Plataforma per a la generació d'horaris per a centres docents

Xavier Martín Alonso

Resum– El projecte que es tracta en aquest document va sobre el desenvolupament d'una solució a un problema real que es troben els directors i coordinadors de cada un dels centres docents, la qual és la generació i gestió dels horaris escolars de cada professor i de cada grup d'alumnes. S'ofereix una possible solució que permeti facilitar i agilitzar aquesta tasca mitjançant una aplicació web. En l'article es presenta i s'explica aquesta solució, començant amb el primer estudi que es realitza, el disseny de l'aplicació que es fa a partir de les conclusions extretes del estudi, el desenvolupament del sistema i els resultats que s'aconsegueixen.

Paraules clau– Generar, gestor, horaris, centres docents, professors, assignatures, aules, cursos, grups, restriccions.

Abstract– The project discussed in this document is about the development of a solution to a real problem faced by school directors and coordinators, which is the generation and management of the school schedules for each teacher and each student group. A possible solution is offered that would facilitate and streamline this task through a web application. The article presents and explains this solution, starting with the first study that was carried out, the design of the application that was made based on the conclusions drawn from this study, the development of the system, and the results that were achieved.

Keywords– Generate, manager, schedules, educational centers, teachers, subjects, classrooms, courses, groups, restrictions.

1 Introducció - Context del treball

Avui en dia, a les escoles i centres docents en general, hi ha cada vegada més alumnes i per tant més classes a donar. Donat al creixement exponencial que està havent-hi en els centres, fa que als directors i a les persones que hi treballen els hi sigui cada vegada més complicat planificar els nous cursos escolars.
Davant d'aquesta problemàtica, en aquest article es veurà una possible solució a una de les principals dificultats que sorgeixen en conseqüència d'aquest creixement, el qual es tracta de la generació dels horaris per als nous cursos escolars, tant dels professors com els dels grups. Aquest és un problema que cada any els centres docents tenen molt present, ja que no és una tasca gens fàcil la de quadrar tots els horaris de les classes que es donen i els dels professors que imparteixen les classe, fet que també fa que sigui necessari invertir molt de temps i recursos a la creació d'aquests horaris, sobretot per als centres docents més grans, ja que com més alumnes hi haguin, més classes s'imparteixen i per tant més complicat es fa generar els horaris per al centre.
Per tant, en aquest projecte se'ns presenta aquest problema real que existeix en cada centre, en el que s'ha estudiat i generat una possible solució que permeti automatitzar recursos i temps, per a que aquesta tasca es pugui fer d'una manera més senzilla i amena.

2 Estat de l'art

La generació d'horaris de manera automàtica, no és una funcionalitat que es trobi en moltes aplicacions avui en dia per la complexitat que aquesta comporta, tot i així, sí que es troben algunes eines en el mercat que ofereixen aquesta opció, les quals en alguns centres ja utilitzen. Aquestes poden utilitzar tecnologies com la inteligència artificial, les quals poden donar bons resultats, tot i que també poden existir eines que no fascin ús d'aquesta.
Aquesta és una funcionalitat que comporta molt de temps en realitzar i consumeix molts recursos, per lo que es convenient que aquestes utilitzin altres eines o tècniques d'automatització.

• E-mail de contacte: 1460616@uab.cat
• Mención realitzada: Tecnologies de la Informació
• Treball tutoritzat per: Vladimir Bellavista Parent (Departament d'enginyeria de la informació i la comunicació)
• Curs 2022/23

Tot i que aquestes eines que hi han al mercat puguin donar bons resultats, el procés per a arribar a aquests és complicat, ja que aquestes no son eines que ofereixin una gran facilitat als usuaris alhora d'utilitzar-les, així com tampoc son accessibles des de qualsevol dispositiu, ja que la majoria d'aquestes eines son aplicacions d'escriptori, per el que comporta que funcionin sobre dispositius amb unes característiques determinades.

3 OBJECTIUS

Tal i com es diu en la introducció del article, els centres docents tenen dificultat per a generar els horaris dels cursos i dels professors que tenen, ja que hi han moltes variables que s'han de tenir en compte, sobretot si el centre docent és gran, ja que son més variables les que s'han de tenir en compte. Per tant, es consumeix més temps, recursos i també és més fàcil descuidar-se d'alguna d'aquestes variables i cometre errors. Aquest fet es un problema ja que poden sorgir riscos a les portes de començar un curs escolar, com pot ser que hi hagui algun professor que estigui impartint dos classes a la vegada sobre el mateix horari, que no s'hagui tingut en compte quants dies lectius té un professor i per tant, estigui donant classe més dies dels que li tocaria, etc. Aixins, molts més exemples d'errors que es poden trobar en els horaris dels centres al comença un curs o pitjor encara, amb el curs ja començat. Tots aquests riscos poden comportar despres greus conseqüències per als centres, ja que la seva imatge pot quedar afectada i per tant es poden produir perdues econòmiques per les baixes que puguin haver-hi.
Tots aquests riscos que s'han comentat podrien donar-se per errors en la planificació del curs escolar, per tant, el principal objectiu del projecte és desenvolupar una eina que generi aquests horaris de manera automàtica per tal d'evitar tots aquests problemes que poden sorgir, minimitzar els riscos i facilitar la tasca de generar els horaris per a tots els cursos i professors, així també permetent un millor control i planificació del curs escolar.
A part, també hi ha altres objectius, ja que aquest sistema que genera els horaris ha de ser de fàcil accessibilitat per als directors i coordinadors de cada centre, és a dir, que puguin accedir des de qualsevol dispositiu i que sigui fàcil e intuitiu d'utilitzar, ja que el fet de desenvolupar aquest sistema, ha de ser una solució per als centres i no un problema més.

Tal i com està fet el projecte, per a futures versions es poden establir nous objectius que facilitin encara més les gestió dels centres docents, com la de construir moduls que permetin gestionar les matrícules dels alumnes i vincular-los amb els grups existents, com també gestionar la contabilitat dels centres, podent així convertir el sistema en un ERP.

4 METODOLOGIA

Per a arribar a una proposta de solució s'han hagut de fer varies activitats abans de començar a desenvolupar el sistema per tal d'establir una metodologia de treball, és a dir, decidir una manera de desenvolupar l'aplicació per tal que compleixi de la millor manera els objectius que s'han marcat.
Primer de tot es va necessitar recollir informació per a saber com funciona un centre docent i quins mètodes s'utilitzen avui en dia per a generar els horaris. Un cop recollida aquesta informació, es va començar a traçar un pla per a resoldre el problema de la millor manera possible, per tant, primer de tot es va estudiar el problema on es va decidir quines tecnologies s'utilitzarien per a desenvolupar el sistema, es realitza un disseny de classes per a organitzar el sistema de tal manera que sigui el més robust possible i senzill per a fer-hi modificacions en un futur. Despres es va dissenyar la base de dades de tal manera que les dades que es generin en el sistema es guardin de forma coherent. Finalment es va començar el desenvolupament i a codificar els diferents algoritmes que generaran els horaris.

