilitat de que altres puguin reaprofitar el seu treball. Molts frameworks i utilitats d'aquest projecte es poden trobar a Github, com ara Bootstrap, LESS, Bower o Grunt. També és on es troba el codi base (UniBuddy Timetable Planner) que s'ha fet servir en aquesta aplicació, i és el repositori on s'ha portat el control de versions d'aquest projecte.

4.2.5 Arquitectura de l'aplicació

El patró MVC (Model-Vista-Controlador) és àmpliament utilitzat i conegut. Consisteix en la separació de preocupacions, on cada part s'encarrega d'un àrea del programa i la gestiona independentment de les altres parts. MVC divideix les àrees en dades, presentació i lògica, respectivament, de forma que:

- el model s'encarrega de guardar i manipular l'estat de les dades, oferint eines per al seu ús al controlador, quan les necessiti
- la vista s'encarrega de la representació d'aquestes dades per a l'usuari, el controlador li envia les dades del model per a mostrar-les
- el controlador s'encarrega de les decisions per a comunicar les dues parts: rep les peticions de l'usuari, decideix els passos necessaris que s'han de fer, dóna les ordres necessàries al model, i passa els resultats a la vista

En aquest projecte el codi està organitzat, en línies generals, per a seguir el patró MVC. Per tant, es pot parlar de Model, Vista i Controlador per a explicar l'estructura del codi que fa les funcions descrites. La part que gestiona les tasques relacionades amb la vista està escrita en HTML5, CSS3, LESS, Bootstrap i AngularJS, mentre que les parts que gestionen el model i els controladors són purament en AngularJS.

Dit això, cal indicar que l'aplicació no segueix el patró MVC estrictament, a causa de l'ús d'AngularJS. Gràcies al seu *data binding* bidireccional, els canvis al model es reflecteixen a la vista, i canvis a la vista modifiquen el model, sense que el controlador actuï. A més, la vista pot obtenir dades del model directament. En aquest sentit, es pot dir que el patró al que s'assembla és Model-View-ViewModel (també anomenat Model-View-Binder).

En el cas de Model-View-ViewModel, el model i la vista fan les mateixes funcions, però, en comptes de confiar passivament en el controlador per a que gestioni la seva comunicació, el ViewModel (o model de vista) proporciona destinacions per a l'enllaç de dades per a que la vista hi accedeixi, fent de *binder* entre els dos. En molts casos, el ViewModel exposa el model directament, o proporciona membres que encapsulen elements de model específics. El ViewModel també pot definir elements per a realitzar un seguiment de les dades que són rellevants per a la interfície d'usuari, però no pel model.

Per això es pot dir que AngularJS, a més de MVC, també és MVVM. En aquest cas, el ViewModel és el *binder* que uneix el model i la vista, i el Controlador actua per a afegir lògica en parts compartimentades de vista i model, actuant sobre components concretes d'aquests. A la figura 12 es mostra com queden les relacions.

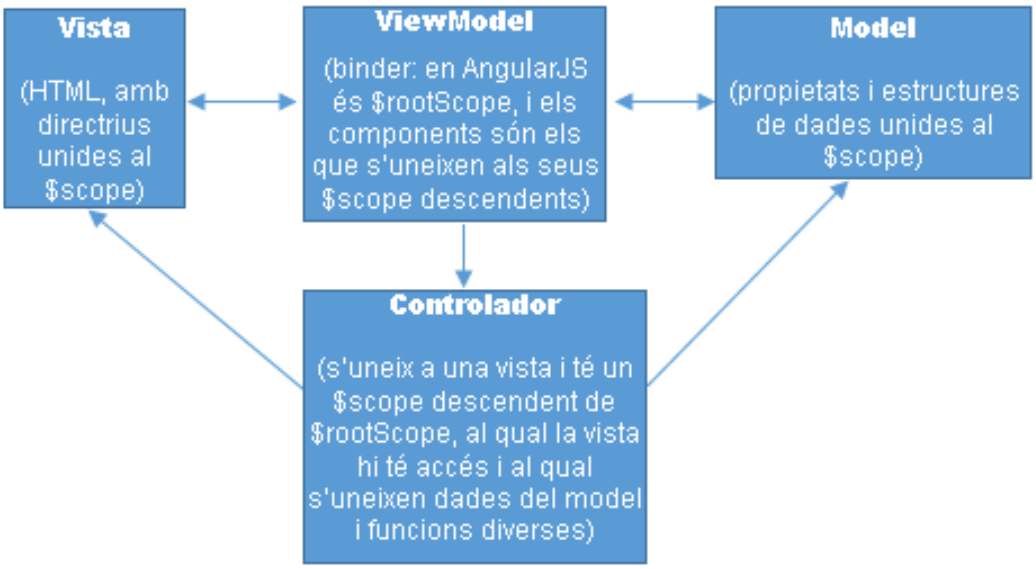

Figura 12: Arquitectura del projecte, híbrida entre el patró MVC i les propietats MVVM d'AngularJS.

Com es pot veure, en AngularJS aquest *binder* és el component `$rootScope`, del qual descendeixen els diversos `$scope` dels components, que es fan servir per unir-los. Parts de la vista hi poden accedir i fer ús de les seves propietats, i s'hi guarden estructures de dades i propietats que corresponen al model. A més, cada controlador té el seu propi `$scope`, que pot lligar a una vista i a unes dades que també pot manipular.

No obstant, si bé el disseny d'AngularJS pot tenir propietats de MVVM que s'estenguin a les aplicacions, és prou flexible per a fer-lo servir amb una organització de MVC sense problemes. Encara que la nostra aplicació tingui algunes excepcions al patró, el seu codi està organitzat en model, vista i controladors, i segueix la separació de preocupacions que s'ha explicat.

4.3 Implementació del codi

En aquest apartat s'explicarà com funciona el codi. Són necessaris certs coneixements d'AngularJS per a entendre el funcionament de l'aplicació, de forma que en els següents apartats també s'explicaran alguns conceptes d'aquest llenguatge.

4.3.1 Inicialització

Donat que AngularJS és qui porta el pes de totes les funcionalitats de l'aplicació, volem que englobi tot el codi que el navegador llegirà. Per tant, si volem que aquest sàpiga posar el projecte a punt, hem d'inserir en *index.html* la següent marca:

```
<html ng-app="horaris"> … </html>
```

L'atribut `ng-app` és una directriu d'AngularJS, i els seus efectes s'apliquen a tot el codi que estigui dins del tag `<html>` fins que es tanqui. Aquesta directriu indica on està l'arrel de l'aplicació, i li diu al script que carrega AngularJS que comenci el *bootstrapping* (la inicialització de components clau) en aquest punt. Alguns d'aquests components clau són `$rootScope`, el *binder* que lliga la vista i el model, o la compilació de directrius incloses dins del tag, per a habilitar el *data binding*.

Així doncs, inicialitzem tot el codi amb `ng-app="horaris"`. Així es carrega a l'arrel el mòdul `horaris`, que es troba a l'arxiu *app.js* i s'encarrega de definir la ruta per defecte de la URL. A més, es pot veure que aquest mòdul té com a dependències els mòduls `'horaris.timetable'` i `'templates-app'`. Els *templates*, en aquest cas, són fitxers amb codi HTML, i que AngularJS permet lligar a les seves directrius, les quals permeten que el codi es pugui inserir en qualsevol lloc i tantes vegades com aquestes indiquin. En aquest projecte, el codi HTML de l'aplicació està en *templates* separats del fitxer *index.html*, que només conté la capçalera i el peu de pàgina de la UB.

Afegint aquestes dependències al mòdul principal podem fer que *index.html* carregui el *template* de la vista de l'aplicació, que es troba dins de *app/timetable/*, si al mòdul `horaris.timetable` (que es troba a *timetable/timetable.js*) posem aquest codi:

```
.config(function config($stateProvider) {
      $stateProvider.state('home', {
           url: '/',
           views: {
           "main": {
               controller: 'TimetableUrlController',
               templateUrl: 'timetable/timetable.tpl.html'
           }
```

```
        }
    });
})
```

i a *index.html* posem el següent:

```
<div ui-view="main"> … </div>
```

Aquest codi fa que es carregui a `main` el *template* que es troba a la ruta que diu `templateUrl`, sota l'efecte del controlador `TimetableUrlController`.

A aquestes altures, si tot va bé, AngularJS s'haurà carregat dins del codi sota la directriu `ng-app`, i haurà carregat els components principals de l'aplicació dins de la directriu `ui-view` sota els efectes del controlador `TimetableUrlController`. L'aplicació ja estarà llesta per a rebre events.

4.3.2 Actualitzador de la URL

Una de les funcionalitats del nostre generador d'horaris és la d'actualitzar les dades dels horaris en la direcció URL de la pàgina, i ser capaç de carregar horaris a partir d'una URL vàlida. Aquesta URL té l'aspecte que es mostra a la figura 13.

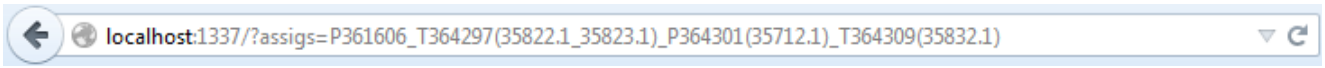

Figura 13: URL de la pàgina amb 4 assignatures carregades, amb informació de grups entre parèntesis.

D'això s'encarrega `TimetableUrlController`, el mateix controlador que vam lligar al *template* de l'aplicació principal. En aquest cas controla la URL de la pàgina, que té les seves dades a `$location`. Cada vegada que `$location` canviï carregarà les noves dades que hi apareguin dins de l'aplicació, i cada vegada que canviïn les dades a l'aplicació guardarà aquestes dades a `$location` per actualitzar la URL. D'això s'encarreguen dos *listeners* que s'han afegit al `$scope` del controlador:

```
$scope.$watch( $location.search(),
               function () {loadFromUrl();} );
```

Per una banda, aquest listener posa en observació a `$location.search()`, que retorna el *query string* de la URL (la part rere el ?), i quan es produeix un canvi fa saltar el codi dins de `function()`, que en aquest cas és una crida a la funció de carregar dades de la URL, que està implementada al controlador.

```
$scope.$on('chosenClassesUpdate', function () { … });
```

Per l'altra banda, aquest *listener* s'activa quan una altra part del codi fa *broadcast* al nom `'chosenClassesUpdate'`. Tant `$watch` com `$on` com `$broadcast`

penjen de `$rootScope`, del qual penjen tots els `$scope` de tots els controladors, de forma que amb aquestes funcions es poden comunicar events entre diferents parts de l'aplicació. Així doncs, quan es produeix un *broadcast* en aquest nom s'executa el codi dins de `function()`, que en aquest cas parseja la informació de les assignatures i els seus grups i construeix l'expressió que es guardarà a la URL.

