UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escola Politècnica Superior de Gandia

Una ferramenta basada en optimització per l'assignació d'estudiants i tutors a pràctiques d'empresa

Treball Fi de Grau

Grau en Tecnologies Interactives

AUTOR/A: Fernàndez Fuster, David

Tutor/a: Sánchez Anguix, Víctor

Cotutor/a: Alberola Oltra, Juan Miguel

CURS ACADÈMIC: 2022/2023

Índex

Figures

Resum

L'assignació d'estudiants i professors a pràctiques d'empresa és un problema al qual els centres de formació professional s'enfronten cada any. Dades personals sobre estudiants, professors i empreses són emmagatzemats en fulls de càlcul que aleshores són emprats com a referència per un coordinador quan es fan les assignacions de forma manual. Aquest és un procés llarg i repetitiu i, per tant, automatitzable. Amb tal de facilitar el treball del coordinador, una interfície web és creada que extrau la informació necessària per a fer les assignacions. A més a més, a través d'un model d'optimització matemàtica implementat amb el software d'optimització OR-Tools, les diferents possibilitats d'assignació són explorades fins que la més òptima siga trobada: la de menors distàncies entre les cases dels estudiants i les empreses. Aquesta interfície també li permet al coordinador rectificar de forma manual qualsevol de les assignacions, tenint així en compte les decisions de naturalesa subjectiva. D'un total de 60 estudiants s'aconsegueix que solament 6 hagen de viatjar més de 10 km per a fer la pràctica. Aquesta solució al problema d'assignació pot ser fàcilment adaptada a qualsevol centre, el que demostra com els models d'optimització matemàtica són un gran aliat per als centres de formació professional i milloren part de l'experiència dels estudiants en les practiques.

Resumen

La asignación de estudiantes y profesores a prácticas de empresa es un problema al que los centros de formación profesional se enfrentan cada año. Datos personales sobre estudiantes, profesores y empresas son almacenados en hojas de cálculo que entonces son usadas como referencia por un coordinador cuando hace las asignaciones de forma manual. Este es un proceso largo y repetitivo y, por lo tanto, automatizable. Con tal de facilitar el trabajo del coordinador, una interfaz web es creada que extrae de los archivos la información necesaria para hacer las asignaciones. Además, a través de un modelo de optimización matemática implementado con el software de optimización OR-Tools, las diferentes posibilidades de asignación son exploradas hasta que la más óptima es encontrada: la de menores distancias entre las casas de los estudiantes y las empresas. Esta interfaz también le permite al coordinador rectificar de forma manual cualquiera de las asignaciones, tomando así en cuenta decisiones de naturaleza subjetiva. De un total de 60 estudiantes se logra que solamente 6 tengan que viajar más de 10 km para hacer la práctica. Esta solución al problema de la asignación puede ser fácilmente adaptada a cualquier centro, lo que demuestra como los modelos de optimización matemática son un gran aliado para los centros de formación profesional y mejoran parte de la experiencia de los estudiantes en las prácticas.

Abstract

The assignment of students and teachers to internships is a problem that vocational training centres face every year. Personal data about students, teachers and companies are stored on spreadsheet files that are then taken as a reference by a coordinator when making assignments manually. This is a long and repetitive, and therefore automatable, process. In order to facilitate the work of the coordinator, a web interface is created that extracts from the files the information necessary to make the assignments. Through a mathematical optimization model implemented with the optimization software OR-Tools, the different assignment possibilities are explored until the most optimal one is found: the one with the smallest distances between the students houses and the companies. This interface also allows the coordinator to manually rectify any of the assignments, thus taking into account decisions of a more subjective nature. From a total of 60 students, it is achieved that only 6 have to travel more than 10 km to do the internship. This solution to the assignment problem can be easily adapted to any centre, which shows how mathematical optimization models can become a great ally for vocational training centres and improve part of the experience of students in internships.

1. Introducció

Un dels moments més importants d'un centre de formació professional és quan els alumnes són assignats a una empresa per a posar en pràctica tot el aprés al llarg del cicle. Per a avaluar l'exercici de l'alumne en la pràctica i vigilar que l'empresa respecte el pactat en el conveni, també s'assigna un professor amb els coneixements necessaris.

Com un centre de formació professional sol estar compost per més d'un cicle formatiu (e.g., en informàtica tenim Desenvolupament d'Aplicacions Web o DAW, Desenvolupament d'Aplicacions Mòbils o DAM, Administració de Sistemes Informàtics i Xarxes o ASIX, etc.) i l'assignació té lloc per a cadascun al mateix temps, la figura d'un coordinador de pràctiques és necessària. Aquest s'encarrega de prendre les decisions d'assignació d'acord amb informació de les tres parts, que en el cas de l'alumne seria la ciutat de residència, que determina la distància a recórrer per l'alumne per a aplegar al lloc de pràctiques, i les seues preferències, que determinen quin tipus de pràctica sol ser més del seu gust.

A mesura que nous cicles entren en l'oferta educativa del centre i noves empreses es mostren dispostes a acceptar alumnes, aquesta tasca es torna cada volta més i més gran. La complexitat de la tasca des del punt de vista del coordinador és gran i més quan es busca l'assignació òptima d'acord a criteris racionals.

Aquesta tasca d'assignació òptima pot afrontar-se amb l'optimització matemàtica que, d'acord amb Ramos, A. et al. (2010), consisteix en l'aplicació del mètode científic als problemes de direcció i gestió. Aquesta s'aconsegueix amb la construcció d'un model matemàtic del sistema a partir del qual es poden predir i comparar els resultats de diverses estratègies i decisions, incorporant mesures de l'atzar i del risc. En el cas actual, l'optimització matemàtica ajudaria al coordinador de pràctiques a prendre millors decisions en l'assignació d'alumnes i professors a pràctiques d'empresa, sempre i quan el model emprat reflectisca bé els objectius a perseguir (e.g., minimitzar desplaçaments o maximitzar satisfacció) i les limitacions del escenari (e.g., nombre de pràctiques màxim al que l'alumne i professor es puga assignar).

Tenint en compte la naturalesa repetitiva d'aquesta tasca, es procedeix a automatitzar-la, dissenyant per a això un model d'optimització matemàtica, concretament un model de programació entera lineal, que represente aquest problema i implementant-ho en forma d'algoritme amb la llibreria OR-Tools. Seguidament s'implementarà una aplicació web que permeta al coordinador de pràctiques fer l'assignació en un clic, amb possibilitat de rectificar de forma manual la decisió presa per a donar cabuda a decisions subjectives.

2. Objectius

Segons el que s'ha comentat anteriorment, tenim com a objectiu principal desenvolupar una aplicació que permeta trobar l'assignació òptima d'estudiants i professors a pràctiques en en funció de la distància recorreguda per els primers.

Amb tal de complir aquests objectius s'ha de:

1. Desglossar el problema per a identificar-ne els elements que ens permeten modelar l'escenari.
2. Dissenyar el model d'optimització matemàtica.
3. Dissenyar la base de dades e interfície web que permetran al coordinador realitzar l'assignació de forma intuïtiva.
4. Dissenyar l'algoritme d'optimització.
5. Implementar i validar l'aplicació seguint una metodologia i ferramentes ben conegudes.