4.1 Recollida d'informació

En aquest primer procés, es va intentar captar tota la informació possible per a després poder generar una proposta de solució, fer el disseny d'aquesta i finalment implementar-la. Per això, es va necessitar estudiar com funcionen els centres docents a escala d'organització per a així poder establir uns atributs als centres, els objectes que tindran aquests i tenir una idea de com interconnectar aquests atributs i objectes en el disseny que es veurà en el següent apartat, per exemple, quins horaris lectius poden tenir els centres, si organitzen les classes en cursos i en grups, si els professors poden tenir uns dies lectius a la setmana, si tenen hores lliures, etc.
També es va estudiar com ho fan actualment els centres docents per a generar els horaris i portar l'organització d'aquests, quines eines hi ha i utilitzen per a fer-ho. Amb aquesta investigació es va trobar que existeix una eina anomenada GHC [1], una aplicació d'escriptori que recull totes les dades de cada centre (hores lectives, professors, dies lectius de cada professor, assignatures, etc.) i genera un horari per a cada professor. Per tant, es va descarregar e instal·lar aquesta aplicació per a testejar-la i agafar una idea de com desenvolupar la nova eina.
Al utilitzar l'aplicació GHC, es va veure que era una aplicació poc intuïtiva i difícil d'utilitzar, ja que la interfície d'usuari és molt poc atractiva, amb molts botons petits i en un principi no se sap que s'ha de fer ni quins passos s'han de seguir per a que l'aplicació et generi l'horari. A part, al ser una aplicació d'escriptori, aquesta consumeix molts recursos del dispositiu al generar els horaris, ja que és un procés molt complex. Des del punt de vista de l'accessibilitat, aquesta aplicació tampoc és de fàcil accés al haver de ser instal·lada en un ordinador, pel fet que només es podrà utilitzar des de l'ordinador on s'hagui instal·lat, el qual haurà de tenir unes característiques mínimes i uns recursos mínims per a poder executar-la correctament.
Un cop fet tot aquest procés i amb la informació necessària recollida, es va poder extreure quins són els problemes actuals a resoldre i unes conclusions sobre com es treballa i s'organitzen en els centres docents per a poder començar a fer un primer disseny de la solució a implementar.
En aquestes conclusions que s'extreuen, hem vist que cada centre pot gestionar tota la seva logística a la seva manera. Per exemple, al que respecte als professors, cada un d'aquests pot tenir uns dies lectius o uns altres depenent

del centre al que estiguin, al igual que amb les hores que poden tenir lliures al dia i quines. També cada centre pot gestionar les seves hores lectives que es faran, quants grups i cursos tindran, etc. Respecte als grups d'alumnes, cada centre també ha de poder gestionar els cursos que donarà i els grups que puguin haver-hi en cada curs, al igual que les assignatures que es donaran en cada grup i les aules en les que es podran donar certes assignatures.
Per tant, la solució a implementar ha de ser flexible en aquest aspecte per a que cada director pugui introduir les suficients dades per a poder gestionar el seu centre i en conseqüència, generar els horaris acord amb les dades que s'introdueixen.
Despres de veure el sistema GHC, s'ha arribat a la conclusió de que el projecte a implementar ha d'oferir una solució diferent i més senzilla per als centres, que els hi doni un valor afegit respecte a altres sistemes que puguin haver-hi en el mercat. Per això, tal com s'ha dit abans, aquesta nova solució haurà de permetre que es pugui accedir des de qualsevol dispositiu de manera senzilla i que un cop dins del sistema, es tingui tota la informació a gestionar a l'abast, amb una interfície d'usuari que sigui comprensible e intuïtiva per a qualsevol persona que pugui utilitzar el sistema.

4.2 Disseny base de dades

Una vegada es tenen fixats els objectius, les principals característiques i les funcionalitats que haurà de tenir el nostre projecte, les quals són extretes de les conclusions a les que s'han arribat en la recollida de la informació i de l'estudi previ, es comença a idear un primer disseny a implementar. Tot projecte que es vulgui desenvolupar correctament i amb èxit, s'ha de realitzar una primera fase de disseny abans de començar a desenvolupar la nostra solució, ja que ens servirà per a saber com s'haurà de construir e implementar el sistema, és a dir, ens servirà de guia en el moment de desenvolupar la solució, permetent així poder avançar-nos a futurs problemes que puguin sorgir en el desenvolupament del projecte, organitzar la feina a realitzar, aconseguir implementar una solució robusta i comprensible, que permeti implementar canvis amb facilitat sense afectar a altres processos del sistema i d'aquesta manera minimitzar possibles riscos que puguin sorgir en un futur.
Llavors, el primer que es fa, és decidir com serà la nostra base de dades i fer un primer disseny d'aquesta. Amb la informació recollida en el pas previ, el sistema haurà de guardar les dades de tots els objectes que els centres docents puguin tenir, com també les mateixes dades del centre. Aquestes dades seran les de les aules, les assignatures, els cursos, els grups, els professors i els horaris d'aquests professors i grups. A part, per al correcte funcionament del sistema, també s'haurà de guardar les dades dels usuaris que puguin accedir a l'aplicació. Aquests objectes tindran una relació entre ells, ja que n'hi ha que poden tenir dependències amb altres objectes.

A la figura 1 es mostra el model relacional que es dissenya per a la base de dades, amb totes les seves taules i relacions entre elles. Començant per la taula Centres, veiem que té relació amb la taula Professors, Aules, Cursos, Grups, Assignatures i Usuaris, ja que tots aquests objectes

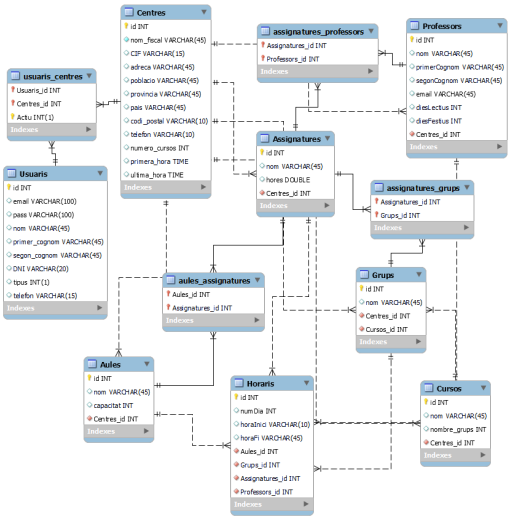

Fig. 1: Model relacional

són els que conformen el centre.
Per tant, les relacions es venen a llegir de la següent manera. La taula Centres tindrà una relació 1..n amb la taula Professors, ja que un centre podrà tenir n professors, però un mateix professor no podrà estar donant classes a més d'un centre. Amb la taula Assignatures també té una relació 1..n, per la mateixa raó, pel fet que cada centre crearà les seves pròpies assignatures en el sistema, per tant, una assignatura creada per un centre no podrà estar mai relacionada amb qualsevol altre centre. Per la mateixa raó, hi ha la mateixa relació 1..n amb les taules Aules i Cursos.
A part, la taula Centres també tindrà relació amb la taula Usuaris, però en aquest cas serà una relació n..n, pel que es genera una taula intermitja anomenada üsuaris centres"que permet aquesta relació. En aquest cas, es llegeix com que un centre podrà tenir n usuaris i un usuari podrà estar vinculat a n centres. Aquesta relació està pensada perquè en un futur pugui haver-hi un tipus d'usuari que pugui gestionar més d'un centre, per tant, haurà de tenir accés a aquests centres als quals està vinculat.
Respecte a les altres taules que conformen els centres, aquestes també tindran dependència entre elles. Per exemple, les aules i les assignatures tindran una dependència, ja que les assignatures es poden donar segons quines aules i, per tant, aquestes hauran d'anar assignades a les aules que hi haguin en el centre. D'aquesta manera, la taula Aula i la taula Assignatura, tindran una relació n..n, la qual significa que una aula pot estar assignada a més d'una assignatura i una assignatura a més d'una aula. Per aquesta raó es crea la taula intermitja "aules assignatures", per a fer la relació entre les dues taules de la mateixa manera que hem vist amb els centres i els usuaris.
Les mateixes assignatures, també tindran dependència amb els professors, ja que cada assignatura haurà d'estar assignada a un dels professors que tingui el centre, per la qual cosa cada un dels professors només podrà donar classe de la seva matèria. Un professor podrà donar més d'una assignatura i una assignatura podrà ser donada per a més d'un professor en el centre, per tant, aquestes dues taules també tindran una relació n..n, per lo que es genera la taula intermitja "assignatures professors".
Respecte als professors, aquests també hauran d'estar