Per últim, s'ha de dir que tots els controladors d'aquesta aplicació deleguen part de la lògica i de les seves operacions en el que a AngularJS s'anomenen *providers*. En aquest projecte, les anomenades factories s'encarreguen de la construcció de les estructures de dades que fa servir l'aplicació, i els serveis dónen funcions auxiliars, reutilitzables per altres controladors, factories i serveis. En el cas de `TimetableUrlController`, les seves dependències es poden veure a la figura 14.

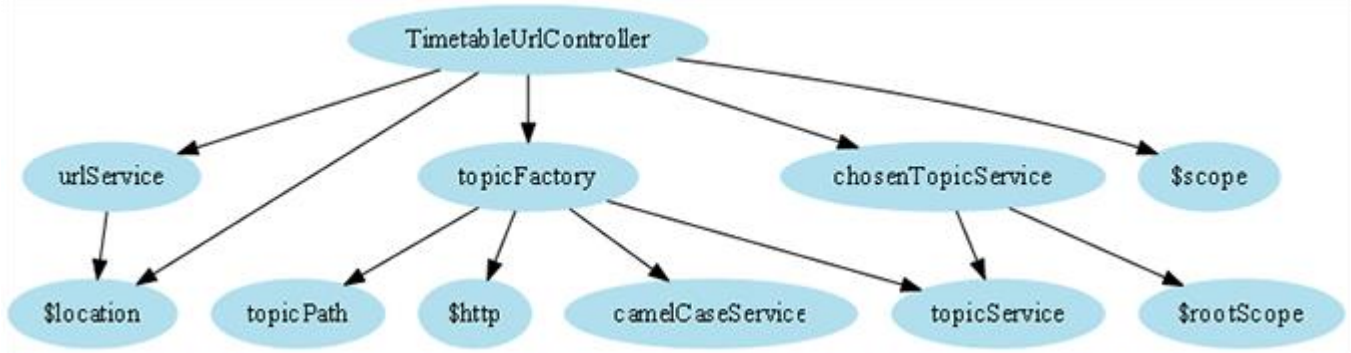

Figura 14: Mapa de dependències de TimetableUrlController.

Podem veure, per exemple, que depèn de `urlService`, qui també depèn de `$location`: això és perquè delega moltes de les operacions de manipulació de la URL a aquest servei. També es poden veure les dependències de `topicFactory` i `chosenTopicService`, que s'explicaran al següent apartat.

4.3.3 Tria d'assignatures

Una de les funcionalitats clau de la nostra aplicació és la d'afegir assignatures, i que aquestes es mostrin en el visualitzador. En aquest cas, el controlador encarregat és `TopicController`, i aquesta vegada anirà dins d'un tag HTML en comptes d'unir-lo a un *template*, de forma similar al que hem fet amb `ng-app`:

```
<div ng-controller="TopicController"> … </div>
```

La directriu encarregada de lligar el controlador directament al codi HTML és `ng-controller`, i tot el codi que estigui dins del `<div>` tindrà accés al seu `$scope`. Recordem que el controlador uneix al seu `$scope` les dades del model, que la vista podrà mostrar. A continuació s'explicaran a fons tant el codi de `TopicController` com el de la vista, per entendre el funcionament d'ambdues parts i com es relacionen.

Començem pel controlador. Les seves dependències es poden veure a la figura 15.

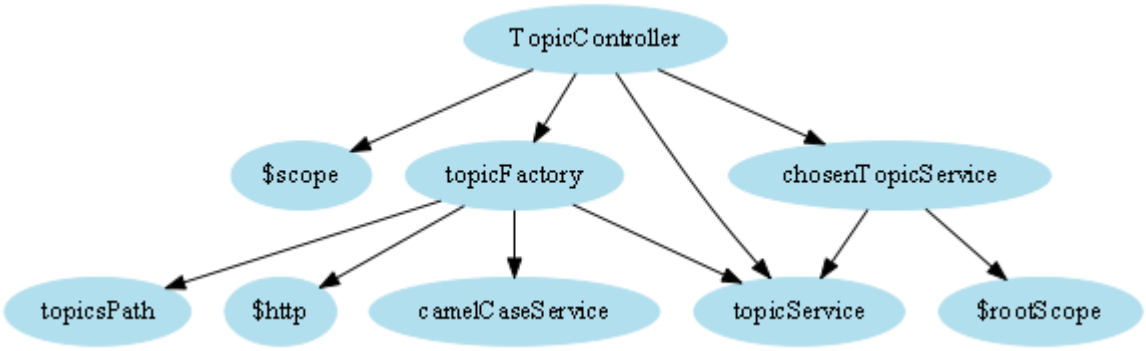

Figura 15: Mapa de dependències de TopicController.

Igual que `TimetableUrlController`, `TopicController` també depèn de `topicFactory` i `chosenTopicService`. Per una banda, `topicFactory` s'encarrega de llegir els fitxers JSON (d'aquí la dependència de `$http`) i de construir les estructures de dades de les assignatures, per a que `TopicController` i `TimetableUrlController` puguin carregar-les i incorporar-les al seu `$scope`. Mentrestant, `chosenTopicService` depèn de `$rootScope`, i això és perquè aquest servei té utilitats que afecten totes les àrees de l'aplicació: aquí es defineixen les funcions de *broadcast* que activen els *listeners* de `TimetableUrlController` o d'altres parts del codi, i fan que l'aplicació actualitzi les dades dels horaris i de la URL. També hi són les funcions que afegeixen o eliminen una assignatura de la llista d'assignatures escollides, per després cridar les funcions de *broadcast* ja esmentades. Altres dependències com `topicService` o `camelCaseService` ofereixen utilitats generals, com ordenació alfabètica d'assignatures o conversió de text a lowerCamelCase.

Respecte a les dades i funcions que s'han afegit al `$scope` de TopicController, per una banda `$scope.semesters` i `$scope.years` contenen els semestres i cursos per filtrar assignatures, mentre que `$scope.activeYear` i `$scope.activeSemester` inicialitzen un valor per defecte que es mostrarà a la vista ja marcat, i que quan la vista el canviï s'hi guardarà el nou valor. Les funcions de `$scope` són les que la vista necessita per poder treballar: `addTopic(topic)` i `removeTopic(topic)`, `updateAvailableTopics()` i `getAvailableTopics()`, `validateTopic(topic)` i `getCredits()`. Totes han de portar `$scope` al davant, o si no serien funcions locals al controlador.

Vista l'estructura del controlador i el model, veiem la vista. El seu aspecte surt a la figura 16.

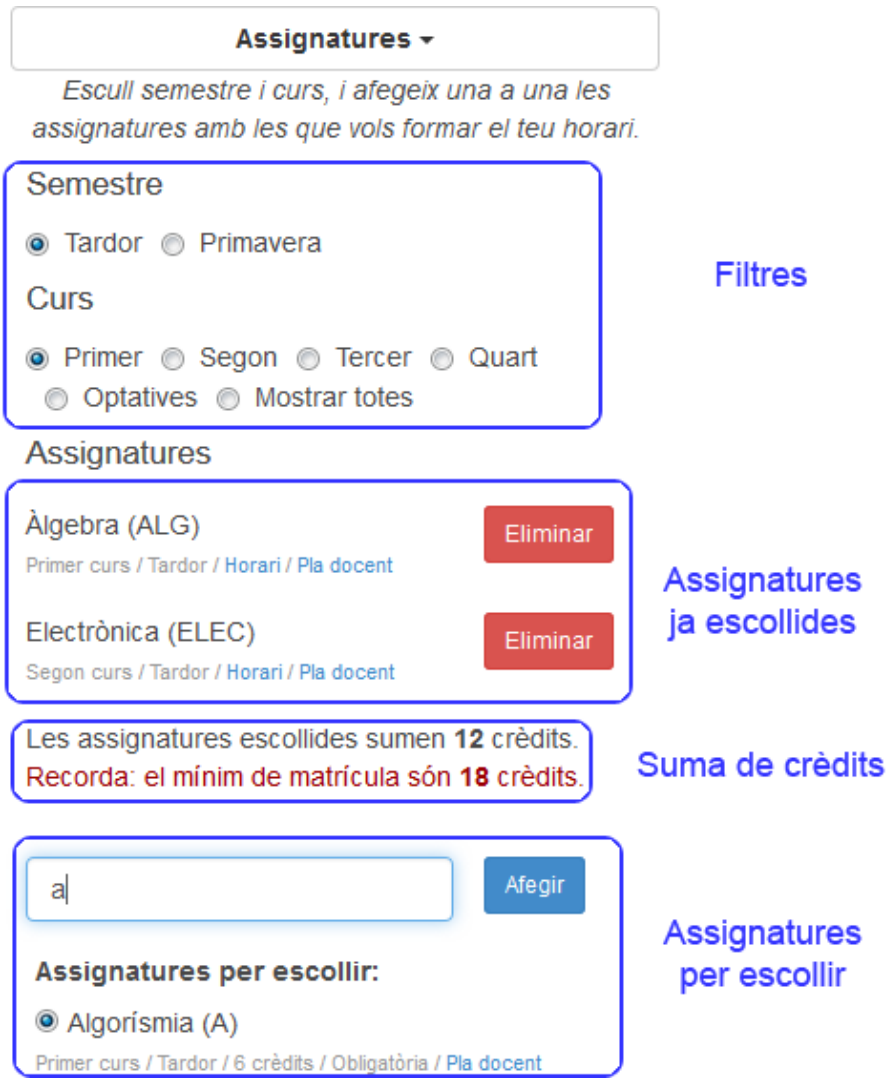

Figura 16: Aspecte de la vista sota TopicController. S'han ressaltat els elements principals.