3. Marc Teòric

Previ al procés de disseny i desenvolupament del model d'optimització i de l'aplicació que farà ús d'aquest, cal repassar-ne els conceptes més importants, fer una xicoteta investigació sobre anteriors implementacions d'aquest tipus de models en l'àmbit educatiu, així com fer una comparació entre les diverses tecnologies disponibles que podrien fer d'això possible. Per tant, en l'apartat 2.1 s'explicarà en què consisteix l'optimització matemàtica, en el 2.2 es descriurà els seus components principals, en el 2.3 es detallaran les principals qualitats del tipus de problema que volem resoldre i com repercuteixen en el model, en el 2.4 es farà una comparativa entre la proposta actual i antigues propostes de models d'assignació en l'àmbit educatiu, en el 2.5 es descriurà la llibreria de programació més important per la implementació dels models i, finalment, en el 2.6 es recorreran les diverses tecnologies en què es podria crear l'aplicació.

3.1. Què és l'Optimització Matemàtica?

L'optimització matemàtica consta d'una sèrie de tècniques quantitatives per a prendre i preparar decisions, determinant la forma més eficient de fer-ne les coses (Eiselt & Sandblom, 2012).

Per exemple, si es vol preparar un viatge baix cost per a les vacances, atés a un pressupost i durada determinada, s'ha de decidir les activitats que ofereixen la major satisfacció a baix preu possible. Això es podria de fer de forma manual, amb els equivocs (potser es passe per alt una activitat que done igual satisfacció que altra, però que siga més barata) i els biaixos (s'incrementa el pressupost a última hora per a donar cabuda a activitats més cares) propis de l'ésser humà, però amb l'optimització matemàtica es disposa d'un algoritme que buscarà la millor solució possible i de forma objectiva.

És precisament per aquesta eficiència, que l'optimització matemàtica és prou comuna en entorns de treball on es prenguen decisions crítiques, com puguen ser aeroports, on és planifiquen els vols de forma que els clients sàpien quan han d'esperar i els avions no es queden sense combustible, i hospitals, on el personal s'ha d'assignar a distintes franges horàries per a assegurar que els pacients puguen ser atesos sense cap problema.

Per tal que l'eficiència en la recerca de solucions siga la millor possible, cal analitzar l'escenari i el problema que se'ns planteja i representar els seus aspectes principals com a formules, igualtats i desigualtats, en el que és coneix com un model d'optimització.

3.2. Components d'un Model d'Optimització Matemàtica

El model que construïm per a un determinat escenari és una imatge propera de la realitat, que inclou components observables (Eiselt & Sandblom, 2012). Aquestos components són:

- **Funcions Objectiu**: són la forma en la que es mesura la qualitat amb la qual es resol un problema, i es representen com funcions que busquen maximitzar o minimitzar un valor jugant amb la resta de components del model. En funció del problema, pot haver-hi una sola funció objectiu o múltiples. En el cas del problema del viatge baix cost esmentat en l'anterior apartat, seria multiobjectiu: per una part es busca minimitzar els gastos del viatge, però per una altra part també es busca maximitzar la satisfacció per les activitats escollides, amb la complexitat addicional de que les solucions siguen Pareto òptimes.
- **Restriccions:** per a que la solució d'un objectiu puga considerar-se valida, ha de complir amb una sèrie de requisits, que en el model es coneixen com a restriccions. En el cas de l'exemple, les restriccions serien la quantitat de diners que es permet gastar (per exemple, <=3000 €) i la quantitat d'activitats que podem triar (entre 10 i 30 activitats): qualsevol decisió presa no ha de superar aquests gastos i ha de tindre un rang d'activitats raonable.
- **Paràmetres**: són tota la informació coneguda o deduïble però no modificable, a l'estar ocasionada per factors externs sobre els quals no tenim control total, com ho aplega a ser el pressupost (3000 €) en el nostre viatge i la nostra capacitat (30 activitats). A pesar que en la vida real el pressupost es pot expandir amb préstecs i que el temps de vacances (i, per tant, la capacitat) es puga reduir, aquests són canvis per factors externs que en el model no es poden veure reflectits. El model reflecteix el problema que es va analitzar en el seu moment.
- **Variables:** són l'antítesi dels paràmetres, és a dir, tota informació desconeguda però modificable (ja que es té capacitat de decisió sobre aquesta), com puga ser les activitats a escollir, cadascuna amb la seua satisfacció i gastos. En altres paraules, són els valors que es poden anar ajustant per a aconseguir els objectius de tindre unes vacances de baixos costos però alta satisfacció.

Per tant, es té que un model d'optimització matemàtica és el reflex d'un entorn en el moment de ser analitzat. En aquesta anàlisi s'extrau quin és el problema que volem solucionar i quins objectius poden portar a aquesta solució, així com les restriccions que s'imposen a l'hora d'aconseguir-ho i els paràmetres que formen part d'aquestes. Finalment, s'identifiquen les variables que tenen un impacte sobre la solució i que es poden anar ajustant fins a trobar la més òptima.

Generalment, els problemes a resoldre solen seguir uns patrons que permeten classificar-los en distints tipus, i el mateix passa amb els models que es dissenyen per a resoldre'ls. En aquest cas, d'assignació o emparellament.

3.3. Models d'Assignació/Emparellament

D'acord amb Eiselt & Sandblom (2012), els models d'assignació o emparellament s'empren per a resoldre problemes de programació lineal d'enters centrats en l'assignació de recursos a tasques. Aquests problemes es caracteritzen per la participació de dos conjunts d'elements, que s'han d'interrelacionar en un procés anomenat assignació, ja siga de forma exacta en el cas que siga requisit que tots els elements tinguen una relació o inexacta en el cas contrari. Per exemple, l'assignació de pacients a metges seria un cas d'assignació inexacta, ja que no cal que tots els metges tinguen assignat un pacient, però sí que tots els pacients tinguen assignat un metge.

A més a més, aquestes relacions també es poden classificar en funció de la seua cardinalitat:

- **1 a 1:** s'interrelacionen un element de cada conjunt.
- **1 a molts:** un element d'un conjunt pot estar relacionat amb molts altres de l'altre conjunt.
- **Molts a molts:** molts elements d'un conjunt poden estar relacionats amb molts de l'altre conjunt.

En la seua versió més senzilla, aquests problemes també compleixen amb les següents propietats:

- **Deterministes:** els paràmetres es coneixen amb certesa. En el cas dels models d'assignació, un exemple típic de paràmetre seria la quantitat màxima de tasques a la que un recurs pot ser assignat (e.g., la seua capacitat).
- **Linearitat:** totes les expressions, funcions objectiu i restriccions són lineals, per el que qualsevol canvi que es faça en aquestes tindrà unes conseqüències predictibles en la solució final.

Un exemple de model d'assignació seria l'assignació de pacients a metges en un hospital. Suposant que un hospital crea un servici a partir del qual els pacients poden demanar cita especificant símptomes i rang de disponibilitat:

- **Funció Objectiu:** aquesta seria la maximització de la satisfacció dels pacients atesos. Es buscaria assignar els pacients a un metge que sàpia identificar els seus problemes i donar-li solució.
- **Restriccions:** aquestes serien la disponibilitat del pacient i la quantitat de pacients que pot atendre un metge.
- **Paràmetres:** la capacitat i l'especialitat del metge, així com els símptomes del pacient.
- **Variables:** la satisfacció del pacient de ser atés per tal metge.