assignats als grups d'estudiants que puguin donar classe, ja que està pensat per a que hi hagui la possibilitat que cada professor pugui donar classe a uns grups concrets si s'escau. Així que la taula Professors tindrà també una relació n..n amb la taula Grups, per el que es genera la taula intermitja "assignatures grups", perquè un professor pugui donar classe a més d'un grup i que cada grup pugui tenir més d'un professor que doni classe.
Respecte als grups, aquests també hauran d'anar assignats a un curs del centre. Per tant, aquests també tindran una relació amb la taula Cursos. En aquest cas, serà una relació 1..n, ja que cada curs del centre, pot tenir més d'un grup d'estudiants, mentres que un grup, només podrà estar assignat a un dels cursos del centre. D'aquesta manera, es té pensat que el sistema permeti organitzar els estudiants del centre amb cursos segons les edats i si en aquests hi han masses estudiants, que es puguin dividir les classes en 2 grups. Per exemple, podrà haver-hi el primer curs de la ESO amb el grup A i el grup B, tot i que es permet donar un nom a cada un dels cursos i centres per a permetre que cada coordinador organitzi els seus grups d'estudiants de la manera que millor li convingui.
Finalment, les taules Aules, Grups, Assignatures i Professors, estaran assignades a un horari, per el que cada una d'aquestes tindran una relació 1..n amb la taula Horaris. En aquesta última és on es guardaran els horaris dels centres, pel que és necessari guardar l'assignatura que es donarà, el professor que la donarà, en quina aula es donarà i per a quin grup. A part també es guarda el dia de la setmana, l'hora d'inici i la final de la classe.
Pel que fa al dia de la setmana que es guarda, serà un valor numèric de l'1 al 5, de tal manera que el dia 1 representa el dilluns, el dia 2 el dimarts, etc.

Amb totes aquestes taules i relacions, tindríem la base de dades llesta per a emmagatzemar les dades necessàries per al correcte funcionament de l'aplicació i sobretot, per a generar els horaris.
Aquesta tasca de disseny de la base de dades és important per a tenir totes les dades organitzades de manera coherent i que d'aquesta manera hi hagui una consistència.

4.3 Disseny del sistema

Un cop tenim el disseny de la base de dades llest i revisat, procedim a fer un disseny de com serà el nostre sistema. Per començar, decidim el llenguatge de programació i tecnologies que s'utilitzaran per a construir l'aplicació.

Com s'ha comentat, haurà de ser una aplicació que pugui ser de fàcil accés per a tothom i des de qualsevol dispositiu, per el que haurà de ser una aplicació web, de tal manera que s'hi pugui accedir des d'un navegador. Partint d'aquest punt, es decideix que l'aplicació es construeixi amb el llenguatge de programació PHP, ja que és un llenguatge creat per al desenvolupament web. Aquest llenguatge s'executa en servidor, en què un cop estigui l'aplicació allotjada, ens servirà per a agafar i tractar las dades de la base de dades, i muntar les vistes que es mostraran als usuaris. Respecte a aquestes vistes, es construiran en HTML, CSS i JavaScript, ja que aquestes tecnologies ens permetran donar estils i comportaments a les nostres vistes de manera senzilla. També es té en compte que PHP permet construir aquestes vistes des del servidor, pel fet que interpreta el codi HTML, per tant, és una bona opció per a mostrar les dades extretes de la base de dades a les vistes de manera senzilla. També permet incloure codis HTML a través de funcions de PHP, el qual li podrem treure profit a l'hora de separar el codi de tal manera que quedi més organitzat i comprensible, com també ens permetrà evitar molta repetició de codi.
Al construir les vistes amb les tecnologies esmentades anteriorment, ens ajudarem amb altres llibreries d'estils i una plantilla ja creada, la qual ens servirà per a agafar components amb uns estils ja predefinits, però que podrem modificar al nostre gust. Aquestes llibreries d'estils que utilitzarem és bootstrap i fullcalendar. La primera ens servirà per a donar uns estils predeterminats als components que formin les nostres vistes, les quals després ens permetrà modificar o afegir-hi de més, però sobretot ens serà de molta ajuda per a fer l'aplicació tingui un disseny adaptatiu de manera senzilla, és a dir, que el disseny de cada vista s'adapti a la pantalla de qualsevol dispositiu. Respecte a la llibreria fullcalendar JS, ens servirà per a incorporar un calendari a la nostra vista, el qual ens podrà servir per a mostrar els horaris de cada professor i de cada grup.
Un cop es tenen clares les tecnologies i llenguatges que es faran servir per a construir el sistema, fem un primer disseny de com funcionarà el nostre codi, fent un diagrama de classes [2] i considerant els patrons de disseny [3] que ens puguin anar bé per al desenvolupament de l'aplicació. D'aquesta manera, tindrem una pauta de com desenvolupar el sistema de manera robusta, d'organitzar el nostre codi i evitar problemes que puguin sorgir en futures implementacions. També ens permetrà tenir un codi net i organitzat, de tal manera que sigui senzill d'entendre per a qualsevol altra persona, que es puguin fer canvis i/o afegir més implementacions al sistema sense que suposi un cost molt elevat. Així també evitem possibles errors que puguin sorgir en altres parts del programa a l'hora d'implementar un canvi en aquest.