Els elements principals de la figura 16 són:

- els filtres de semestre i curs, que redueixen la llista d'assignatures per escollir;
- les assignatures ja escollides, que composen l'horari i es poden eliminar aquí;
- la suma de crèdits, que mostra el nombre de credits de les assignatures escollides i si entra dins dels límits de matrícula,
- i les assignatures per escollir, que es poden introduir al quadre de text i es van filtrant conforme s'escriu el nom al quadre de text.

Filtres

El codi HTML dels filtres segueix aquest esquema:

```
<label class="radio-inline" ng-repeat="semester in semesters">
<input type="radio" name="activeSemester" ng-model="$parent.activeSemester"
ng-value="semester" ng-change="updateAvailableTopics()"/>{{semester}}
</label>
```

Tenim noves directrius: `ng-repeat`, `ng-model`, `ng-value`, i `ng-change`:

- La directriu `ng-repeat` permet iterar sobre l'array `$scope.semesters`, i crear una instància del tag on es troba la directriu per a cada element. Això permet instanciar tants *radio buttons* com elements tingui l'array.
- La directriu `ng-model` lliga el valor del *radio button* actiu a `$scope.activeSemester`, per a que el controlador pugui operar sabent quina elecció ha fet l'usuari. Donat que `ng-repeat` crea un `$scope` nou per a cada instància, i que en JavaScript el `$scope` heredat amaga el del pare si té el mateix nom, la directriu `ng-model` fa servir `$parent` per indicar que es refereix al `$scope` del controlador.
- La directriu `ng-value` lliga el valor de cada element de `ng-repeat` a cada *radio button*, de forma que, si un d'ells passa a ser el botó actiu, `ng-model` faci adoptar aquest valor.
- La directriu `ng-change` crida la funció `updateAvailableTopics()` quan el radio button canvia. En aquest cas, la funció crida una de les utilitats de `topicFactory`, per a obtenir les assignatures que compleixen els filtres i carregar les seves dades.

Cal assenyalar també l'element `{{semester}}`. Tots els elements entre {{}} són expressions d'AngularJS, i en aquest cas, el que s'indica és que avaluï l'element `semester`, de forma que cada element del bucle de `ng-repeat` es mostri amb el valor de l'array adequat.

Assignatures ja escollides

Aquest codi segueix un esquema similar a l'anterior. Els elements importants són la directriu `ng-repeat="topic in chosenTopics"`, l'ús de {{}} per a insertar els elements del bucle, i `ng-click="removeTopic(topic)"`. En aquest cas, la vista recorre l'array de `$scope.chosenTopics`, que es troba al model, i mostra els seus elements. A més, quan es clica el botó, `ng-click` crida la funció `removeTopic(topic)` sobre l'element concret de `chosenTopics`. Aquesta funció va unida al `$scope` del controlador, que crida `removeTopic` de `chosenTopicService`, qui fa un *broadcast* que activa tots els *listeners* del codi que actualitzen els horaris i la URL, entre altres coses.

Suma de crèdits

El codi és simple: la vista té l'expressió `{{getCredits()}}`, que fa una crida a la funció $scope.getCredits() del controlador. Es mostra el resultat en pantalla, i les directrius `ng-show` mostren en pantalla el que hi hagi dins del tag en el que estiguin, segons si compleix l'expressió o no. En aquest cas, si el resultat de la suma està fora dels límits de matrícula es mostra el contingut dels tags: un text avisant d'aquest fet.

Assignatures per escollir

El primer que fa `TopicController` a l'iniciar-se és obtenir amb `updateAvailableTopics()` les dades de les assignatures que compleixen els filtres seleccionats, i filtrar les que ja estan escollides. En aquesta secció surt una llista amb aquestes assignatures, i un camp de text per filtrar les que no coincideixin amb el que l'estudiant hi ha escrit.

El camp de text té el següent codi:

```
<input class="form-control" ng-model="topicSearch" … />
```

Que té un *listener* associat en el controlador:

```
$scope.$watch('topicSearch', function (newValue) {
            applyTopicSearchFilter(newValue); });
```

Cada vegada que `topicSearch` canviï (és a dir, cada vegada que l'input rebi text) s'executarà la funció del *listener*, que filtrarà les assignatures que coincideixin amb el que hi hagi escrit al camp de text de les assignatures per escollir. Si està seleccionada una assignatura vàlida el botó d'Afegir estarà activat, i al fer click es cridarà la funció `addTopic(topic)`. Té mecanismes similars als de `removeTopic(topic)`, i a més de la crida a `chosenTopicService` s'inicialitzen grups actius a l'assignatura, per a tenir dades que carregar a l'horari i a la URL (i que es canvien amb altres crides *broadcast*).

A sota del camp de text estan les opcions que coincideixen amb el que hi hagi escrit, i si és buit es mostren totes les assignatures que es poden escollir. Cada cop que es crida `addTopic(topic)` o `removeTopic(topic)` es fa la crida corresponent a `chosenTopicService`, i a més s'actualitza aquesta llista, eliminant del llistat l'assignatura que s'esculli o al revés.

4.3.4 Tria manual de grups

La funcionalitat de la selecció manual de grups de la nostra aplicació està a càrrec del controlador ManualClassChooserController. Consisteix en mostrar els grups de les assignatures, i afegir una funció que permeti canviar els grups actius, els que es mostren a l'horari. El seu codi és més senzill que l'apartat anterior, com es pot comprovar veient el mapa de dependències de la figura 17.

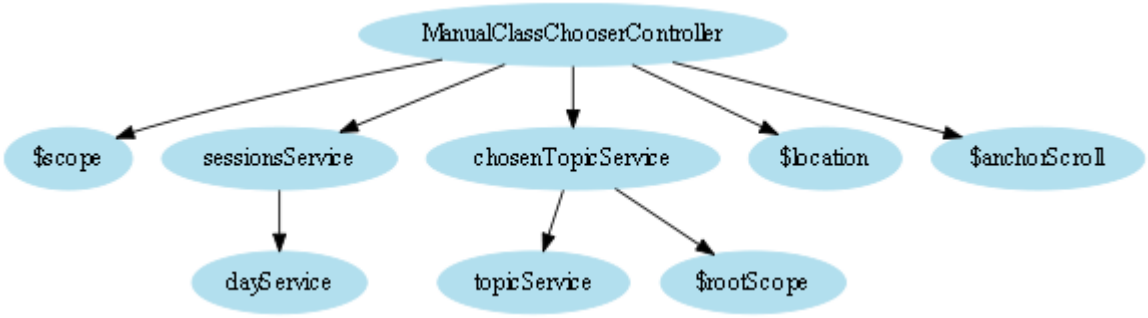

Figura 17: Mapa de dependències de ManualClassChooserController.

Tots els controladors de la nostra aplicació tenen accés a `chosenTopicsService`, ja que per a generar horaris a mida s'ha de treballar amb les eleccions de l'estudiant, i aquestes es guarden aquí. `$location` i `$anchorScroll` permeten marcar un punt d'ancoratge a un tag de la pàgina i fer scroll automàticament a aquest punt, i es fan servir per a posar l'horari en pantalla cada cop que es canvia un grup, guiant a l'estudiant. Finalment, `sessionsService` es fa servir per ordenar cada grup de les assignatures segons els dies i hores que tenen assignats, i per això depèn de `dayService`.

El model d'aquest controlador és simple: `$scope.chosenTopics` té les assignatures escollides, i `$scope.sortSessions` crida la funció d'ordenació de `sessionsService`. La funció `$scope.selectClassGroup` es crida des de la vista quan s'escull un grup.

L'aspecte de la vista es veu a la figura 18:

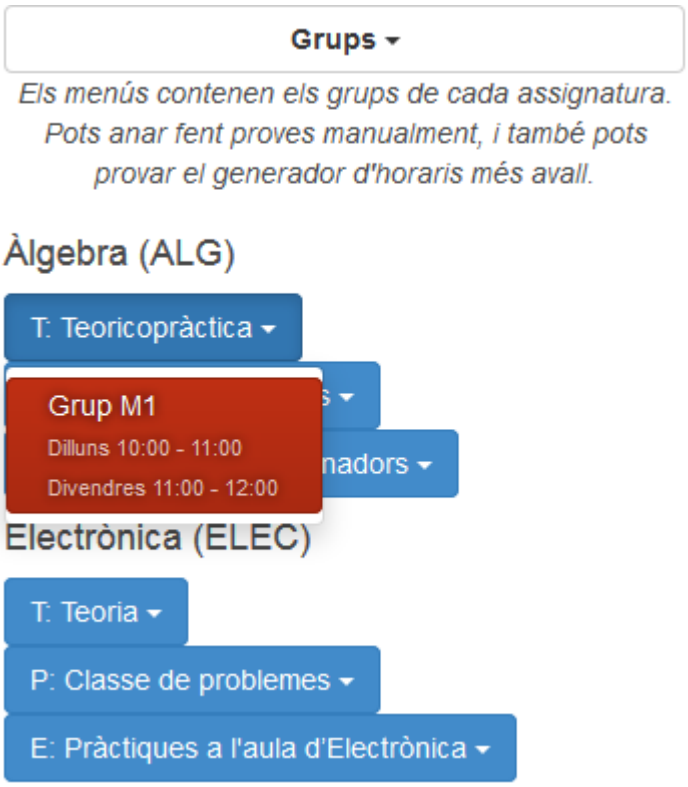

Figura 18: Aspecte de la vista sota ManualClassChooserController, amb un menú de grups obert.