La versatilitat d'aquests tipus de problemes ajuda a fer que el seu ús es puga estendre a distints entorns de treball. En el cas de l'àmbit educatiu hi ha prou casos d'aplicació amb resultats prou positius.

3.4. Models d'Emparellament en Educació

Els problemes d'assignació són molt freqüents en l'àmbit acadèmic, independentment del país i del tipus de centre, tant que es poden trobar publicacions en revistes científiques on es descriuen models d'emparellament aplicats en entorns reals.

Per exemple, Sanchez-Anguix et al. (2019) proposen un algoritme genètic multiobjectiu i proper a Pareto per a l'assignació d'estudiants a supervisors per a treballs d'investigació. Aquest té en compte les preferències dels estudiants i supervisors per a un determinat tòpic d'investigació, així com la càrrega de treball dels últims, el que equival a l'actual cas on els alumnes tenen les seues preferències i els professors tenen una capacitat, però els professors no tenen preferències, sinó perfils. En contrast, fa ús d'un algoritme genètic Pareto, amb els conseqüents avantatges de proposar múltiples solucions properes a la Pareto optimalitat per a les assignacions, i la similitud entre les preferències dels alumnes i dels professors es calcula a través de la similitud de les paraules claus emprades i la seua posició en una llista.

Per altra banda, en el cas de Sanchez-Anguix l'assignació és solament entre dues parts (supervisors i alumnes) i no permet la incorporació d'altres parts (com puguen ser les empreses de l'actual proposta). Pel que fa als professors, en l'actual proposta no hi ha preferències, sinó perfils que corresponen als distints cicles formatius de la seua institució, pel que un alumne sempre tindrà assignat un professor amb els coneixements suficients per a avaluar-lo. En el cas de Sanchez-Anguix, els supervisors tenen les seues pròpies preferències, que disten molt de l'alumnat, dificultant l'assignació òptima.

Alberola et al. (2016) proposen una ferramenta per a la formació d'equips en entorns educatius i de treball, combinant tècniques d'intel·ligència artificial com generació d'estructura col·lativa, aprenentatge Bayesià i la teoria de rols de Belbin. En aquest cas es té en compte el feedback d'altres companys d'equip amb tal d'establir el rol més predominat d'un estudiant. Igual que en l'actual proposta es programa una ferramenta que posteriorment s'integra en la plataforma d'una institució, que permet als professors la creació automàtica d'equips. A més a més, les dades dels alumnes són emmagatzemades en una base de dades i està prou modularitzat perquè es puga adaptar a plataformes d'altres institucions. A diferència de l'actual proposta, es retroalimenta amb l'opinió dels companys d'equip d'un determinat alumne i no hi ha involucració de parts addicionals, com puguen ser els professors i les empreses de l'actual proposta.

Per altra banda, a l'estar basat completament en el caràcter de l'estudiant, pot donar lloc a inconsistències en cas de que l'alumne evolucione madurativament entre les iteracions, encara que les probabilitats de que això ocórrega son prou baixes.

Harper et al. (2005) presenten un algoritme genètic com la solució a una tasca d'assignació d'estudiants a projectes, a partir de les preferències emeses pels estudiants. Igual que en el present problema d'assignació d'estudiants i professors a pràctiques d'empresa, s'enfronta amb el dilema de que fer quan més d'un estudiant té com a principal preferència un mateix projecte i ofereix la possibilitat de rectificar qualsevol assignació. En contrast, fa ús dels

avantatges que ofereix l'algoritme genètic de donar accés a la resta de possibles solucions per a possibles discussions.

Per altra banda, l'actual proposta està pensada per ser aplicada en el quadern de bitàcola de qualsevol institució educativa amb les dades d'alumnes, professors i empreses emmagatzemats en una base de dades, mentre que la de Harper s'empra a través d'una aplicació de Visual Basic que no està recolzada per cap base de dades.

Cechlárová et al. (2014) assignen professors en pràctiques a escoles, amb aquests professors especialitzant-se en dues assignatures. Amb tècniques de flux de xarxes s'aconsegueix assignar un professor a una escola per assignatura. Similar a l'actual proposta, en cas que algun alumne tinga pendent almenys una de les assignatures, no podrà fer la pràctica d'aquesta assignatura. Per contra, l'assignació és solament d'estudiant a dos professors (que representen les assignatures en les quals volen especialitzar-se), i cap de les dues parts tenen alguna preferència.

Per altra banda, no es té en compte les preferències de cap de les parts i el recorregut més òptim entre dues escoles per part d'un alumne no és Pareto.

Per últim, Thiruvady et al. (2021) fan ús tant d'un model de programació per enters (per a casos amb menys de 3000 estudiants) com una heurística d'optimització de colònia de formigues (per a casos amb més de 3000 estudiants) amb el fi d'automatitzar l'assignació d'estudiants universitaris a llocs de treball, maximitzant compromís de l'estudiant i satisfacció del negoci, i afavorint la diversitat en la mateixa empresa. Igual que en el present problema fa ús d'un model de programació per enters i assigna alumnes a pràctiques d'empresa. Per contra, te en compte la necessitat de les empreses que la seua plantilla de treballadors tinga un nombre paregut d'homes i dones, així com l'interés en agafar a un alumne d'acord amb els resultats de la seua entrevista.

Per altra banda, no té en compte els professors en l'assignació, a pesar d'avaluar junt amb l'empresa la pràctica de l'alumne.

Una característica important que comparteixen Harper et al. (2005) i Cechlárová et al. (2014) és l'ús del solver CPLEX (una llibreria matemàtica propietària) per a obtindre les solucions al problema que plantegen. La tria d'un solver és un pas important en la resolució d'un problema d'optimització matemàtica, així com l'API que gastarem per a fer-ne ús d'ell.

En el cas actual, s'emprarà OR-Tools, una API lliure i que es pot accedir a partir de llenguatges de programació tradicional, per a la seua fàcil integració en un entorn web.

3.5. OR-Tools

OR-Tools és una llibreria de codi obert de Google emprada per a l'optimització en general, que busca la millor solució possible a partir d'una gran quantitat de possibles solucions, fent ús d'algoritmes estandarditzats. Entre aquests s'inclouen:

- **Programació de Restriccions:** per a trobar solucions factibles a un problema expressat en restriccions, com puga ser un problema de calendari (assignació d'esdeveniments a un temps determinat) i problemes de seqüenciació (assignació de tasques a màquines). Addicionalment, pot servir d'alternativa per a problemes d'assignació que tinguen una convergència lenta.
- **Programació Lineal i de Nombres Enters Mixtos:** troba el valor òptim d'una funció objectiu lineal, donat un conjunt de desigualtats lineals com a restricció. S'empra per a resoldre problemes lineals com, per exemples, els de assignació.
- **Ruta de Vehicles:** per a identificar les millors rutes de vehicles donades unes restriccions.
- **Algoritmes de Grafs:** per a trobar els camins més curts en grafs, fluxos de mínim costos, fluxos màxims i assignacions de suma lineal.