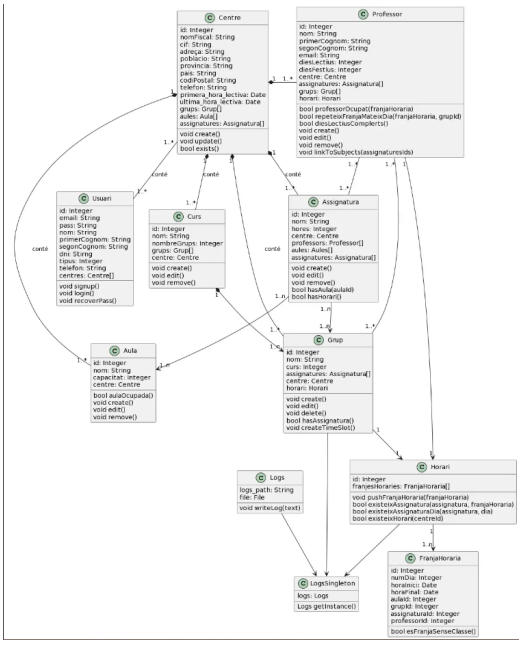

Fig. 2: Diagrama de classes del model

Amb totes aquestes consideracions, es genera el diagrama de classes de la figura 2. En aquest es pot observar com de la classe Centre pengen totes les altres classes que la conformen, les quals són les classes Professor, Assignatura, Aula, Curs i Grup. Aquestes tindran una relació de composició, la qual es representa amb el rombe negre a l'extrem de la classe pare, en aquest cas en la classe Centre. Aquesta relació ve a dir que la classe Centre està composta per les classes que hem esmentat, per el que aquestes no podran existir si no existeix un centre abans, per exemple, una aula no podrà ser generada si no existeix un centre on relacionar-la. Aquesta mateixa situació, per tant, també s'aplica a les classes que estan relacionades amb la classe Centre de tal manera que tinguin una relació de composició. En aquestes relacions també es poden veure les cardinalitats, les quals ens indica quants objectes d'una classe van relacionats amb els objectes de l'altra classe. En aquests casos que hem comentat, es pot veure en les cardinalitats indicades que una classe Centre podrà tenir només un objecte centre relacionat amb 1 o més objectes de les classes Curs, Grup, Assignatura, Professor i Aula.
Per una altra banda, la classe Centre també tindrà una relació amb la classe Usuari, ja que aquest haurà de poder interactuar amb el centre en el qual estigui vinculat i amb tots els seus components. Per tant, aquestes dues classes estaran relacionades amb una associació bidireccional, de tal manera que es pugui recorre en els dos sentits, és a dir, que des de la classe Centre es puguin obtenir els usuaris vinculats i des de la classe Usuari es pugui obtenir també els centres als quals està vinculat. Respecte a la cardinalitat, en aquest cas un centre podrà tenir més d'un usuari vinculat i un usuari podrà estar vinculat a més d'un centre, per tant, seguirà una cardinalitat 1..n en ambdós sentits.
Els components de cada centre, també estaran relacionats entre si. Primer veiem que la classe Professor té una relació d'associació amb la classe Assignatura, amb la cardinalitat 1..n en ambdós sentits. Això es llegeix que un professor pot donar classe de més d'una assignatura en el centre i que una assignatura també pot ser donada per a més d'un professor del centre. També serà una associació bidireccional per a que es pugui accedir als objectes de cada una de les classes amb ambdós sentits com en l'exemple anterior.
La mateixa classe Professor també tindrà una relació d'associació amb la classe Grup, amb la cardinalitat 1..n en ambdós sentits també, com en el cas vist anteriorment, ja que un professor haurà d'estar assignat als grups als quals podrà donar classe i cada grup també podrà tenir més d'un professor que pugui donar classe al mateix grup.
Aquesta mateixa classe Grup, també haurà d'estar relacionada amb la classe Curs, ja que com s'ha explicat, cada curs del centre tindrà grups d'alumnes per a gestionar les classes. Per tant, serà una relació de composició, perquè un grup només podrà existir si hi ha algun curs al que poder vincular-lo. Per al que fa a la cardinalitat, un curs podrà tenir més d'un grup, però un grup només podrà estar vinculat a un sol curs.
Tornant a la classe Assignatura, aquesta també tindrà una relació amb les aules, ja que depenent de l'assignatura es podrà fer en una aula o en una altre, per tant tindrà una relació d'associació amb cardinalitat 1..n en ambdós sentits, perquè en una aula es podrà donar més d'una assignatura i una assignatura podrà ser donada en més d'una aula del centre.
La mateixa classe Assignatura tindrà una última relació amb la classe Grup, ja que per a cada grup se li podrà assignar quines assignatures es donaran, per tant, aquestes dues classes tindran també una relació d'associació amb cardinalitat 1..n en ambdós sentits, pel que una assignatura es podrà donar en més d'un grup i en un grup es podran donar més d'una assignatura..
Després hi haurà les classes més importants del sistema per a la generació d'horaris, la qual és la classe Horari i FranjaHoraria. La solució que s'ha pensat és la d'assignar una classe Horari per a cada Grup i Professor, on es guardaran franges horàries les quals representaran una hora del dia. En aquestes franges s'assignaran les assignatures, professors, aules, grups, el dia i les hores en què es donarà aquella classe. Amb tota aquesta informació es podrà muntar l'horari a mostrar, és per això que es necessita que la classe FranjaHoraria estigui relacionada amb la classe Horari, de manera direccional, de tal manera que a les franges horàries si puguin accedir des de la classe Horari. La cardinalitat serà de "1"a "1..n", ja que un Horari haurà de tenir tantes franges horàries com siguin necessàries per a crear l'horari, mentre que una franja horària que es crei, estarà vinculada a un sol horari.
Com que seran els professors i els grups els que tindran un horari, seran aquestes classes les que estaran vinculades amb la classe Horari, totes dues amb una relació d'associació direccional cap a Horari, ja que als horaris només es podran accedir des de les classes Grup i Professor. Respecte a la cardinalitat, de moment es té pensat que un grup i/o professor només tingui un horari, per tant, serà de "1"a "1", de tal manera que els professors i grups només puguin tenir un horari i que un horari només pugui estar vinculat a un professor o grup.
Com que implementar un sistema de generació d'horaris té moltes variables i restriccions, és molt fàcil que hi haguin errors en el sistema, és per això que s'ha pensat en implementar un sistema de registres per a saber en tot moment que està fent l'algoritme de generació dels horaris i així veure si està funcionant correctament. D'aquesta manera es crea una classe Logs que permetrà escriure cada registre en un fitxer de text pla, el qual es cridarà des de la classe Grup, ja que és la que contindrà la classe que generarà les franges horàries per a cada grup. Aquesta funcionalitat, però, es vol que es pugui cridar des de qualsevol punt del programa a part de la classe Grup, per tant, es construeix seguint el patró de disseny Singleton, el qual permetrà que només hi hagui una instància de la classe Logs en tot el programa. D'aquesta manera, a part d'estalviar recursos, evitarem conflictes en l'escriptura dels fitxers. És per això, que es crea una classe LogsSingleton, la qual crearà una sola instància de la classe Logs per a tot el programa amb la funció getInstance, la qual comprovarà si existeix una instància o no. Si existeix una instància, simplement retornarà aquesta i si no, simplement la crearà i la retornarà.

Un cop explicat el model i les classes que hi haurà en el sistema, passarem a veure com aquest interactuara amb la base de dades, en el que es seguirà el patró de disseny DAO [4] per a la connexió a aquesta.

Primerament, hi haurà una classe Connexió la qual se-

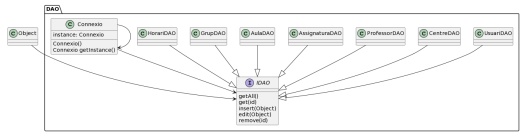

Fig. 3: Diagrama de classes de la connexió a la base de dades

guirà el patró de disseny Singleton com en el cas dels registres. En aquesta classe hi haurà la funció getInstance que retornarà una instància de l'objecte MySQLi de PHP que permetrà connectar amb la base de dades i manipular les dades. D'aquesta manera no es crearan connexions innecessàries a la base de dades, s'estalviaran recursos i s'evitarà que la base de dades tingui molta càrrega per les connexions.
Un cop es té la classe Connexió que proporcioni una connexió a la base de dades MySQL, creem una classe DAO per a cada classe creada en el model. En aquestes classes és on es cridarà a la funció getInstance de la classe Connexió per a obtenir la connexió a la base de dades i realitzar les consultes necessàries, és a dir, aquestes classes DAO seran les encarregades de consultar i manipular les dades de la base de dades, per la qual cosa seran les classes que contindran les consultes SQL.
Aquestes classes DAO estaran creades amb una interfície anomenada IDAO (Interface DAO) de la qual s'heretaran les funcions necessàries per a realitzar el CRUD (create, read, update i delete) de cada objecte.
Tot aquest disseny correspon al model de l'aplicació, ara bé, respecte a l'arquitectura del sistema, se seguirà la metodologia model-vista-controlador (MVC), la qual separa el codi del sistema segons l'ús que tingui aquest, és a dir, el codi encarregat de mostrar les vistes per pantalla, anirà a la part de la vista. Generalment, aquest codi serà tot el codi HTML, JavaScript, CSS, imatges, etc.
En la part del controlador, anirà el codi PHP encarregat de gestionar i mostrar totes les vistes segons les dades que li arribin. I finalment, a la part del model hi haurà tot el codi que gestiona i manipula les dades del sistema. En aquesta part és on es construirà en codi PHP tot el disseny del model que s'ha creat amb el diagrama de classes.