El codi es simple, ja que bàsicament consisteix en recórrer l'estructura dels objectes de les assignatures, amb poques modificacions respecte la seva obtenció dels fitxers JSON. La directriu `ng-repeat` itera sobre `$scope.chosenTopics`, creant diverses instàncies del tag on es troba la directriu, i permetent l'accés al codi dins del tag a les propietats de cada element individual, que en aquest cas són assignatures. Cada assignatura té un altre array dins, amb els tipus de classes que pot tenir (teoria, problemes i pràctiques, normalment), de forma que es pot tornar a fer un `ng-repeat` sobre els seus elements, i així consecutivament fins a obtenir els grups amb les hores de classe. Cada conjunt de grups té un grup actiu, que és el que es mostra a l'horari, de forma que, quan es fa click sobre un grup, s'activa la funció `$scope.selectClassGroup`, amb l'assignatura actual, conjunt de grups actual, i grup actual com a paràmetres. Aquesta funció marca el grup que rep com a grup actiu del conjunt de grups de l'assignatura a la que pertany, a més d'enviar un *broadcast* a la resta del codi per a actualitzar l'horari i la URL.

Val la pena indicar que aquesta vista, igual que el generador i el visualitzador d'horaris, no es mostren fins que no s'hagi escollit almenys una assignatura. Això es fa posant el codi dins d'un `<div>` amb la directriu `ng-show`, condicionada al nombre de `chosenTopics`. Els menús desplegables dels grups són la funcionalitat `dropdown` de Bootstrap, i els botons que permeten amagar el contigut de cada menú fan servir `collapse`.

4.3.5 Generador d'horaris

La funcionalitat de generar horaris seguint les preferències de l'estudiant és la més complexa de l'aplicació. El seu controlador és TimetableGeneratorController, i el seu mapa de dependències és el de la figura 19:

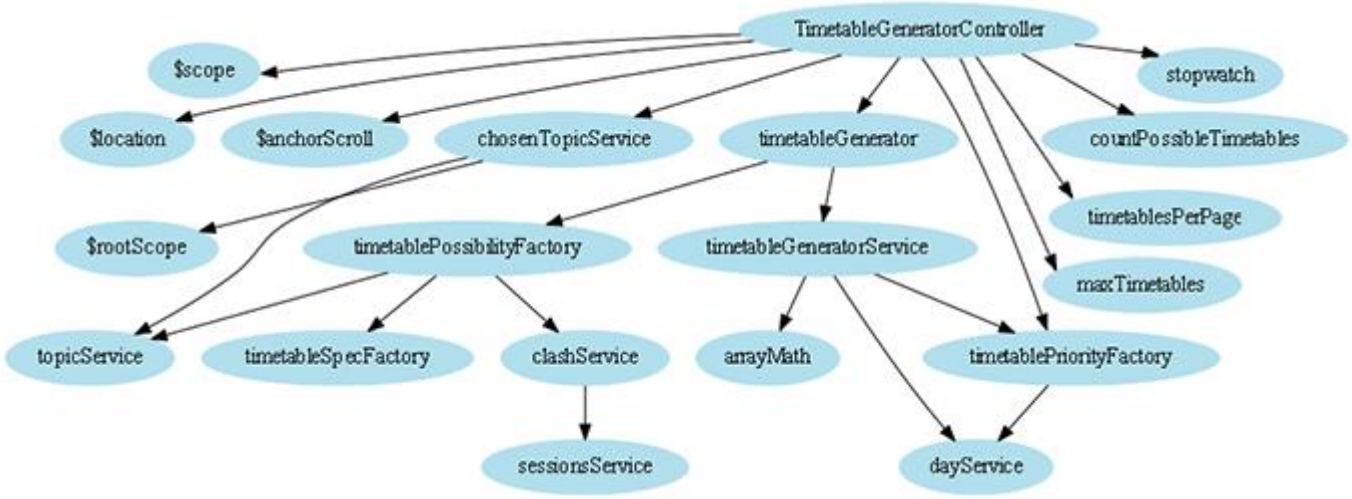

Figura 19: Mapa de dependències de TimetableGeneratorController.

A més dels ja coneguts `chosenTopicService`, `$location` i `$anchorScroll`, té moltes utilitats que fa servir: `countPossibleTimetables` calcula les combinacions possibles multiplicant la quantitat de grups de cada assignatura, `timetablesPerPage` i `maxTimetables` són constants que el controlador fa servir per calcular quantes pàgines fan falta per mostrar els resultats (el filtre `paginate` de la vista farà servir aquests resultats per a mostrar les suggerències), i `stopwatch` es fa servir per mesurar el temps que tarda en calcular els horaris. A part d'això, el gruix de les operacions es delega a `timetableGenerator` i les seves dependències.

Per tant, el controlador primerament uneix al `$scope` la llista d'assignatures escollides (`chosenTopics`), la configuració per defecte de les preferències (`config`, `numPossibleTimetables`), i les dades amb les que l'estudiant interactuarà (`clashAllowanceChoices`, `maxTimetablesChoices`, `timetablePriorities`). Després prepara les funcions `$scope.movePreference`, que la vista cridarà per permetre a l'estudiant ordenar les preferències de `timetablePriorities`, i `$scope.applyClassGroupSelection`, que canvia els grups actius actuals pels de l'horari generat i envia un *broadcast* per a que es carreguin a l'horari i la URL. Finalment, prepara `$scope.generateTimetables`, que crida `timetableGenerator` (amb `chosenTopics`, `config` i `timetablePriorities` com a paràmetres), mesura el temps que aquest ha tardat en trobar els horaris i calcula les pàgines necessàries (`numPages` i `suggestionsPerPage`) per a que la vista distribueixi els resultats.

El *provider* `timetableGenerator` té les dependències `timetablePossibilityFactory` (que depèn principalment de `clashService`) i `timetableGeneratorService` (que depèn de `timetablePriorityFactory`, qui també és dependència de `TimetableGeneratorController`). Aquest crida les funcions principals que tant la factoria com el servei tenen, i que fan el següent:

- En primer lloc, `timetablePossibilityFactory` és qui crea els horaris: rep `chosenTopics` i `config` com a paràmetres per saber quins límits de solapaments i grups té, i troba combinacions entre els diversos grups de les assignatures que compleixen aquestes limitacions, amb un algoritme de backtracking que fa servir `clashService` per anar podant candidats.
- Després, `TimetableGeneratorService` rep aquestes combinacions de grups i `timetablePriorities` (que ve del controlador), i calcula estadístiques que després s'afegeixen a cada horari. Finalment, s'ordenen els horaris segons les preferències de l'estudiant que s'han guardat a `timetablePriorities`, fent servir la funció comparadora de `timetablePriorityFactory`.

Un cop ha fet aquestes dues crides, retorna els resultats al controlador.

La vista unida a aquest controlador té l'aspecte que es veu a la figura 20.

Figura 20: Aspecte de la vista sota TimetableGeneratorController. El menú de la dreta es mostra a sota del botó de Generar horaris, quan es fa click. S'han ressaltat els elements principals.

Com es pot veure, cada part de la vista ressaltada és un element del `$scope`: el nombre d'horaris s'insereix amb `{{numPossibleTimetables}}`, les preferències ordenables es mostren amb un `ng-repeat` sobre `timetablePriorities` (i les fletxes criden a `movePreference`), i les eleccions sobre solapaments, grups sense places i comparacions d'horaris van units als valors de `config`. El botó de Generar horaris crida la funció `generateTimetables` del controlador, que posa a `true` la directriu `ng-show` del menú de la dreta, amb les dades dels horaris generats.

El menú de la dreta mostra amb `ng-repeat` les dades de cada horari generat `candidate` de l'array d'horaris `timetableCandidates`, i el botó Mostrar crida a `applyClassGroupSelection`. A més, mostra una previsualització de l'horari amb `<timetable-mini>`, una directriu personalitzada d'AngularJS, amb la llista d'assignatures `chosenTopics` i la combinació de grups de `candidate` com atributs. El mecanisme per a crear-la i el seu funcionament particular són similars a la directriu del tag `<timetable>`, que s'explica al pròxim apartat. Finalment, la llista de candidats obtinguts va unida al filtre `paginate`, que agafa una secció d'aquesta llista, determina un nombre de pàgina `pageIndex` segons els candidats que s'estiguin mostrant, i fa que només és mostrin els resultats determinats per

aquest nombre i per `suggestionsPerPage`. El valor de `pageIndex` canvia al fer click en els botons Mostrar horaris anteriors i Mostrar horaris següents.

4.3.6 Visualitzador d'horaris

La funcionalitat clau de la nostra aplicació és el visualitzador, ja que és on es mostren els resultats de tots els apartats vistos fins ara. El seu controlador es `TimetableController`, i és molt simple: només té com a dependències `$scope` i `chosenTopicService`. El seu codi consisteix en tenir la llista de `chosenTopics` al `$scope`, i un *listener*, que s'activa cada vegada que `TopicController`, `ManualClassChooserController`, `TimetableGeneratorController`, o `TimetableUrlController` envien un *broadcast*, i respon actualitzant les dades. En aquest cas, el controlador s'uneix amb `ng-controller` a un tag personalitzat que és qui fa el gruix del treball: `<timetable>`, amb `chosenTopics` i `classSelections` com a atributs.

Recordem que AngularJS té les seves pròpies directrius, però també ofereix la possibilitat de crear tags i atributs personalitzats. Per això es fa servir la declaració `directive`, que normalment es lliga a un *template* de codi HTML. Quan s'introdueix la directriu al codi HTML, la compilació del *bootstrapping* d'AngularJS la prepara per a que el navegador la pugui llegir. Algunes característiques són:

- Es poden afegir atributs, i convertir els valors que tinguin al `$scope` intern de la directriu, aïllat i diferenciat del que té el pare (normalment, el del controlador).
- Es pot restringir per a que es pugui fer servir com un tag HTML, com un atribut, com una classe, o com més d'una cosa.
- Es pot afegir una funció `link`, que permet introduir codi que pugui treballar amb el `$scope`, i fer crides a funcions de les dependències de la directriu.