Cada un d'aquests algoritmes fa ús d'una llibreria matemàtica (solver), que pot ser gratuïta (però amb una llicència a respectar que limita la seua distribució) o privada. Per exemple, SCIP és un solver amb llicència Zlib que s'empra per a resoldre problemes de programació de restriccions, de programació (no) lineal i de nombres enters, mentre que GLPK sols serveix per a programació lineal i d'enters però disposa de llicència GLPv3.

El llenguatge natiu d'OR-Tools és C++, però també suporta Python, Java i C#, llenguatges al que es tracta de traduir el model previ a la cerca de solucions seguint una sèrie de passos:

1. **Importació de dades (tant internes com externes) a partir de les quals s'extrauen els paràmetres.**
2. **Declaració de variables.**
3. **Declaració de restriccions.**
4. **Declaració de la funció objectiu.**

Alternativament a OR-Tools, existeixen llenguatges d'alt nivell quasi estandarditzats com AMPL i GAMS, altres APIs genèriques per a diversos solvers com Pyomo o una API específica per al solver d'IBM CPLEX, però per familiaritat s'ha optat finalment per OR-Tools.

3.6. Tecnologies per a la Ferramenta Web

A part d'emprar OR-Tools per a obtindre les assignacions més òptimes, també s'ha de tindre en compte altres aspectes referents a l'aplicació web que donarà suport als usuaris. Així, caldrà definir aspectes com en quina base de dades s'emmagatzemarà la informació que aquest requereix, complint així el primer pas d'importar dades a paràmetres esmentat en l'anterior apartat, que és necessari per a traduir el model a un llenguatge comprensible per OR-Tools, així com un marc de treball que facilite el manteniment i actualització de l'aplicació.

3.6.1. Bases de Dades

En el cas de la base de dades, cal tindre en compte que, per norma general, consisteix en una sèrie d'elements interrelacionats que són presentats a l'usuari d'una forma abstracta, és a dir, sense tindre coneixement de com estava emmagatzemada la informació que se li presenta (Rivera F., 2018). Per tant, és important determinar com aquestes dades estaran relacionades, ja que determinarà l'eficiència d'emmagatzemament i la facilitat en l'accés i administració d'aquesta que pot variar també en funció de la llibreria i llenguatge de programació emprats. Per a establir les relacions, disposem de diverses aproximacions possibles:

- **Base de Dades Relacional:** les dades estan emmagatzemades d'una forma estructurada que evita la redundància i que fa de més visible la relació.

  Un concepte, per exemple, cine, pot estar compost per diversos elements importants, com pel·lícula, director i companyia, que tenen relació entre si, però amb cadascú tenint la seua pròpia informació. Per aquest mateix motiu, cadascun d'aquests elements s'emmagatzema en una relació pròpia amb un identificador que relaciona un dels seus registres amb una entrada d'una de les altres taules. Amb una consulta complexa, es podria accedir a la informació completa d'una determinada pel·lícula, incloent-hi tot el relacionat amb el director i la companyia.

  Gràcies a aquesta estructura basada en taules i relacionades amb un identificador, s'evita repetir-ne les dades de la companyia i de l'autor per cada pel·lícula en la qual hagen participat. De cara al desenvolupador, això pot facilitar el manteniment i correcció d'errates en la base de dades, sobretot en els casos on la quantitat de registres siga enorme (fent-la, per tant, més robusta), però pot suposar una major complexitat en el disseny i operacions en la base de dades.

  Pel que fa a programes d'aquest tipus i destaquen MySQL i Access, que són propietaris, encara que el primer cas te l'avantatge que els servicis d'allotjament web solen vindre amb ells. Entre les alternatives gratuïtes destaquen SQLAlchemy, MariaDB i SQLite.

- **Base de Dades No Relacional:** d'acord amb Strauch, C. et al (2011), aquest tipus de base de dades naix de la necessitat de trobar-ne formes més eficients i barates de gestionar dades. Això s'aconsegueix evadint complexitats que poden resultar innecessàries en certs casos d'ús com, per exemple, l'emmagatzematge de dades de sessió; fent ús d'un mapatge objecte-relació simple i oferint alt rendiment,

escalabilitat horitzontal i execució en maquinari modest. Es classifiquen en funció de la seua forma d'emmagatzemar valors en:

- **Clau-Valor:** estan basats en mapes/diccionaris, permetent la demanda i inserció de valors en funció de la clau, afavorint escalabilitat sobre consistència. Entre exemples d'aquest tipus trobem Amazon Dynamo, Project Voldermort i Redis.
- **Basats en Documents:** que admeten dades més complexes i significatives al permetre encapsular parells claus-valor en el document, com per exemple Apache Couch DB i Mongo DB.
- **Orientats a Columna:** emprats per a l'anàlisi de dades en arquitectures paral·leles amb tal d'obtindre un major rendiment. Es caracteritzen principalment per ser escasses, distribuïdes i d'ordenació persistent multidimensional. D'aquest tipus destaquen Google BigTable, HBase i Cassandra.
- **Orientats a Grafs:** amb els nodes representant informació i les arestes la relació entre aquests, com HyperGraphDB.

Seguint l'exemple del cine, en una estructura basada en documents tindríem una col·lecció de pel·lícules (cadascuna sent un document), amb tota la informació referent al director i la companyia sent un document incrustat al document principal (l'equivalent a fer un LEFT JOIN en una base de dades relacional).

Per contra, la redundància que comporta pot complicar el seu manteniment i la correcció d'errates, especialment si disposem d'una col·lecció molt gran. En una base de dades relacional tant l'autor com la companyia solament tenen un registre, pel que si hi ha un problema en aquest registre, serà més fàcil d'identificar. En una basada en documents, caldria comprovar en cada registre de pel·lícula si s'ha produït una errata en l'autor o en la companyia. Una possible solució per a mitigar aquest problema seria tindre una col·lecció de directors en lloc de pel·lícules (amb cada director tenint un llistat de pel·lícules), ja que, en general, una pel·lícula sol tindre un únic director, per la qual cosa les dades dels dos elements quedaran més aïllades si s'organitzen d'aquesta forma, sent així més fàcils de mantindre, encara que es continua tenint el mateix problema en les companyies.

3.6.2. Marcs de Treball

Els marcs de treball són entorns de treball construïts d'acord amb patrons de disseny, incentivant les bones pràctiques i la creació d'aplicacions escalables i optimitzades. Solen disposar d'un conjunt de llibreries que agilitzen el desenvolupament i manteniment, així com uns estàndards en la nomenclatura i en l'estructura de directoris que faciliten l'enteniment del codi font per part de qualsevol persona que tinga familiaritat amb el marc de treball emprat. A pesar que existisca l'opció de programar des de zero, en el cas que ens ocupa, la modularització i estandardització que facilita aquest tipus de tecnologia són de les característiques principals que deu tindre l'aplicació, ja que volem que es puga adaptar a qualsevol institució educativa.

Hi ha dos tipus de marcs de treball: *FrontEnd* si faciliten la part de client de l'aplicació i *BackEnd* si faciliten la part de servidor. Ambos tipus comparteixen els avantatges mencionats en l'anterior paràgraf però, per les característiques del seu entorn, se solen emprar per distints motius.