4.4 Desenvolupament

Quan tenim tota la idea i disseny de el que serà el nostre projecte, es comença a implementar aquesta solució al problema plantejat.
Primer de tot es comença per a preparar l'entorn de desenvolupament, per tant, s'instal·len i es configuren totes les eines necessàries a l'equip on es desenvoluparà el sistema.
Respecte a la base de dades, s'instal·la i s'utilitza el MySQL Workbench, un programa que et permet gestionar connexions a diferents servidors de bases de dades i executar sentències SQL sobre aquestes. És una eina gratuïta molt potent, ja que també et permet muntar els models relacionals de manera senzilla, permetent visualitzar de manera gràfica les taules i les relacions que tenen entre elles per a després construir la base de dades automàticament a partir del disseny del model relacional creat. Per tant, un cop instal·lat MySQL Workbench i tots els components que utilitza, es munta el model relacional mostrat i explicat en el punt 4.2. Un cop es té creat el model relacional que es va dissenyar, amb una eina que ofereix el Workbench anomenada Forward Engineering, es genera l'script de creació de la base de dades a partir del model relacional creat. Aquest script es pot guardar tot i que el mateix programa et permet executar l'script automàticament, fet que fa que la creació de la base de dades amb totes les seves taules i relacions, sigui una tasca més senzilla i ràpida. Aquesta eina, però et genera la base de dades des de 0, pel que si aquesta ja està creada i funcionant, és a dir, amb dades ja guardades, no és la millor eina a utilitzar si es necessita afegir un canvi en l'estructura de la base de dades des del model relacional, ja que esborraria totes les dades que ja hi han guardades. En aquest cas, Workbench ofereix l'eina Sinchronize Model, la qual permet sincronitzar la base de dades ja creada amb el nou model relacional, detectant els canvis que s'han fet en el model respecte a la base de dades. Igual que Forward Engineering, també genera un script que executa automàticament, però en aquest cas, només aplica a la base de dades els canvis realitzats en el model, sense eliminar cap dada i evitant així que el sistema es pugui comprometre.
Paral·lelament, es descarrega i s'instal·la un servidor web local a l'equip. En aquest cas s'instal·la l'eina XAMPP, ja que porta un servidor Apache incorporat i l'intèrpret de PHP, el qual servirà per a interpretar i executar els fitxers i el codi escrit en PHP. A més d'aquests serveis que ens instal·la XAMPP, també ens ofereix d'altres serveis que de moment no es faran servir, però que poden ser d'utilitat en un futur.
En aquest servidor d'Apache és on s'allotjarà la nostra aplicació web en local per al procés de desenvolupament.
Per a no tenir tot el codi en local, també es descarrega e instal·la l'eina git per a pujar tot el codi i canvis que es vaguin realitzant a un repositori de Github. Al treballar un sol desenvolupador, aquesta eina no és molt necessària, però al tenir com a un dels objectius allotjar l'aplicació web a un servidor d'AWS, ens serà d'ajuda per a pujar el codi i tots els canvis de manera senzilla amb una eina d'integració continua que ofereix Amazon.
També cal descarregar e instal·lar un editor de text, en aquest cas s'utilitza Visual Studio Code, ja que és senzill d'utilitzar i molt lleuger.

Un cop tenim tot el programari instal·lat i amb la base de dades creada i funcionant, es comença a codificar l'aplicació web, on primer de tot es crea tota l'estructura de carpetes i fitxers.

En la figura 4 es mostra l'estructura de fitxers creada. Primer de tot es crea un fitxer index.php el qual servirà d'encaminador. Aquest agafarà el paràmetre view de l'URL, el qual el seu valor dirigirà al controlador corresponent en què es gestionarà les vistes a mostrar del controlador corresponent. Si aquest paràmetre view no existeix o no es troba el fitxer del controlador corresponent, el mateix encaminador mostrarà una pàgina d'error 404.
Respecte al fitxer ajax.php, serà l'encarregat de cridar als fitxers que contenen les funcions ajax a realitzar quan hi ha una crida d'aquestes. Aquests fitxers ajax es troben en el directori core/bin. En aquesta carpeta core, serà on s'allotjarà la part del model i el controlador de l'arquitectura MVC esmentat anteriorment. A part de la