La nostra aplicació fa servir 7 directrius personalitzades:

- El tag `<timetable>`, per a la taula amb l'horari del visualitzador d'horaris. Fa servir el *template* de *views/timetable.tpl.html*, i també té codi a `link`.
- El tag `<timetable-mini>`, per a cada horari del generador d'horaris. Fa servir el *template* de *views/timetable-mini.tpl.html*, i també té codi a `link`.
- El tag `<clash-group>`, que es fa servir dins de `<timetable>` i `<timetable-mini>`, i que serveix per a organitzar cada casella dia-hora de la taula d'horaris, ja que pot tenir més d'una assignatura si hi ha un solapament. Fa servir el *template* de *views/clashGroup.tpl.html*, però no té codi a `link`.
- El tag `<booking>`, que es fa servir dins de `<clash-group>`, i mostra la informació d'una hora concreta d'un grup d'una assignatura. Fa servir el *template* de *views/booking.tpl.html*, però no té codi definit a `link`.

- L'atribut `booking-top-offset`, que es fa servir com a atribut de `<booking>`, i que calcula el valor *offset* necessari per a posicionar amb CSS el `booking` a la taula horària de `<timetable>` i `<timetable-mini>`, des de la posició de dalt de la taula. No té associat un *template*, però sí que té codi a `link`.
- L'atribut `topic-highlight`, que es fa servir com a atribut de `<booking>`, que fa ús del *provider* `colorAllocator` per a assignar un color de fons a cada element `booking`. Aquest color és constant per a tots els `booking` de cada assignatura, i així diferencia visualment quines classes té cadascuna. No té associat un *template*, però sí que té codi a `link`.
- L'atribut `booking-locked-indicator`, que es fa servir com a atribut de `<booking>`, i que afegeix un fons a ratlles amb CSS per a diferenciar les classes en les que no hi ha cap altre grup alternatiu (per exemple, les classes de teoria). No té associat un *template*, però sí que té codi a `link`.

Per tant, per a mostrar l'horari que formen les assignatures escollides, primer s'insereix la directriu `<timetable>` com un tag HTML més, amb `TimetableController` lligat amb `ng-controller`. Aquesta directriu marca com es carreguen les dades a l'horari dins de `link`, i té un `$scope` que llegeix el codi HTML del *template* que hi està associat. En certa manera, fa el paper que tenia el controlador fins ara, respecte el codi del *template*. El seu mapa de dependències és el de la figura 21:

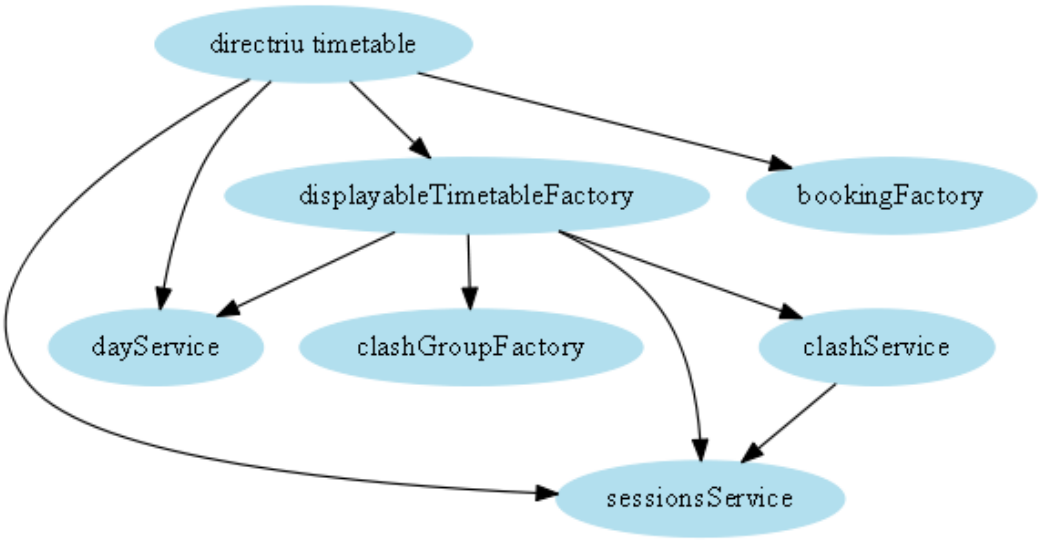

Figura 21: Mapa de dependències de la directriu timetable.

Com ja s'ha dit, la directriu `<timetable>` s'invoca amb uns atributs al codi HTML, que s'uneixen al seu `scope` intern amb aquest codi dins de la directriu:

```
scope: {
   topics: '=',
   classSelections: '='
},
```

I el codi següent com a atributs:

```
<timetable topics="chosenTopics" class-selections="" … ></timetable>
```

D'aquesta manera, s'uneix `chosenTopics` a la variable `topics` del `$scope` intern, i s'inicialitza `classSelections` en blanc, que s'anirà actualitzant conforme s'afegeixin grups d'assignatures. El símbol = activa el *data binding* bidireccional entre l'atribut HTML (que obté el seu valor del controlador) i la variable interna.

Seguint amb el codi de la directriu, dins de `link` s'especifica com es carreguen les dades de l'horari. Al seu `$scope` s'afegeixen `days` i `hours`, que la vista (el *template* HTML) farà servir per a crear la taula horària, i la funció `updateTimetable`, que fa el següent:

1. Crida la funció `createBookingsForTopics` de `bookingFactory`, que rep `topics` i `classSelections` com a paràmetres, i crea un array amb els objectes `booking` que contenen tota la informació que es mostra a l'horari: assignatura, grup, hora d'inici, hora de fi, aula, etcètera.
2. Aquest resultat el passa com a paràmetre a la funció `createTimetableForBookings` de `displayableTimetableFactory`, qui crea i retorna l'estructura de dades de la taula horària: un array de 5 arrays, un per cada dia, amb estructures `clashGroup`, que contenen un o més `bookings`. Per tant, cada `clashGroup` es una agrupació de classes que hi ha a un dia, col·locades en columnes que poden sol·lapar-se en una mateixa hora (cada fila).

Aquesta estructura es guarda a `$scope.timetable`, per a que la vista pugui omplir la taula horària. Finalment, a la directriu també s'afegeixen dos *listeners*, un per a `topics` i un per a `classSelections`, de manera que activin la funció `updateTimetable` quan canviïn.

Ara que ja sabem com funciona el tag `<timetable>` podem entendre el tag `<timetable-mini>` que fa servir la vista sota `TimetableGeneratorController`. La diferència és que el codi a `link` de la seva directriu, en comptes de crear arrays amb dies i hores, prepara un estil per a mostrar les dades en menor tamany (ja que hi haurà molts tags per pàgina), que s'uneix al seu *template* amb `ng-style` (que permet associar un estil de CSS a un tag de forma condicional). La resta del seu funcionament és similar, i fa ús de `updateTimetable` de la mateixa forma.

Veiem ara la vista resultant de les directrius `timetable`, `clashGroup` i `booking` i els seus *templates* a la figura 22.

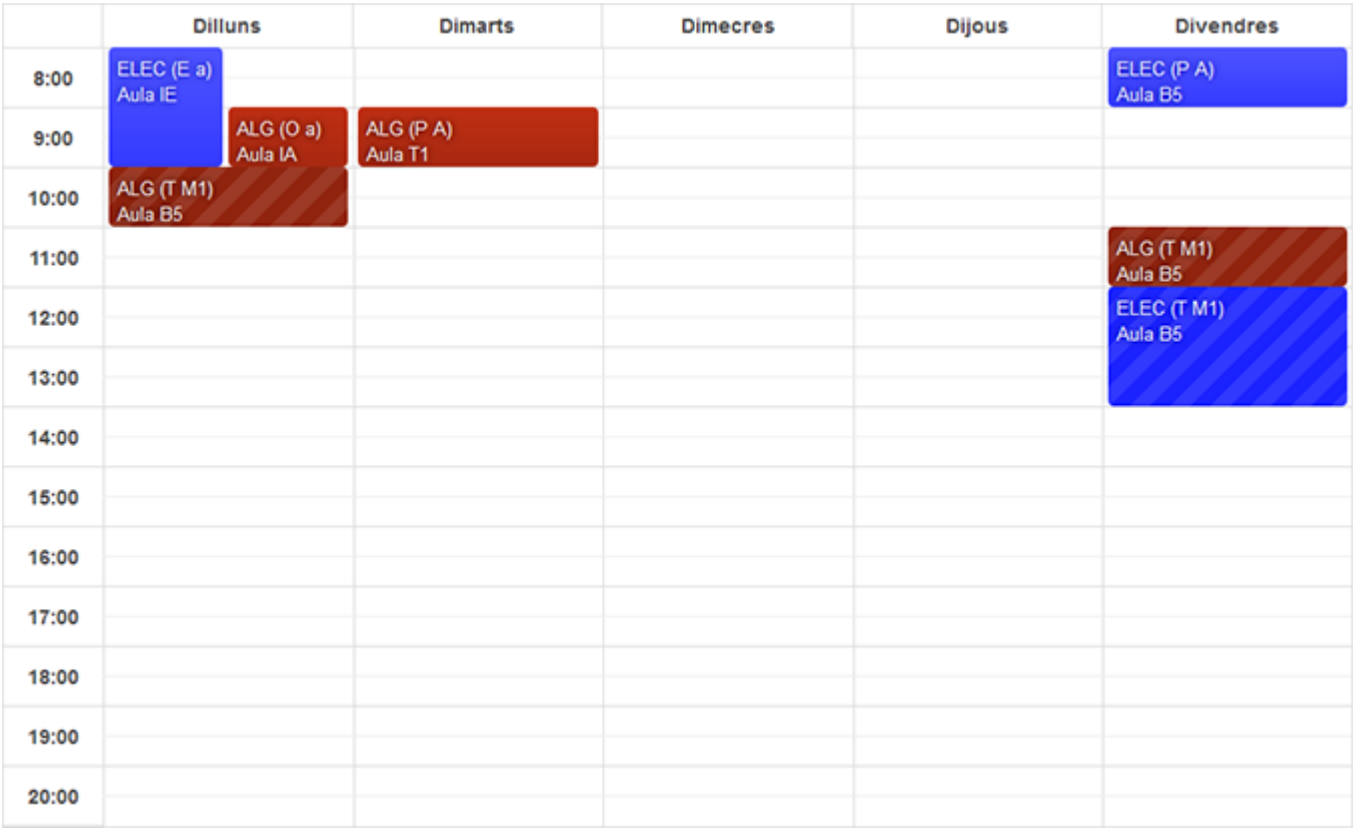

Figura 22: Aspecte de la vista sota TimetableController.