Els marcs de treball *FrontEnd* s'especialitzen en la part de presentació de la informació a l'usuari i en la seua interacció amb aquesta. Per tant, són útils a l'hora de desenvolupar un ràpid prototip de l'aplicació, a la vegada que la seua modularització permet el reciclatge de les funcionalitats *JavaScript*.

Respecte als marcs de treball BackEnd, s'especialitzen en la part lògica de l'aplicació, que involucra l'emmagatzematge i recuperació de dades per part de l'usuari, així com la seguretat de l'aplicació. Per tant, són útils per a la creació d'APIs que poden ser emprades des de qualsevol client, independentment de la tecnologia emprada per aquest, així com per a facilitar la interacció amb la base de dades i el xifrat de la informació.

Alternativament a l'API, aquests marcs de treball solen comptar amb motors de plantilles que faciliten la presentació de la informació però de les que se sol recomanar en cas que la part de client de l'aplicació siga simple.

En cas de que aquesta siga complexa, es pot combinar amb un marc de treball *FrontEnd* que cridarà a l'API del marc de treball *BackEnd*.

3.6.2.1. Marcs de Treball FrontEnd

Entre els possibles marcs de treball *FrontEnd* a emprar ens podem trobar amb:

- **Angular:** un marc del treball de Google que fa ús del llenguatge de programació de Microsoft, *TypeScript*. Està centrat en el desenvolupament web *FrontEnd* (amb llibreries que permeten exportar l'aplicació a altres plataformes, com puga ser mòbil o escriptori), pel que destaca per les seues llibreries de caràcter més visual, com *PrimeNG* i *MaterialDesign*. També està enfocat en la creació d'aplicacions modulars, amb una estructura de directoris basada en components, servicis i rutes, permetent una forta escalabilitat y fàcil manteniment i actualització d'aquestes, i una càrrega

ràpida de les pàgines al descarregar-ne solament els recursos necessaris per a mostrar la pàgina a la qual s'ha accedit.

El principal desavantatge és la gran diferència entre la versió 2 i posteriors d'Angular, que obliga a vigilar les dates de publicació de les entrades sobre Angular que hi ha en la web, així com l'enorme quantitat de llibreries publicades per a aquest, amb bona part d'elles no sent actualment funcionals i amb moltes opcions per a resoldre un problema en particular, sent indispensable un parell d'anys d'experiència per a tindre un catàleg localitzat de llibreries que complisquen totes les necessitats.

- **Vue.js:** un marc de treball *FrontEnd* emprat per a la creació d'interfícies d'usuari, que facilita la modificació de la sintaxi HTML d'acord amb l'estat actual del codi *JavasScript*. A causa de la seua flexibilitat, es pot emprar per a distints tipus d'aplicació, independentment de la seua complexitat, com puguen ser aplicacions mòbils i d'escriptori, aplicacions estàtiques, aplicacions d'una sola pàgina o com a components web que s'adhereixen a qualsevol pàgina.

  Vue es promociona com el "marc de treball progressiu", en el sentit que, el que comença com una simple llibreria, va fent-se més i més gran a mesura que es requerisca les seues funcionalitats, fins a convertir-se en un marc de treball complet. En altres paraules, és un marc de treball que pot créixer i adaptar-se a les necessitats del programador.

  Igual que amb Angular, es basa en components, però aquests components són d'un sol arxiu, és a dir, mentre que en Angular un component està compost d'un fitxer *HTML*, un altre *CSS*, un tercer *TypeScript* i un últim opcional per a fer proves, en Vue.js un component solament està compost d'un arxiu amb el codi *JavaScript* estant envolt per una etiqueta script, les etiquetes HTML dins d'una etiqueta *template* i el CSS en una anomenada *style*.
- **React:** és un marc de treball FrontEnd emprat per a la creació d'interfícies d'usuari. Destaca sobretot per no generar cap dependència de l'aplicació amb la seua tecnologia, amb la seua presencia podent variar des d'un simple widget en l'aplicació a alimentar completament aquesta.

  Igual que amb Vue.js, es basa en components que contenen tots els elements (plantilles, estils i codi JavaScript), però aquesta vegada amb l'ajuda d'una extensió de JavaScript anomenada JSX que fa de més visible la interacció del codi JavaScript sobre la plantilla.

3.6.2.2. Marcs de Treball BackEnd

Entre els possibles marcs de treball *BackEnd* a emprar ens podem trobar amb:

- **Flask:** un micromarc de treball *BackEnd* basat en *Python*, que també es pot emprar a nivell de *FrontEnd* fent us de *Jinja2*, un motor de plantilles HTML que permet el filtrat i mostra dinàmica de les dades.

  El terme "micromarc de treball", que empra *Flask* en la seua promoció, es refereix al fet que la seua instal·lació és nueta (incloent-hi el bàsic per a fer peticions HTTP i el motor de plantilles *Jinja2*, què es pot canviar fàcilment per un altre si es veu convenient), però amb la possibilitat d'estendre a base de llibreries (conegudes ací com a extensions), que faciliten la integració d'altres tecnologies com *Redis*, *SQLAlchemy* i *MongoDB*.

  Pel que fa a estandarditzacions, busca unes carpetes anomenades *static* i *templates* per a ubicar els recursos *FrontEnd* de l'aplicació, amb la possibilitat d'incloure el patró Fàbrica i els *Blueprints* de voler-ne fer una aplicació modular.

  Destaca per la seua simplicitat, que facilita l'adaptació de nous desenvolupadors al codi; la seua flexibilitat, amb la possibilitat de poder modificar les extensions d'acord amb els requeriments de l'aplicació; alt rendiment a causa de les poques capes d'abstracció entre l'usuari, la base de dades, la cau i les peticions; i la seua naturalesa modular, que simplifica el complet procés de desenvolupament i testabilitat, així com permet crear múltiples instàncies de servidors i aplicacions Flask que són distribuïdes a través d'una xarxa extensiva de servidors amb propòsits específics.

  Per contra, a l'emprar *Python* de llenguatge de programació, l'execució de peticions més complexes pot aplegar a ser lenta i restringeix els servicis d'allotjament web a aquells que disposen d'un sistema operatiu amb accés per terminal, amb tal d'executar el fitxer principal. Aquests tipus d'allotjament, resulten més cars que els basats en panels de control com *Plesk*.

  Addicionalment, no està pensat per a projectes grans i el suport a la comunitat és més escàs en comparació amb *Django*.

- **Django:** un marc de treball prou potent basat en *Python*, que destaca per la seua versatilitat (al poder ser emprat per al desenvolupament de qualsevol mena d'aplicació) i pel seu èmfasis en la seguretat.

  Django s'estructura en capes: Models (o capa d'abstracció) per a estructurar i manipular dades en l'aplicació; Vistes, per a encapsular la lògica responsable de processar les peticions de l'usuari i retornar la resposta i Plantilles, que renderitzen la informació presentada a l'usuari. També empra la filosofia DRY (no et repetisques) per a fomentar la reutilització del codi i evitar la creació de blocs iguals

  De nou, a l'estar basat en Python, compta amb els mateixos desavantatges que Flask, amb l'afegit de la seua naturalesa monolítica que porta a un menor control del flux de

treball i de la seua complexitat., tant per codi necessari per a programar l'aplicació (per l'escassetat de dependències) com per a consum d'amplada de banda i temps de còmput. Per últim, la carència de convencions dificulta l'adaptació dels nous desenvolupadors al codi i pot portar a un alt contrast entre blocs de codi.