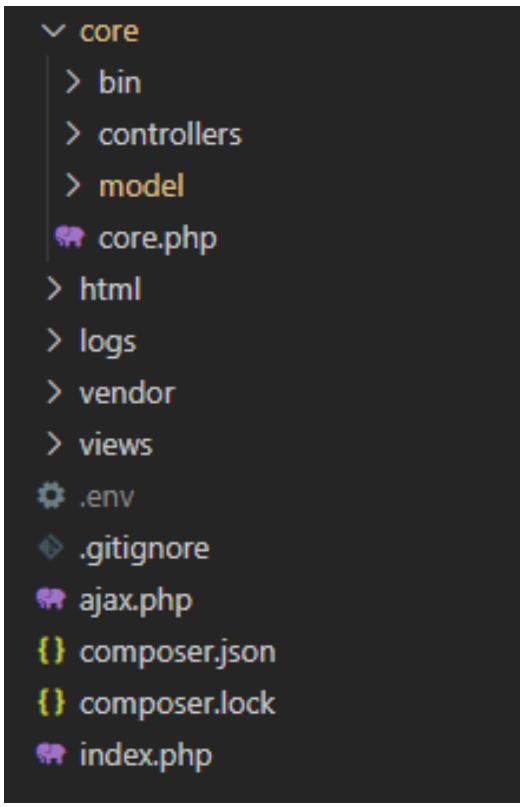

Fig. 4: Estructura de fitxers

carpeta bin que s'ha dit, hi ha la carpeta controllers, la qual s'allotjaran els fitxers controladors, els quals s'invocaran des del fitxer encaminador segons el paràmetre view de la URL que agafarà aquest mateix. Un cop s'invoca un fitxer controlador, aquest serà l'encarregat d'agafar altres paràmetres de l'URL i/o del model, que li permetrà muntar i mostrar una vista o una altra.
Respecte a la carpeta model, és on s'allotjaran tots els fitxers de les classes dels models, amb tots els seus atributs i funcions. En aquesta mateixa carpeta, hi haurà una altra anomenada DAO on s'allotjaran les classes DAO de cada objecte i on hi haurà totes les sentències SQL.
El fitxer core, serà el nucli de l'aplicació, ja que inclourà tots els fitxers del model, constants i tots els paràmetres de configuració de l'aplicació. Aquest fitxer es cridarà des de l'encaminador, per el que tots els fitxers que s'incloguin i totes les variables que s'inicialitzin en aquest, estaran disponibles en tota l'aplicació.
En la carpeta html hi hauran les vistes de l'aplicació, és a dir, tots els fitxers html que s'aniran invocant des dels controladors corresponents.
En la carpeta logs hi haurà tots els fitxers de text pla, amb els registres que es creen alhora de la generació dels horaris.
En la carpeta vendor s'allotjaran totes les dependències de l'aplicació. Aquestes s'instal·len amb l'eina Composer, la qual crea el fitxer composer.json i composer.lock a l'arrel del projecte. En el fitxer composer.json s'indica les dependències a instal·lar, amb la seva versió, les quals Composer es descarrega i allotja a la carpeta vendor amb la comanda çomposer install"o çomposer require ¡dependència¿"si la dependència a descarregar no està especificada en el fitxer composer.json.
Per últim, en la carpeta views s'allotjaran tots els fitxers d'estils CSS, fitxers JavaScript, imatges i altres dependències en la part de la vista. Aquestes s'inclouen des dels fitxers de les vistes, és a dir, des dels fitxers html, els quals es poden executar des de la part del client, en aquest cas des del navegador web.
També hi ha un fitxer .env on es guardarà les variables d'entorn, les quals seran per a informació delicada, com les credencials per a la connexió a la base de dades. Per a llegir aquestes variables, però, cal instal·lar la dependència phpdotenv amb Composer. Per aquest motiu es crea la carpeta vendor, ja que conté tots els fitxers de la dependència.
Per a que aquestes funcionin, s'han de carregar al sistema, invocant el fitxer autoload.php que crea en la mateixa eina Composer dins del directori vendor. Aquest fitxer s'invocarà des del fitxer core.php.

Un cop explicada l'estructura de carpetes de l'aplicació, el que primer es fa és codificar el model de l'aplicació, pel que es crea en la carpeta model tots els fitxers de cada classe, ja que cada un representarà una classe del diagrama de classes. Aquestes també es codifiquen pensant en les relacions que tindran entre elles i amb els patrons de disseny que s'utilitzaran, per tant, les classes tindran els atributs necessaris per a poder guardar el/s objecte/s de la classe relacionada. També creen les classes DAO per a cada classe i la interfície IDAO, la qual la implementarà cada classe DAO, ja que totes aquestes contindran les funcions amb les sentències SQL per a obtenir i manipular les dades de la base de dades. Aquestes funcions són les d'obtenir tots els objectes de la classe, obtenir un objecte, crear un objecte, actualitzar un objecte i eliminar un objecte, totes aquestes sobre la base de dades.
Es programa amb la idea que només es podran cridar als objectes DAO des de les classes del model, evitant així que s'haguin de fer les consultes des de la vista o els controladors, tenint d'aquesta manera totes aquestes consultes centralitzades en un punt de l'aplicació, reutilitzar les consultes i evitar repetició de codi, i tenint organitzat el codi de l'aplicació.
Respecte als controladors, es codificaran de tal manera que aquests recullin les dades que es retornen des del model per a després mostrar-les per pantalla en les vistes corresponents. Un controlador representarà una classe del model, el qual es cridarà segons la variable view que anirà definida en l'URL, però en cada controlador es gestionen més d'una vista, per el que aquestes es gestionen a partir d'un altre paràmetre que també vindrà de l'URL, la qual es dirà mode. Aquesta ens serveix per a saber quina vista hem de mostrar respecte a un objecte, per exemple, la vista on es mostri una llista amb tots els objectes, un formulari per a la creació de l'objecte, la d'edició de l'objecte, etc.
Per al que fa a les vistes, aquestes es construeixen amb html, seguint un estil d'una plantilla anomenada Inspinia com a referència per a després afegir-li els nostres estils. Aquesta porta bootstrap i altres llibreries que utilitzem i adaptem al nostre gust i a les nostres necessitats per a crear la vista de l'aplicació.
A partir d'aquí es comencen a desenvolupar les funcionalitats de l'aplicació, començant primer per al sistema d'inici de sessió i de registre d'usuaris, ja que aquestes seran les primeres vistes que es mostraran a l'entrar a l'aplicació.

Quan l'usuari inicia sessió, l'aplicació consulta si les credencials són correctes o no. En cas afirmatiu, es guarda la id del usuari en un variable de sessió, la qual es comprova si existeix i és correcte constantment a l'aplicació per a que l'usuari pugui accedir a totes les vistes. Si les credencials són incorrectes, no es guarda cap variable de sessió, per el que l'usuari no passarà de la vista d'inici de sessió i de registre.
Un cop l'usuari ha iniciat sessió, es cridarà a la vista del dashboard, la qual serà la vista principal dins l'escenari en què hi ha una sessió oberta, des del qual hi ha es pot accedir a totes les funcionalitats que té l'aplicació.
Aquesta mateixa vista principal i totes les altres vistes, estan dividides en 3 parts, les quals són una barra de navegació a la part d'adalt, on de moment hi haurà un botó per a tancar la sessió, una barra de navegació a la part esquerra de la vista i la part central de la vista. En aquesta part central, serà on es mostrarà la vista de la funcionalitat on es troba l'usuari en aquell moment, és a dir, si està en la vista per a editar les dades del centre docent, en aquesta part central es mostrarà el formulari corresponent per a omplir les dades necessàries del centre.

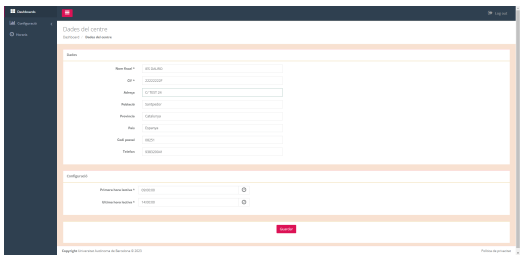

Fig. 5: Disseny de l'aplicació

Respecte a la barra de navegació que hi ha a l'esquerra, és on es mostren i et permet accedir a totes les funcionalitats que ofereix. Aquestes es poden veure en la figura 6.

Aquestes funcionalitats es mostren per ordre d'execució amb la intenció de guiar a l'usuari, és a dir, en primer lloc, el que s'hauria de fer és anar a la part de configuració del centre per a entrar totes les dades necessàries per a que la generació d'horaris funcioni correctament. Per tant, el que l'usuari es trobarà primer serà l'opció de la configuració del centre, ja que hi ha paràmetres que són necessaris omplir.
Un cop les dades del centre ja estan emplenades, el que segueix és començar a crear els objectes necessaris que tindrà el centre, com són les aules, les assignatures, els professors i els cursos. Aquests objectes s'haurien d'omplir amb aquest mateix ordre que es mostra en el menú de navegació, ja que com que aquests objectes tenen dependència entre si, l'aplicació s'ha pensat en què s'hagin d'introduir en aquest ordre per anar relacionant els objectes que es van creant, per exemple, si primer es crea una aula, en el següent pas de crear una assignatura, al crear-ne una se li haurà d'assignar en quines aules es pot donar aquesta assignatura, aules que ja s'han creat en el pas anterior. Com amb aquest exemple, passa el mateix amb la resta d'objectes, no es pot crear un objecte o acabar de crear-lo fins que l'anterior no s'ha creat.
Cada un d'aquests objectes es mostren com a un llistat dels objectes que hi ha, amb la possibilitat d'afegir, editar i eliminar. En el cas d'afegir i eliminar un objecte, es mostra