El *template* associat a `<timetable>` és bastant simple: iterant amb `ng-repeat` crea la fila dels dies i la columna de les hores, i totes les cel·les de la taula, incialment buides. Per cada dia s'insereix amb `ng-repeat` un tag `<clashgroup>`, i li passa cada agrupació de classes corresponent a aquell dia com a valor dels seus atributs `clash-group` i `start-offset`, que s'uneixen al `$scope` intern de la seva directiva.

Similarment, el *template* associat a cada `<clashgroup>` crea una taula on per cada cel·la s'insereix un `<booking>` amb `ng-repeat`. Cada instància de `<clashgroup>` itera sobre els elements de l'estructura que té al seu `$scope` intern, i passa aquesta informació als atributs `booking` i `start-offset` de `booking`.

Finalment, el *template* associat a `<booking>` mostra la informació de cada classe que es veu a la taula horària (assignatura, tipus de classe, grup, i aula) i es prepara un enllaç a la pàgina de l'aula i el *tooltip* amb informació addicional. També rep com a atributs les directives `booking-top-offset`, `topic-highlight` i `booking-locked-indicator`, que apliquen el codi de les directrius corresponents per a que cada classe es posicioni correctament i es pinti de forma que siguin fàcilment distinguibles. Això es fa per cada `booking` de cada `clashgroup` de cada dia de `timetable`.

4.3.7 Format de les dades

Els fitxers de dades que contenen la informació de les assignatures mereixen una menció especial, ja que cal conèixer la seva estructura per a que l'aplicació pugui fer-los servir correctament. Així doncs, tenim dues classes de fitxers: el primer tipus conté la llista d'assignatures de cada semestre amb informació basica, i l'altre tipus són fitxers per a cada assignatura amb informació detallada.

Començem pel primer tipus: són 2 fitxers amb els noms *assignaturesTardor.json* i *assignaturesPrimavera.json*. Cadascun d'ells és un array d'objectes assignatura. A continuació es mostra part del contingut del segon fitxer:


```
[
  {
    "id": "P364301",
    "name": "Estructura de dades",
    "code": "ED",
    "year": "Primer",
    "semester": "Primavera",
    "type": "Obligatòria",
    "ects": 6,
    "location": "Facultat de Matemàtiques",
    "link":
"http://www.ub.edu/grad/plae/AccesInformePD?curs=2014&codiGiga=364301&idioma=CAT&recurs=publicacio",
    "subscript": null
  },
  {
    "id": "P361606",
    "name": "Biologia Molecular i Cel·lular dels Microorganismes",
    "code": "BMCM",
    "year": "Segon",
    "semester": "Primavera",
    "type": "Optativa",
    "ects": 6,
    "location": "Facultat de Biologia",
    "link":
"http://www.ub.edu/grad/plae/AccesInformePD?curs=2014&codiGiga=361606&idioma=CAT&recurs=publicacio",
    "subscript": "Menció en Bioinformàtica"
  }
]
```


El fitxer mostrat té dues assignatures, però pot tenir un nombre variable. S'han ressaltat en colors diferents per a diferenciar-les. Cada camp vol dir el següent:

1. **id (string):** Identificador *únic* de l'assignatura. Ha de tenir com a mínim un nombre (el codi de l'assignatura) i com a màxim una lletra al davant. La convenció és T per les assignatures de tardor, P per les de primavera.
2. **name (string):** Nom de l'assignatura. Ha de ser el que surt a la pàgina dels horaris, i ha de ser *únic*.
3. **code (string):** Segles o abreviació de l'assignatura. Ha de ser molt curt (2-3 caràcters, 4 com a màxim) per a que es mostri bé a l'horari, i ha de ser *únic*.
4. **year (string):** Curs de l'assignatura. Els seus valors poden ser Primer, Segon, Tercer i Quart. Altres valors no seran reconeguts per l'aplicació.
5. **semester (string):** Semestre de l'assignatura. Els seus valors poden ser Tardor i Primavera. Altres valors no seran reconeguts per l'aplicació.
6. **type (string):** Obligatorietat de l'assignatura. Els seus valors poden ser Formació Bàsica, Obligatòria i Optativa. Altres valors poden ser reconeguts per l'aplicació, però Optativa s'ha de dir exactament així.
7. **ects (nombre, pot tenir decimals):** Nombre de crèdits de l'assignatura.
8. **location (string):** Facultat on s'imparteix l'assignatura.
9. **link (string):** Enllaç al pla docent de l'assignatura, de l'any actual.
10. **subscript (string):** Menció a la que pertany l'assignatura, si és que ve d'una carrera externa al grau d'Enginyeria Informàtica. En cas contrari, el camp és null.

El segon tipus de fitxer conté les dades detallades d'una assignatura, amb la informació dels grups per a cada tipus de classe (teoria, pràctiques i problemes, si n'hi ha) i les seves hores de classe. Hi ha d'haver un fitxer per a cada assignatura, i el nom del fitxer és necessàriament el que surti al camp **id** del tipus de fitxer comentat abans, seguit de *.json*. A continuació es mostra el contingut de *T364309.json*, que correspon a l'assignatura de Xarxes, i s'explicarà què significa cada cosa.


```
{
 "id": "T364309",
 "name": "Xarxes",
 "code": "XAR",
 "year": "Tercer",
 "semester": "Tardor",
 "type": "Obligatòria",
 "ects": 6,
 "location": "Facultat de Matemàtiques",
 "link_horari":
"http://www.ub.edu/grad/infes/fitxaInfe.jsp?n0=2&n1=0&n2=1&curs=2014&ens=TG1077&assig=364309",
 "link_pla_docent":
"http://www.ub.edu/grad/plae/AccesInformePD?curs=2014&codiGiga=364309&idioma=CAT&recurs=publicacio",
 "subscript": null,
 "classes":
```

```
[
  {
    "id": 35831,
    "name": "Teoria",
    "symbol": "T",
    "note": "",
    "class_groups":
    [
      {
        "name": "T1",
        "id": 48340,
        "group_id": 1,
        "note": "",
        "places": 10,
        "activities":
        [
          {
            "day_of_week": "Divendres",
            "time_starts_at": "15:00",
            "time_ends_at": "17:00",
            "seconds_starts_at": 54000,
            "seconds_ends_at": 61200,
            "seconds_duration": 7200,
            "room": "Aula B5",
            "room_link": "http://www.ub.edu/grad/infes/fitxaLoc.jsp?idEspai=11698",
            "language": "Català",
            "teachers":
            [
              {
                "name": "LOPEZ DE MIGUEL, MANUEL",
                "link":
"http://www.ub.edu/grad/infes/fitxaInfe.jsp?n0=P&n1=0&n2=1&curs=2014&prof=3406"
              }
            ]
          }
        ]
      }
    ]
  },
  {
    "id": 35832,
    "name": "Pràctiques amb ordinadors",
    "symbol": "O",
    "note": "",
    "class_groups":
    [
      {
        "name": "a",
        "id": 48341,
        "group_id": 1,
        "note": "",
        "places": 6,
        "activities":
```

```
          [
              {
                  "day_of_week": "Dijous",
                  "time_starts_at": "19:00",
                  "time_ends_at": "21:00",
                  "seconds_starts_at": 68400,
                  "seconds_ends_at": 75600,
                  "seconds_duration": 7200,
                  "room": "Aula IF",
                  "room_link": "http://www.ub.edu/grad/infes/fitxaLoc.jsp?idEspai=11696",
                  "language": "Català",
                  "teachers":
                  [
                      {
                          "name": "LOPEZ DE MIGUEL, MANUEL",
                          "link":
"http://www.ub.edu/grad/infes/fitxaInfe.jsp?n0=P&n1=0&n2=1&curs=2014&prof=3406"
                      }
                  ]
              }
          ]
        },
        {
          "name": "b",
          "id": 48342,
          "group_id": 2,
          "note": "",
          "places": 0,
          "activities":
          [
              {
                  "day_of_week": "Divendres",
                  "time_starts_at": "17:00",
                  "time_ends_at": "19:00",
                  "seconds_starts_at": 61200,
                  "seconds_ends_at": 68400,
                  "seconds_duration": 7200,
                  "room": "Aula IF",
                  "room_link": "http://www.ub.edu/grad/infes/fitxaLoc.jsp?idEspai=11696",
                  "language": "Català",
                  "teachers":
                  [
                      {
                          "name": "PALACIO BONET, FRANCISCO",
                          "link":
"http://www.ub.edu/grad/infes/fitxaInfe.jsp?n0=P&n1=0&n2=1&curs=2014&prof=20057"
                      }
                  ]
              }
          ]
        },
        {
          "name": "c",
```

```
            "id": 48343,
            "group_id": 3,
            "note": "",
            "places": 9,
            "activities":
            [
                {
                    "day_of_week": "Divendres",
                    "time_starts_at": "17:00",
                    "time_ends_at": "19:00",
                    "seconds_starts_at": 61200,
                    "seconds_ends_at": 68400,
                    "seconds_duration": 7200,
                    "room": "Aula IA",
                    "room_link": "http://www.ub.edu/grad/infes/fitxaLoc.jsp?idEspai=11703",
                    "language": "Català",
                    "teachers":
                    [
                        {
                            "name": "BERNAT UBIAGA, IVAN",
                            "link":
"http://www.ub.edu/grad/infes/fitxaInfe.jsp?n0=P&n1=0&n2=1&curs=2014&prof=8192"
                        }
                    ]
                }
            ]
        }
    ]
  }
 ]
}
```


El fitxer mostrat és un únic objecte amb diversos camps, molts dels quals son arrays, amb objectes aniuats fins a 5 nivells. Aquest fitxer en concret té una assignatura amb 2 tipus de classe, que tenen 1 grup de teoria (amb una classe, de 2 hores) i 3 grups de pràctiques (amb una classe cadascuna, de 2 hores). Cada nivell està ressaltat en un color diferent. A sota s'expliquen les característiques de cada camp a cada nivell:

Informació de l'assignatura (ressaltada en blau)

1. **id (string):** Identificador *únic* de l'assignatura. Ha de ser el mateix que el que surt al fitxer general amb la llista d'assignatures, i que el nom del fitxer actual.
2. **name (string):** Nom de l'assignatura. Igual que al fitxer general.
3. **code (string):** Segles o abreviació de l'assignatura. Igual que al fitxer general.
4. **year (string):** Curs de l'assignatura. Igual que al fitxer general.
5. **semester (string):** Semestre de l'assignatura. Igual que al fitxer general.
6. **type (string):** Obligatorietat de l'assignatura. Igual que al fitxer general.