- **Laravel:** és un marc de treball *BackEnd* de codi obert que fa ús de *PHP* i que s'empra per al desenvolupament d'aplicacions web que segueixen l'arquitectura model-vista-controlador (MVC).

  Es caracteritza pel seu sistema de paquets modular amb gestor de dependències i per oferir moltes formes per a accedir a la base de dades. També facilita la programació de funcionalitats com autenticació i l'enviament de correus, el testege de l'aplicació i el control d'errors a través de *logs*.

  També ofereix suport per a Redis i altres tecnologies d'emmagatzematge en cau i facilita la configuració de múltiples configuracions de cau.

3.6.2.3. Elecció del Marc de Treball

Degut a que la major part de la complexitat de l'aplicació es troba en la part *BackEnd*, amb la part *FrontEnd* constant solament de formularis i taules, s'opta per un marc de treball *BackEnd*. Aquest marc de treball deurà contar amb un motor de plantilles que facilite la presentació de les dades en la part *FrontEnd*.

De entre els marcs de treball *BackEnd* contemplats en aquesta secció, *Django* i *Flask* semblen bones opcions al emprar *Python* com a llenguatge de programació, que és el llenguatge que s'utilitzarà per a la programació de l'algoritme d'optimització amb OR-Tools.

Com és busca que l'aplicació siga emprada per més d'una institució acadèmica, deuria de ser un marc de treball que no requerisca d'una corba d'aprenentatge molt gran i que tinga un caràcter modular per a fer de simple l'adaptació del codi a les necessitats de dites institucions. El rendiment també és important, ja que l'execució de l'algoritme d'optimització té un cost computacional prou gran i convé que l'aplicació siga fluida encara que l'estiga executant.

La testabilitat també aplega a ser important en aquest cas, ja que el procés d'assignació és molt important per a l'aplicació i convé tindre unes proves preparades que ens permeten saber si els resultats son fiables.

Totes aquestes qualitats les apleguem a trobar en Flask, amb Django sent una especie d'antítesis per a ell.

3. Model Matemàtic

En aquesta secció es procedeix a fer una modelització del problema des d'una perspectiva matemàtica. Es comença plantejant de forma matemàtica el problema (3.1.) i, finalment, es defineix el model (3.2.).

3.1. Plantejament del Problema

En un centre d'FP amb diferents cicles formatius, hi ha un conjunt de professors *j={1...m}* que s'encarregaran de tutoritzar les pràctiques *i={1...n}* d'un conjunt d'alumnes *k={1...u}* amb les seues respectives empreses *l={1...v}*.

Així mateix, cada combinació d'alumne *k* i pràctica *i* disposa d'una distància d_{ki}, que és la quantitat de km que l'alumne *k* ha de recórrer des del seu lloc de residencia fins al lloc de la pràctica i per a fer la pràctica.

Es busca, per tant, l'assignació òptima d'alumnes i professors a practiques que minimitze el desplaçament dels primers per a aplegar al lloc de pràctiques.

Amb tal d'aconseguir dit objectiu s'ha de tindre en compte els següents factors:

- Un alumne *k* solament pot estar assignat a una pràctica *i*.
- Un professor *j* solament pot estar assignat a un nombre de pràctiques que no el sobrecarregue d'un limit màxim de treball.
- Una empresa *l* ha de tindre una quantitat de pràctiques assignades que siga com a mínim l'establert pel conveni *i* com a màxim les que hi haja posat en oferta.
- Una pràctica ha de tindre assignats tant un alumne com un professor que el tutoritze.

Atenent això es desenvolupa el següent model.

3.2. Definició del Model Matemàtic

Tal com s'ha explicat anteriorment, l'objectiu del model és minimitzar la distància a recórrer d'un alumne *k* per a aplegar al lloc on s'imparteix la pràctica *i*, atenent a les limitacions de capacitat de tant alumne com professor. Formalment queda representat de la següent manera:

$$\min Z \quad = \sum_{k=1}^{u} \sum_{i=1}^{n} d_{ik} Y_{ik}$$

Restringit per:

$$[càrrega\ professor\ j]: \sum_{i=1}^{n} X_{ij} \leq s_j;\ j = 1 \dots m$$

$$[alumne\ k\ ha\ de\ ser\ assignat]: \sum_{i=1}^{n} Y_{ik} = 1;\ k = 1 \dots u$$

$$[nombre\ de\ pràctiques\ assignades\ a\ empresa\ l]: a_l \leq \sum_{k=1}^{u} \sum_{i=1}^{n} Y_{ik}\ \tau_{il} \leq b_l;\ l = 1 \dots v$$

$$[enllaç\ supervisió\ pràctica\ i]: \sum_{j=1}^{m} X_{ij} = \sum_{k=1}^{u} Y_{ik};\ i = 1 \dots n$$

$$[professors\ assignats\ a\ pràctica\ i]: \sum_{j=1}^{m} X_{ij} \leq 1;\ \ i = 1 \dots n$$

$$[alumnes\ assignats\ a\ pràctica\ i]: \sum_{k=1}^{u} Y_{ik} \leq 1;\ \ i = 1 \dots n$$

Amb les següents **variables de decisió**:

X_{ij}: 1 si el professor j està assignat a la pràctica i, 0 en cas contrari ; $j = \{1 \dots m\}$, $i = \{1 \dots n\}, X_{ij} \in \{0,1\}$

Y_{ik}: $1\ si\ l'alumne k\ està\ assignat\ a\ la\ pràctica\ i, 0\ en\ cas\ contrari;\ k = \{1 \dots u\}$, $i = \{1 \dots n\}, Y_{ij} \in \{0,1\}$

Amb els següents **paràmetres**:

τ_{il}: $Paràmetre\ binari\ que\ indica\ amb\ un\ 1\ si\ la\ pràctica\ i\ pertany\ a\ l'empresa\ l, 0\ en\ cas\ contrari;$ $i = \{1 \dots n\},\ l = \{1 \dots v\}, \tau_{il} \in \{0,1\}$

d_{ik}: $Distancia\ entre\ l'alumne\ k\ i\ la\ pràctica\ i;\ k = \{1 \dots u\},\ i = \{1 \dots n\},\ d_{ik} \geq 0, d_{ik} \in \mathbb{R}$

a_l: $Nombre\ mínim\ de\ pràctiques\ a\ cobrir$ de $l'empresa\ l; l = \{1 \dots v\}, a_l \geq 0, a_l \in \mathbb{Z}$

b_l : $Nombre\ màxim\ de\ pràctiques\ cobertes$ de $l'empresa\ l; l = \{1 \dots v\}, b_l \geq 0, b_l \in \mathbb{Z}$

s_j: $Capacitat\ màxima\ de\ supervisió\ del\ professor\ j; j = \{1 \dots m\}, s_j \geq 0, s_j \in \mathbb{Z}$

En la funció objectiu busquem minimitzar el resultat de la suma de totes les distàncies dels alumnes assignats.