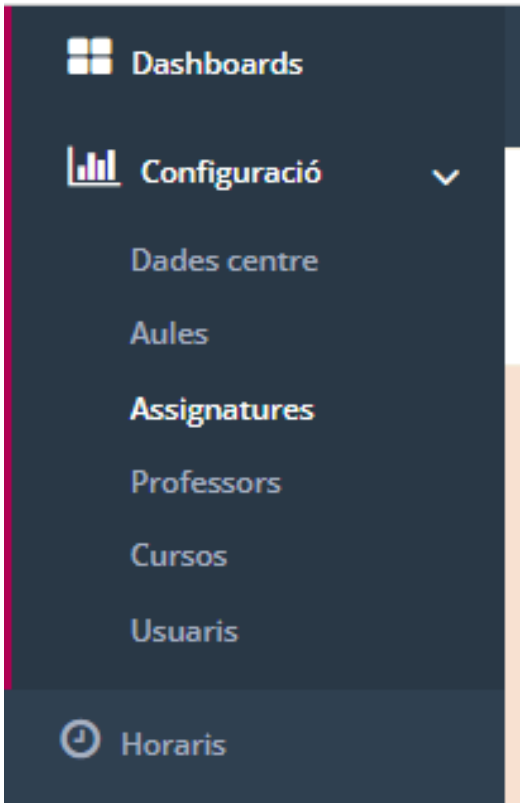

Fig. 6: Barra de navegació

una vista amb un formulari per a emplenar tots els atributs de l'objecte, igual que en la figura 5. Després, per darrere funciona de tal manera que una funció JavaScript captura les dades introduïdes en el formulari per a enviar-les al servidor mitjançant la tecnologia Ajax, la qual ens permet mostrar una icona de càrrega fins que no s'acaba tot el procés. El servidor quan rep aquestes dades, es crea l'objecte corresponent de la classe que hi ha creada del model amb les dades que arriben de la vista, després es crida a la funció que correspongui segons l'acció que s'estigui fent sobre l'objecte, per exemple, si s'està creant un objecte, es cridarà a la funció de create que té, la qual cridarà a la funció create de la seva classe DAO per a guardar les dades a la base de dades sentències SQL.
Per a recuperar els objectes, es fa el mateix procediment però a la inversa. Es crida a la classe DAO que correspongui a l'objecte que es vol recuperar, la qual agafarà les dades de la base de dades amb sentències SQL, crearà l'objecte i el retornarà.
Aquest procediment es seguirà per a tots els objectes, inclòs el mateix centre i els usuaris.

Un cop es té desenvolupat tot al que es refereix als objectes i al sistema d'inici de sessió (registre d'usuaris inclòs), es comença a desenvolupar l'algoritme per a la generació d'horaris amb la idea de les franges horàries explicades en el disseny del sistema.
Aquest es fa a partir d'una crida ajax, ja que serà una tasca que es demorarà en el temps per a la quantitat de comparacions que s'hauran de fer. Per tant, es parteix d'una funció JavaScript que capturarà les dades de la vista per a la generació d'horaris. En aquesta hi haurà un formulari amb les dades que no corresponguin a cap objecte i sigui necessari

establir-les abans de cada creació d'horaris. Ara per ara, de moment hi haurà per a seleccionar en quines hores de cada un dels dies pot no haver-hi classe. Per tant, a la funció JavaScript captura aquestes dades i les envia al servidor per a començar amb la generació.
Primer de tot, el que es fa és comprovar que no hi haguin horaris ja creats, ja que de moment, el sistema està pensat per a que cada grup i professor només tingui un horari, per tant, en el cas que ja existeixin, s'eliminen els horaris per a tornar-los de generar de nou. Està pensat així per a controlar millor si els horaris estan ben generats i sobretot perquè a l'utilitzar un algoritme de força bruta, cada generació tindrà el mateix resultat, per tant, no val la pena tenir més d'un horari generat per a cada grup i professor, ja que serà el mateix. En un futur es poden implementar diferents ordres d'execució per a que hi haguin diferents horaris i, per tant, es podran guardar més d'un horari per grup i professor, i que l'usuari pugui triar el que millor convingui.
Seguidament, es calculen les hores que es faran d'activitat docent al centre. Aquestes es calculen recuperant l'hora inicial i la final d'activitat docent que té el centre, les quals s'han d'haver introduït en la configuració del centre al primer pas de la configuració.
Un cop es tenen aquestes dades inicials, es comença el bucle principal per a la generació d'horaris. Aquest primer bucle recorre els dies de la setmana que es dona classe, els quals seran de l'1 al 5 representant els dies de la setmana de dilluns a divendres. Dins d'aquest hi ha un altre bucle on es recorren les hores, on a cada iteració es calcula l'hora de la franja horària. D'aquesta manera anirem recorrent les hores d'un dia primer, després les del següent i així successivament. Dins d'aquest bucle de les hores es recorren els grups que hi ha en el centre per anar creant les franges horàries sobre el grup.
Un cop sabent on estem (en el dia, hora i grup), es comença a recórrer els objectes que se li han d'assignar a la franja horària i a aplicar les restriccions. Primer de tot, però, es mira si aquella hora és una de les hores sense classe que introdueix l'usuari a la vista per a generar els horaris, ja que si és una hora sense classe se surt del bucle i s'evita que s'hagi de fer totes les comparacions de l'algoritme. Si aquesta no és una hora sense classe, es comença a recórrer les assignatures que es donen en aquell grup que s'està mirant per a guardar-la en la franja horària. Dins d'aquest bucle de les assignatures, es comencen a aplicar les restriccions per al que respecte a l'assignatura, les quals són les següents:

- Comprovar que no es faci la mateixa assignatura en la mateixa franja horària
- Comprovar que es dona l'assignatura en el grup.
- Comprovar que no es doni l'assignatura en el mateix dia.
- Comprovar que no se superin les hores setmanals de l'assignatura.

Si una de les restriccions no es compleix, es passa a mirar la següent assignatura fins a trobar una que compleixi totes les restriccions. Si una es compleix, aquesta s'afegeix a la franja horària i seguidament s'entra en un altre bucle on es recorren les aules on es poden donar l'assignatura que s'està mirant. Per tant, en aquest bucle es mira les restriccions que poden tenir les aules, que de moment només es comprova que l'aula no estigui ocupada per cap altre franja horària. Si l'aula que s'està mirant està lliure, s'assigna a la franja horària i s'entra en un altre bucle on es passa a recórrer i a mirar els professors que poden donar l'assignatura. Dins d'aquest, per al que respecte als professors, es miren les següents restriccions:

- Comprovar que el professor no estigui ocupat.
- Comprovar que el professor no doni la mateixa assignatura al mateix grup en el mateix dia.
- Comprovar que el professor no superi els dies lectius que pot donar.

El bucle on es recorren els professors és l'últim i, per tant, el més intern. Si algun dels professors supera les restriccions i, per tant, pot donar classe, s'assigna a la franja horària i es dona com a completa, per el que es guarda la franja a la base de dades, es passa a mirar la següent hora i es torna a començar tot el procés de restriccions de nou.
Un cop totes les franges horàries estan guardades a la base de dades, les podem fer servir per a mostrar-les en la vista. Per a fer-ho, es fa servir la llibreria fullCalendar [5], la qual et permet mostrar els horaris de forma senzilla creant uns esdeveniments en un "array", on s'introdueix el títol del esdeveniment (noms de l'assignatura i el professor), l'hora d'inici i final de l'esdeveniment.