7. **ects (nombre, pot tenir decimals):** Nombre de crèdits de l'assignatura. Igual que al fitxer general.
8. **location (string):** Facultat on s'imparteix l'assignatura. Igual que al fitxer general.
9. **link_horari (string):** Enllaç a l'horari de l'assignatura al GRAD, de l'any actual.
10. **link_pla_docent (string):** Enllaç al pla docent de l'assignatura, de l'any actual. Igual que al fitxer general.
11. **subscript (string):** Menció a la que pertany l'assignatura. Igual que al fitxer general.
12. **classes (array):** Conté la informació dels tipus de classe, que a la vegada contenen els grups, les hores de classe dels grups i els professors. L'array normalment té un tamany de 2 o 3, depenent de si l'assignatura té grups de problemes o no, però el nombre és variable i pot ser major.

Informació dels tipus de classe amb els seus grups (ressaltada en verd)

1. **id (int):** Identificador del conjunt de grups que té el tipus de classe actual. Ha de ser un nombre *únic*, o sinó el generador d'horaris no funcionarà bé.
2. **name (string):** Nom del tipus de classe. Generalment serà Teoria, Problemes, Pràctiques de laboratori, o variants sobre aquests noms, però no hi ha limitació.
3. **symbol (string):** Segles o abreviació del tipus de classe. Ha de ser molt curt (1-2 caràcters) per a que es mostri bé a l'horari.
4. **note (string):** Anotacions rellevants sobre el tipus de classe i els grups que té.
5. **class_groups (array):** Conté els grups, que a la vegada contenen les hores de classe dels grups i els professors. L'array no sol tenir un tamany major a 5 o 6, ja que no sol haver molts grups, però el nombre és variable i pot ser major.

Informació dels grups (ressaltada en groc)

1. **name (string):** Nom del grup. Ha de ser el que surt a la pàgina dels horaris, i ha de ser molt curt (2-3 caràcters més o menys) per a que es mostri bé a l'horari.
2. **id (int):** Identificador del grup actual. Ha de ser un nombre *únic* entre totes les assignatures, o sinó el generador d'horaris no funcionarà bé.
3. **group_id (int):** Identificador local del grup actual, dins del conjunt de grups d'un tipus de classe.
4. **note (string):** Anotacions rellevants sobre el grup (freqüència de classe, dies de classe exclosos, etc).
5. **places (int):** Nombre de places lliures que queden al grup. Pensat per a ser actualitzat en temps real amb la matrícula. Si són ilimitades, el valor ha

de ser un nombre raonablement alt. En aquest fitxer, els nombres estan posats a l'atzar.

6. **activities (array):** Conté les hores de classe dels grups (a més d'informació com l'aula o l'idioma), i els professors. L'array sol tenir un tamany de 1, 2 o 3, i cada element són hores de classe agrupades: dues hores de classe seguides són 1 element, però si van separades són 2. No obstant, el nombre és variable.

Informació de les hores de classe (ressaltada en taronja)

1. **day_of_week (string):** Dia de la setmana. Pot ser Dilluns, Dimarts, Dimecres, Dijous o Divendres. Altres valors no són reconeguts per l'aplicació.
2. **time_starts_at (string):** Hora en què comença la classe, en format 24h. Pot ser de 8:00 a 21:00, ambdós inclosos. Accepta minuts concrets.
3. **time_ends_at (string):** Hora en què acaba la classe, en format 24h. Pot ser de 8:00 a 21:00, ambdós inclosos. Accepta minuts concrets.
4. **seconds_starts_at (int):** Hora en què comença la classe, en segons. Els valors poden ser de 28800 (les 8:00) a 75600 (les 21:00), ambdós inclosos.
5. **seconds_ends_at (int):** Hora en què acaba la classe, en segons. Els valors poden ser de 28800 (les 8:00) a 75600 (les 21:00), ambdós inclosos.
6. **seconds_duration (int):** Duració de la classe, en segons. Els valors usuals són 3600 (1 hora), 7200 (2 hores) i 10800 (3 hores), però el nombre és variable.
7. **room (string):** Aula on s'imparteix la classe.
8. **room_link (string):** Enllaç a la pàgina de l'aula on s'imparteix la classe.
9. **language (string):** Idioma en què s'imparteix la classe.
10. **teachers (array):** Conté els professors que imparteixen la classe. Normalment és 1, però el nombre és variable i pot ser major.

Informació dels professors (ressaltada en gris)

1. **name (string):** Nom del professor/a.
2. **link (string):** Enllaç a la pàgina amb la informació del professor.

5. PROVES I RESULTATS

L'estructura del projecte està construïda per a facilitar fer proves amb l'aplicació: Grunt automàticament revisa i compila els fitxers tot just es canvien, tot mostrant els errors que trobi, i Karma permet fer proves unitàries d'algunes parts del codi. Per exemple, el fitxer *timetable.spec.js* conté una varietat d'escenaris que posa a prova la funcionalitat encarregada de trobar solapaments entre assignatures. En el cas d'aquest projecte, ha servit per saber quan les modificacions realitzades al codi causaven que alguna cosa deixés de funcionar, i com que l'execució dels tests és part de la rutina de Grunt, que s'executa automàticament, qualsevol fallada es veia d'immediat.

Apart d'això, l'aplicació s'ha provat en *localhost*, primàriament sota Windows 7, als navegadors Mozilla Firefox 38.0.5 i Google Chrome 43.0.2357.130. També s'han fet proves als navegadors Internet Explorer 11, Opera 30, i Safari 8.0.6 sota Mac. Funciona correctament en tots, tot i que en Internet Explorer s'ha d'activar la configuració d'intranet per a que pugui carregar assignatures de la URL, ja que les proves s'han fet en *localhost*.

Així doncs, es pot dir que els resultats del projecte han estat satisfactoris. S'ha assolit l'objectiu general d'una aplicació capaç de facilitar la vida a l'estudiant, ja que està completament adaptada a les assignatures del Grau en Enginyeria Informàtica de la UB, i les capacitats de personalització de l'horari són molt potents. A més, l'ús d'un projecte base com el que s'ha triat m'ha permès aprendre i treballar amb una gran quantitat de tecnologies web d'última generació, de forma que aquest objectiu també ha estat assolit.

Respecte els objectius específics, el fet d'escollir una aplicació base tan completa assegurava que s'acomplirien gairebé tots al final del projecte. No obstant, l'anàlisi en profunditat del codi ha permès la seva integració amb les dades de la UB, i també la seva millora: s'ha obtingut un disseny visual més fàcil i còmode d'utilitzar, i al final del projecte fins i tot s'ha pogut corregir algun *bug* que venia inherent en la base.

D'altra banda, no s'han pogut aconseguir els objectius d'integrar l'aplicació amb les restriccions de matrícula de cada estudiant, o d'obtenir accés a la base de dades de les assignatures a temps, degut a factors externs al projecte. Probablement no era una expectativa realista obtenir en la duració del projecte l'accés a dades confidencials de la universitat, però el fet que part del personal responsable de la seva gestió mostrés interès per aquesta aplicació resulta encoratjador.

6 CONCLUSIONS I TREBALL FUTUR

6.1 Conclusions

Amb l'assoliment de la majoria d'objectius, puc dir que estic bastant satisfeta amb el treball realitzat. S'ha de tenir en compte que aquest projecte s'ha fet amb menys temps de l'habitual, i que, si bé l'aplicació base que s'ha fet servir per a treballar és molt completa, la documentació que tenia era pràcticament nul·la, de forma que he hagut de buscar-me la vida per a entendre les coses més bàsiques que feia el codi. Per tant, si bé m'hauria agradat iterar més sobre l'aplicació, i anar afegint millores al seu funcionament, trobo que el resultat és prou bo. Tot i que sempre es pot ampliar més.

També m'alegra el fet que part del personal administratiu de la universitat s'hagi mostrat interessat pel tipus d'aplicació que he fet. Donat que jo no he seguit molt el ritme típic de matriculació d'assignatures del grau, i que he fet optatives d'una altra carrera, quadrar horaris era una tasca que portava la seva feina, de manera que estaria molt bé si la UB acabés adoptant una eina així.

Així doncs, estic contenta amb tot el que m'ha aportat fer el TFG. He posat en pràctica els coneixements adquirits durant la carrera, he après molt durant la seva realització, he conegut tecnologies web molt interessants amb les que m'agradaria treballar més, i en general, ha estat una experiència enriquidora.

6.2 Possibles ampliacions i treballs futurs

Com diu l'apartat de conclusions, hauria estat bé millorar i ampliar l'aplicació final. A continuació s'enumeren possibles projectes que poden sortir d'aquestes ampliacions.

El treball futur més important és connectar l'aplicació amb les assignatures de la UB. Ja s'ha fet una reunió amb els responsables de la base de dades GR@D i estan d'acord en ajudar, de forma que hauria de ser possible obtenir l'accés necessari per fer un programa que generi els fitxers JSON directament a partir de les dades de la UB.

El següent treball que veig interessant és integrar les dades de l'estudiant amb l'aplicació, per a que apliqui automàticament la seva situació acadèmica. El projecte actual ja restringeix el nombre d'assignatures segons els límits de crèdits que es poden matricular, però si es filtressin les assignatures que l'estudiant ja ha superat, s'indiquessin les que queden pendents i s'han de matricular necessàriament, i s'ajustessin les assignatures optatives a la seva intensificació o menció, entre altres coses, es facilitaria molt més la planificació de la matrícula.