En quant a les restriccions:

- [$càrrega\ professor\ j$]: La quantitat de pràctiques assignades al professor *j* no ha de superar la seua capacitat màxima de supervisió.
- [$alumne\ k\ ha\ de\ ser\ assignat$]: Un alumne *k* ha de ser assignat exactament a una pràctica.
- [$nombre\ de\ pràctiques\ assignades\ a\ empresa\ l$]: La quantitat de pràctiques que una empresa *l* pot tindre cobertes ha de trobar-se entre el establert en el conveni entre el centre educatiu i l'empresa. És a dir, entre un mínim i un màxim.
- [$enllaç\ supervisió\ pràctica\ i$]: El nombre de professors assignats a una pràctica *i* ha de ser el mateix que el nombre d'alumnes assignat a aquesta. És a dir, si assigne un alumne a la pràctica *i* he d'assignar un tutor i viceversa.
- [$professors\ assignats\ a\ pràctica\ i$]: Com a màxim, sols un professor pot estar assignat a la mateixa pràctica *i*.
- [$alumnes\ assignats\ a\ pràctica\ i$]: Com a màxim, sols un alumne pot estar assignat a la mateixa pràctica *i*.

3.3. Dades Emprades

Les dades sobre les quals treballa el model han sigut entregades en forma de fulls de càlcul per la institució interessada per aquesta proposta: 4 fulls de càlcul per als alumnes (1 per cada cicle, de 30 alumnes cadascú, el que fan un total de 120 alumnes), un per als professors (un total de 44) i un altre per a les empreses (unes 200).

En el cas dels alumnes, les dades dels fulls de càlcul van ser extretes a partir d'una enquesta contestada per ells en motiu de les pràctiques d'empresa. Aquests fulls contenen la següent informació:

- **Nom i Cognoms de l'Alumne:** per temes de protecció de dades s'anonimitza seguint el patró *alumneXX*. S'empra per a la creació automàtica del seu usuari i com a forma d'identificar-lo en l'assignació.
- **Ciutat de Residencia:** s'empra com a base per a calcular la distància a recórrer de l'alumne cap al seu lloc de pràctiques.
- **Disponibilitat de Cotxe:** pot ser "Sí" o "No". A l'hora de calcular la distància es té en compte si l'alumne ha d'anar en peu o en cotxe, ja que pot aplegar més de pressa si disposa d'un i, per tant, pot anar a pràctiques més llunyanes.

- **Preferències:** entre 5 i 10 camps en el que és pregunta per la preferència de l'alumne respecte a una de les habilitats treballades durant el cicle (e.g., BackEnd o Robótica). La preferència es mesura en un interval de 0 a 10, on 0 significa que l'alumne no sent cap afinitat per aquest tipus de pràctica i 10 significant que sent molta afinitat.

En el cas dels professors, contenia els següents camps:

- **Nom del Professor:** per temes de protecció de dades s'anonimitza seguint el patró *professorXX*. S'empra per a la creació automàtica del seu usuari i com a forma d'identificar-lo en l'assignació.
- **Perfil:** una sèrie de camps, cadascú sobre un cicle (e.g., DAW, Anglés), amb una "X" en cas d'haver-ne impartit classes i un buit en cas contrari. S'empra per a assignar el professor a una pràctica que quadre amb el seu perfil.
- **Hores Alliberades:** hores que el professor disposa lliures setmanalment. S'empra per a definir la capacitat del professor.

Per últim, en el cas de les empreses, es tracta d'un llistat d'empreses sobre el que s'ha tingut algun conveni i que s'empra com a base per a contactar-ne amb aquestes I veure si estan dispostes a fer-ne una pràctica o no. Originalment d'uns 200 registres, es va haver de fer una neteja fins a aplegar als 98, ja que no mostrava una hegemonia a l'hora de representar la informació, la qual cosa complicava l'extracció de les dades. Finalment va quedar la següent estructura:

- **Nom:** aquesta vegada no està anonimitzat, ja que és d'interés públic. S'empra per a identificar-ne a l'empresa en l'assignació.
- **Ciutat:** ciutat on opera l'empresa. S'empra en el càlcul de la distància dels alumnes cap a aquesta empresa.
- **Pràctiques:** un camp per cada cicle, amb el seu valor indicant el nombre de pràctiques que l'empresa té disponibles per a alumnes d'aquest cicle. En cas de no disposar de cap pràctica per a aquest cicle el camp està buit o mostra un 0.

A causa de la neteja d'empreses, el nombre de pràctiques era inferior al nombre d'alumnes disponibles. Per tant, a l'haver-hi alumes sense empresa, s'incomplia la restricció d'un alumne per pràctica i, per tant, el problema no tenia solució. Amb tal de donar solució a aquest problema, es va reduir el nombre d'alumnes per full de càlcul a uns 15, fent un total de 60 alumnes.

4. Aplicació Web

En aquesta secció s'enumeraran els requeriments de l'aplicació (4.1.) i es farà una tria justificada de les ferramentes web necessàries per al compliment d'aquests (4.2.), així com es presentarà l'estructura de la base de dades (4.3.) i de la informació mostrada per l'aplicació (4.4.). Seguidament es mostrarà com l'aplicació interactua amb la base de dades a partir del diagrama de flux (4.5.) i es mostraran els esquemes (4.6.) i maquetes (4.7.) que serveixen de disseny de la interfície. Finalment es detallarà les distintes funcionalitats (4.8.) i com s'ha anat provant el seu funcionament (4.9.).

4.1. Requeriments de l'Aplicació

En aquest apartat s'enumeren els requeriments que es van tindre en compte a l'hora de desenvolupar l'aplicació. Primer es detallaran els que giren entorn dels quatre tipus d'usuari als que ha de donar suport l'aplicació: alumne, professor, empresa i coordinador (4.1.1.). Finalment es detallaran els que giren entorn de les peculiaritats tècniques de l'aplicació (4.1.2).

4.1.1. Requeriments d'Usuari

En el cas de l'alumne, ha de permetre modificar el seu perfil: nom i cognoms, per a la seua correcta identificació; ciutat de residència, per a poder ubicar-lo i tractar d'assignar-li a empreses properes; disponibilitat de cotxe, per a tindre constància de si es pot permetre anar a empreses llunyanes; les seues preferències, per a saber per quin tipus de pràctica sent més afinitat i les seues observacions, per a indicar qualsevol altra cosa que vulga que es tinga en compte en l'assignació.

En el cas de l'empresa, ha de permetre:

- Modificar el seu perfil: nom d'empresa, per a la seua correcta identificació; població, per a poder assignar-li els alumnes més pròxims; telèfon i correu, com a mètode de posar-se en contacte amb aquestes; persona de contacte, o la que contestarà a les cridades i correus i si volen pràctica que, en cas de marcar que sí, podran accedir al formulari d'inserció de pràctiques).
- Introduir pràctiques, que estarà disponible en cas de marcar que sí que estan interessats a fer pràctica en el formulari de perfil. Conté els següents camps: nom de pràctica, per a la seua correcta identificació; titulació, o perfil d'alumne buscat; descripció, amb informació sobre en què consisteix la pràctica i tecnologies i marcs de treball, on es llisten les ferramentes emprades per a la realització de la pràctica.
- Visualitzar pràctiques assignades i disponibilitat.