En la generació dels horaris, és difícil saber si aquests s'estan generant correctament, ja que hi han moltes variables a l'hora d'aplicar les restriccions, és per això que va ser necessari muntar un sistema de registres per a saber que fa l'algoritme en tot moment. Aquest es crea tal com es mostra i s'explica en el disseny del diagrama de classes, per tant, es creen les classes Logs i LogsSingleton per a obrir i escriure en el fitxer dels registres des de qualsevol punt de l'aplicació i sol·licitant cada vegada la instància ja creada de l'objecte Logs que escriu aquests registres, evitant d'aquesta manera que puguin haver-hi més d'un objecte escrivint els registres i entrin en conflicte.
Un cop creades aquestes classes, des de l'algoritme que genera els horaris s'obté la instància de la classe Logs a través de LogsSingleton i s'escriu en el fitxer el text que millor convingui a cada situació.

Finalment, quan es té la generació d'horaris ja funcionant, s'allotja el sistema en un servidor, en aquest cas s'allotja en un servidor d'Amazon Web Service.
Per a allotjar-la i desplegar-la, s'utilitza el servei Elastic Beanstalk d'amazon, el qual es crea un servidor Apache amb la versió 8.1 de PHP. Aquest es crea de forma senzilla seguint els passos que el mateix servei indica i quan aquest està configurat, es puja el codi del projecte mitjançant un fitxer comprimit amb zip. El projecte es pot visualitzar ja funcionant en el servidor des de la següent URL: http://schedulegenerator-env.eba-7smdkxny.us-east-1.elasticbeanstalk.com.
El següent pas que es fa és la de crear una base de dades en el servidor Amazon per a que l'aplicació ja allotjada hi pugui connectar. Per a fer-ho s'utilitza el servei RDS que ofereix Amazon, on es crea i configura un lloc per a allotjar la base de dades de manera senzilla seguint els passos

que ofereix el servei. Un cop creat el lloc, es proporciona les dades per a la connexió a la base de dades, on s'utilitzen per a connectar-hi a través de MySQL Workbench. Un cop s'aconsegueix connectar, es crea la base de dades a partir del mateix model relacional que es crea per a muntar la base de dades en local, per tant, de la mateixa manera que s'explica en el disseny de la base de dades, s'utilitza l'eina "Forward Engineering"la qual crea i executa un script de creació a partir del model.
Un cop creada la base de dades amb totes les taules i relacions, en el codi es controla la connexió a aquesta de tal manera que si l'aplicació està en l'entorn de producció, es faci la connexió a la base de dades amb les credencials corresponents.
Un cop l'aplicació està desplegada en el servidor, per a no haver de pujar el codi a mà cada vegada que es fan uns canvis, es configura un servei d'integració contínua, el qual és Code Pipeline que ofereix Amazon. Aquest es configura de tal manera que se li assigna un repositori de GitHub, el qual és on hi ha el codi de l'aplicació. D'aquesta manera, cada vegada que es faci un canvi en el repositori, el servei detectarà aquests canvis i els aplicarà a l'entorn de producció automàticament.

5 Resultats

Un cop tenim l'aplicació amb la generació d'horaris funcionant, obtenim que els resultats són satisfactoris, tot i que sempre es poden aplicar millores en l'algoritme, ja que es poden tenir en compte moltes més variables per a generar diferents resultats, per exemple, canviant l'ordre en què es miren les restriccions per a cada objecte tindríem resultats diferents, per tant, es podria indicar a quins objectes se li vol aplicar una preferència, en quants ordres diferents es vol aplicar l'algoritme per a generar diferents resultats en una tongada, etc. Ara per ara, però, els resultats d'una sola execució són els satisfactoris i, per tant, sabem que la base per a crear futures implementacions i millores funciona correctament.

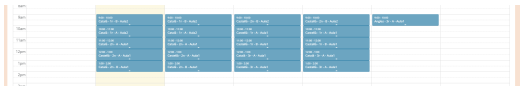

Fig. 7: Horari d'un professor

En la figura 7 es pot veure el resultat de com es mostra l'horari d'un professor, on es poden veure les assignatures que imparteix a cada hora (franja horària), els grups i les aules on s'imparteixen, per tant, es podria dir que un dels objectius del projecte és satisfactori, el qual és la del fet que els usuaris puguin generar els horaris automàticament de forma senzilla.
Respecte al temps d'execució, aquesta és la tasca que triga més per la gran quantitat de combinacions que hi ha a comprovar entre els objectes que conformen el centre, és per això, que com més gran sigui el centre i tingui més professors, més assignatures, grups i aules a comparar, el número de combinacions a comprovar serà més gran i, per tant, el temps d'execució de l'algoritme creixerà. No obstant això, es generen els horaris bastant ràpid per a totes les combinacions que s'han de fer i condicions que s'han d'avaluar, ja que amb l'execució de la figura 7, per a 7 professors, 10 assignatures, 9 aules i 8 grups, es generen tots els horaris amb un total de 75 franges horàries en 11 segons, per el que l'objectiu de generar els horaris per a tots els professors del centre docent de manera ràpida, es pot dir que també és satisfactori, tot i que segur que hi ha marge de millora i es poden generar d'una manera més ràpida.
Aquest temps d'execució, totes les combinacions i restriccions que s'estan mirant, es poden veure en el fitxer dels registres que es genera en el moment la generació dels horaris.

6 Conclusions

Al desenvolupar un projecte com aquest, el qual té moltes variables a tenir en compte per a aplicar les restriccions necessàries al generar els horaris, és important informar-se bé sobre l'entorn i el context que abarca el projecte en concret, ja que és fàcil que l'aplicació generi problemes en un futur, sobretot si no es planteja adequadament. A l'haver-hi tants objectes i tots relacionats entre si, les restriccions també van lligades entre si, per el que si és necessari modificar aquests objectes o les seves propietats, és fàcil que després hi haguin errors en la generació dels horaris, els quals són difícils de trobar per la quantitat de variables que hi ha.
Per aquesta raó és important fer un primer estudi sobre el tema, per a saber com funciona un centre docent en l'àmbit organitzatiu i d'aquesta manera saber que necessita l'aplicació i enfocar-la de manera que la tasca de manteniment sigui la més senzilla possible, per el que sigui fàcil de modificar, afegir o eliminar restriccions i que aquestes afectin el mínim possible a l'algoritme general per a la generació d'horaris i a altres funcionalitats del sistema.

Agraïments

En primer lloc, vull expressar el meu agraïment al tutor del treball de final de recerca Vladimir Bellavista per a la seva ajuda e implicació en el projecte, guiant-me en com fer cada una de les parts que abarca el treball en tot moment que he necessitat i facilitant-me tot el material que s'ha calgut en tot moment. També donar les gràcies per la confiança que m'ha donat des del primer dia.
També donar les gràcies a la universitat per a tots els recursos i facilitats que ha aportat en la realització del treball.
Per últim, donar les gràcies a la meva família i amics que m'han donat tot el suport moral durant tot el projecte, així com en tot el camí que he realitzat des de que vaig entrar a la universitat.

Referències

[1] ttps://www.penalara.com/es/ES

[2] ttp://www.vc.ehu.es/jiwotvim/IngenieriaSoftware/Teoria/BloqueII/U
3.pdf

[3] ttps://refactoring.guru/es/design-patterns

[4] ttps://www.oscarblancarteblog.com/2018/12/10/data-access-object-dao-pattern/

[5] ttps://fullcalendar.io/demos