Una cosa més que seria convenient és una versió de l'aplicació que sigui *responsive*, i s'adapti a les mides dels dispositius on s'executi. Si bé actualment la matrícula de la UB no està especialment pensada amb mòbils i tablets en ment, en el futur ho estarà, i caldrà habilitat aquesta aplicació per a l'ocasió. Això pot consistir en adaptar el disseny de la web i que es recol·loquin els elements segons el *viewport*, o en fer una aplicació nativa per a Android o iOS. Si es fa una versió nativa tindria l'avantatge de poder consultar els horaris offline, ja que, un cop s'han descarregat els fitxers amb les dades, realment no és necessària una connexió amb internet.

Un altre projecte, més a llarg termini, tindria en compte que algunes assignatures optatives no tenen els grups de matrícula pensats per a encaixar en horaris setmanals. En la menció de Bioinformàtica n'hi ha algunes que els seus grups, en comptes de tenir unes hores fixes cada setmana, tenen una setmana al semestre on ocupen bona part del matí o la tarda durant uns quants dies. Per a acomodar aquestes assignatures, aniria bé que a més de calcular l'horari setmanal es pogués calcular també un calendari semestral. Totes les assignatures es beneficiarien d'una funcionalitat així: a més dels grups amb docència limitada es podria afegir el calendari d'exàmens, i seria fantàstic si també inclogués les dates d'entrega de pràctiques. L'aplicació base d'aquest projecte tenia una funcionalitat d'exportar a Google Calendar que es va eliminar per no funcionar molt bé, però potser es podria rescatar i adaptar-la a aquesta proposta.

Respecte a millores de l'aplicació actual, sempre es possible afegir més opcions. Es podrien afegir restriccions més dures a la generació d'horaris, com per exemple la prohibició d'hores concretes, pels compromisos horaris fora de la universitat. Tal i com funciona ara, es pot configurar per a que minimitzi el temps de classe un dia concret, però una funcionalitat així pot fer que directament es filtrin els grups que se solapin amb aquestes hores vetades (i si no hi ha grups alternatius als que s'han filtrat, no és possible fer un horari amb els requisits demanats). També es podria afegir una funció que compari si dues assignatures seguides a la taula s'imparteixen a facultats diferents, i que quan això passi ho indiqui, ja que vol dir que es perdrà temps de classe pel desplaçament. Aquestes són algunes millores que m'hauria agradat incorporar, però que per falta de temps no ha estat possible.

7 ANNEX

7.1 Contingut de l'entrega

L'estructura de fitxers d'aquest projecte està basada en la del projecte de Github **ng-boilerplate**, que té per objectiu proveir una estructura inicial per a projectes d'AngularJS, i a més incorpora la configuració de Bower, Grunt i Karma que s'ha explicat abans. Aquest projecte està molt ben documentat, i la majoria del que diu es pot aplicar aquí, però tot i així s'explicarà breument com està organitzada l'entrega.

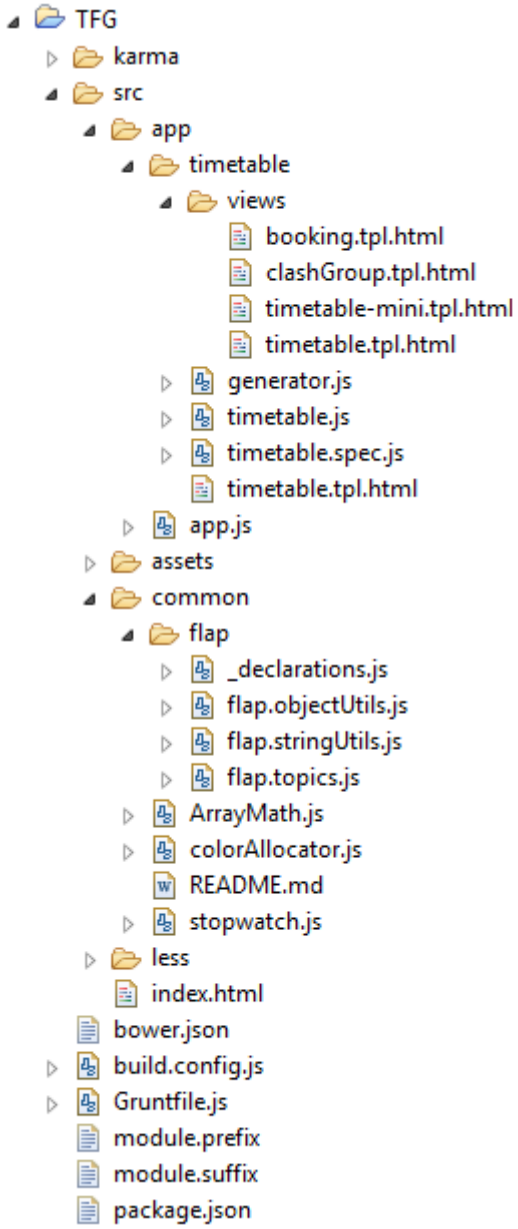

Figura 23: Organització dels fitxers. S'ha expandit la vista fins a src/app/timetable/views i common/flap.

La carpeta *src* és on està el codi font, la resta son les instal·lacions de Grunt, Bower i Karma, així com els seus fitxers de configuració. Dins de *src*, la carpeta *app* conté els arxius principals de l'aplicació: el *bootstrapping* d'AngularJS carrega *app.js*, qui insereix *app/timetable/timetable.tpl.html* a la vista principal.

Dins de *app/timetable*, *timetable.js* conté un mòdul amb la gran majoria de codi de l'aplicació: tots els controladors, les funcionalitats d'elecció d'assignatures, d'elecció manual de grups, el visualitzador d'horaris, i moltes de les seves funcions auxiliars. *generator.js* conté el mòdul amb les funcions i operacions que la funcionalitat de generació d'horaris fa servir per a calcular els diferents horaris i ordenar-los segons les preferències de l'usuari. *timetable.spec.js* és un mòdul de proves, que primàriament testeja el funcionament de la detecció de solapaments entre horaris. Finalment, la carpeta *views* conté els *templates* HTML que fan servir les directrius personalitzades d'AngularJS, i es corresponen als tags amb el mateix nom.

Les altres carpetes dins de *src* són les següents: *assets* conté les imatges i les fonts necessàries per a representar la pàgina, i també conté els fitxers JSON amb les assignatures. *common* té moltes utilitats genèriques no específiques al projecte, excepte *common/flap/flap.topics.js*, que conté classes de lectura i manipulació dels fitxers JSON de les assignatures. *less* conté els fitxers amb CSS i LESS.

7.2 Instal·lació i execució de l'aplicació

Cal instal·lar node.js (https://nodejs.org/) per a fer servir el seu gestor de paquets NPM. També cal instal·lar Git (https://git-scm.com/downloads), per a que Bower pugui descarregar els paquets necessaris. Un cop descarregats i instal·lats (Git s'ha d'instal·lar amb l'opció d'afegir-se al PATH de la línia de comandes), s'ha d'executar la següent comanda:

(Windows): `npm -g install grunt-cli karma bower`

(Linux): `sudo npm -g install grunt-cli karma bower`

Un cop feta la instal·lació global de Grunt, Karma i Bower, es canvia al directori de la carpeta on es trobi l'estructura de fitxers de l'aplicació. En el directori del codi s'han d'executar les comandes `npm install` i `bower install`, per instal·lar localment els components necessaris.

Un cop s'ha instal·lat tot, per a properes execucions de l'aplicació només caldrà canviar al directori on es trobi el codi i executar la comanda **`grunt watch`** per a que posi l'aplicació a punt. Es pot trobar a http://localhost:1337/.

Val la pena dir que una aplicació d'AngularJS, al ser purament client-side, es pot obrir directament des del navegador. No obstant, per motius de seguretat, els navegadors moderns limiten severament les accions que els arxius de JavaScript locals poden fer. Per tant, és recomanable que se serveixin des d'un servidor, com fa Grunt.

7.3 Glossari de termes i conceptes

La majoria de termes tècnics que es fan servir per a explicar el funcionament del projecte es descriuen dins de l'explicació. No obstant, hi ha alguns termes sobre programació web general que cal conèixer.

Client i client-side

A la programació web, el client és la part de l'aplicació que s'executa a l'ordinador local de l'usuari, i es connecta a un servidor per tal d'obtenir les dades a mostrar. Client-side és el costat del client, per a referir-se al que s'executa localment.

Servidor i server-side

A la programació web, el servidor és la part remota de l'aplicació a la qual es connecta el client, i que li retorna les peticions que demana. Server-side és el costat del servidor, per a referir-se al que s'executa remotament.

Front-end

A la programació web, el front-end és la part de l'aplicació encarregada de la interacció amb l'usuari. En aplicacions client-servidor sol estar associada amb el client, però no sempre ha de ser així.

Back-end

A la programació web, el back-end és la part de l'aplicació encarregada de les operacions i la lògica que no veu l'usuari. En aplicacions client-servidor sol estar associada amb el servidor, però no sempre ha de ser així.

Framework

Un framework és una estructura base que defineix el comportament de la utilització dels fitxers i les funcionalitats que conté. S'utilitza per construir projectes sobre una bona base, obligant a mantenir una estructura de fitxers correcte i ordenada, mantenint en tot moment una alta seguretat i proporcionant eines de desenvolupament ja creades.

8 BIBLIOGRAFIA

Aquí es poden trobar els enllaços consultats durant el desenvolupament del projecte.

Bootstrap 3 Tutorial. Retrieved June 25, 2015, from
http://www.w3schools.com/bootstrap/default.asp
http://bower.io/
http://karma-runner.github.io/0.10/config/configuration-file.html
http://adrianmejia.com/blog/2014/10/07/grunt-js-tutorial-from-beginner-to-ninja/
http://www.w3schools.com/angular/
https://docs.angularjs.org/guide/
https://github.com/angular/angular.js/wiki/Understanding-Scopes
http://www.sitepoint.com/practical-guide-angularjs-directives/
http://weblogs.asp.net/dwahlin/creating-custom-angularjs-directives-part-2-isolate-scope
http://stackoverflow.com/
https://github.com/TobiasWooldridge/UniBuddyTimetable
https://github.com/ngbp/ngbp