En el cas del professor, ha de permetre:

- Modificar el seu perfil: nom i cognoms, per a la seua correcta identificació; titulacions, per a llistar els perfils d'alumne als que pot tutoritzar i hores alliberades

setmanalment, per a indicar-ne quantes hores té disponibles per a fer el seguiment dels alumnes que tinga assignats.

- Ajustar les seues ràtios d'alumnes per hores alliberades: per cada *x* nombre d'alumnes, quantes hores de la seua disponibilitat pot oferir.
- Visualitzar alumnes assignats d'FCT i DUAL, el total d'hores alliberades i la disponibilitat que li queda, amb aquesta disponibilitat sent presentada en forma de percentatge.

En el cas del coordinador de pràctiques, ha de permetre:

- Introduir/modificar informació d'alumnes, professors i empreses. Són els mateixos camps mencionats anteriorment més alguns addicionals als quals sols pot tindre control el coordinador de pràctiques.

  En el cas de l'alumne aquests serien: grup, o cicle formatiu al qual pertany, que s'empra per a poder assignar-li pràctiques que busquen alumnes d'aquesta titulació; tipus de pràctica, on es pot seleccionar entre FCT i DUAL; ai accedeix a FCT, per a indiciar-ne si compleix els requeriments per a fer la pràctica; aporta pràctica, per a tindre en compte durant l'assignació automàtica que l'alumne ja ha entrat en contacte amb una empresa i el si és d'Erasmus, per a tindre en compte el si és estranger.

  En el cas del professor i de l'empresa no hi ha canvis, però sí en el formulari de pràctiques d'aquesta última, on s'afegeixen: tutor del centre, o professor encarregat de supervisar la pràctica.

- Importar informació d'alumnes, professors i empreses mitjançant fitxers Excel o CSV (aquests han de tindre l'estructura descrita en l'apartat 3.3.).
- Exportar informació d'alumnes, professors i empreses en format Excel o CSV.
- Ajustar les ràtios d'alumnes per hores alliberades de cada professor.
- Realitzar assignació automàtica.
- Realitzar/modificar assignació manualment.

Amb tal de complir tots aquests requeriments, convé estructurar la base de dades de tal forma que siga fàcil emmagatzemar i accedir a les dades.

4.1.2. Requeriments Tècnics

Addicionalment, ens trobem amb les següents particularitats tècniques:

- **Python és un llenguatge de programació d'ús obligatori en la part BackEnd:** al fer ús d'OR-Tools, i deguda la necessitat de processar dades de fitxers i de calcular la distància entre els alumnes i les pràctiques d'empresa, la seua versatilitat el fa el llenguatge més adequat.
- **El càlcul de les distàncies s'ha de guardar en cau:** per cada 15 alumnes, el temps que comporta aquest càlcul puja 5 minuts. Això és perquè, per cada alumne registrat, l'algoritme recorre totes les empreses, arreplegant la ciutat de residència de tant alumne com empresa i extraient les seues coordenades fent ús de la llibreria Nominatim. A partir d'aquestes coordenades, es calcula la distància entre les ciutats manant una petició de càlcul de traçat de ruta més curta a OpenStreetMap, atenent si l'alumne disposa de cotxe o no.

  En el cas d'un centre de formació professional amb molt de cicles el procés anteriorment descrit pot portar a una duració superior a les dues hores, amb els consegüents perjudicis que podria comportar-ne en el cas de ser necessari tornar a executar el procés d'assignació. Per tant, és indispensable tindre alguna forma de guardar-ne en cau aquestes dades.
- **L'assignació ha de fer-se en segon pla**: pels mateixos motius esmentats en l'anterior punt, convé que el procés d'assignació es faça en segon pla, permetent així al coordinador realitzar altres tasques de mentre espera que finalitze el procés. A més s'haurà d'eliminar qualsevol mena d'incertesa que puga tindre el coordinador sobre l'estat del procés.
- **L'aplicació està pensada per a poder ser adaptada i emprada per qualsevol institució:** per tant, l'aplicació ha de ser modular, optimitzada per al seu ús en servidors de distinta capacitat i amb una paleta de colors i logotip fàcil de canviar. També ha de fer ús de la menor quantitat de llibreries i de llenguatges de programació possible per a afavorir el seu manteniment.
- **A l'hora de fer proves, convindria que aquestes es feren sobre una base de dades de pega**: això és de vital importància per l'important que aplega a ser el procés d'assignació per a aquesta aplicació. Si s'aplegara a trobar una errata a dies de fer l'assignació i, a l'executar el test, la base de dades real es perdera, o s'inseriren dades falses que no s'esborren, es perdria molta informació important que costaria recuperar, amb els consegüents perjudicis de totes les parts.

Amb tal de complir tots aquests requeriments, convé triar les ferramentes web més adequades. Aquesta tria i les raons darrere d'aquestes es detallen en el següent apartat.

4.2. Tria de Ferramentes Web

En aquest apartat es detallara el procés de raonament darrere de la tria de les ferramentes web més adequades per a complir amb els requeriments esmentats en l'apartat anterior. Primer s'explicarà quin marc de treball s'ha triat d'entre els exposats en el punt 3.6.2 per a cobrir la necessitat de fer ús de Python (4.2.1.). Finalment, es farà el mateix per al tipus de base de dades i de com ens pot permetre canviar a una no persistent en cas de trobar-se en mode desenvolupament així com emmagatzemar el càlcul de les distàncies en cau (4.2.2.).

4.2.1. Marc de Treball

Com a marc de treball s'ha escollit finalment Flask. A part de per estar basat en Python, les raons obeeixen la simplicitat de la mateixa aplicació, que està principalment basada en formularis, amb les parts més complexes sent les que involucren l'algoritme d'optimització.

Flask comença amb el bàsic: un sistema basat en rutes amb les que muntar l'API i un motor de plantilles anomenat Jinja2 que simplifica la cridà a aquesta API i el tractament de les dades que retorna. Això es pot anar expandint amb les anomenades extensions a mesura que l'aplicació es trobe amb nous requisits. Com aquesta està pensada per a ser adaptada a qualsevol institució, i cada institució tindrà els seus propis requeriments, a partir d'un projecte base aquestes poden llevar-ne i afegir les extensions que facen falta.

Entre les extensions de les quals faciliten el desenvolupament de l'aplicació es troben Flask-WTForms per a la creació i gestió de formularis i Flask-Session per a l'emmagatzemament de dades en cau.

Addicionalment hi ha molta documentació sobre l'ús de Celery (una llibreria per a l'execució asíncrona de tasques en segó pla) en el marc de treball. A part de Celery, l'extensió Flask-Worker i la llibreria RQ estaven entre les altres opcions per a realitzar aquest tipus de tasca, però finalment es va optar per Celery al ser l'opció més coneguda, flexible i escalable, qualitats imprescindibles per al que aplega a ser el pilar de l'aplicació i, per tant, la part que més interessa optimitzar.

Entre altres opcions estava Angular, el qual podia inclús complementar-se amb el mateix Flask fent les cridades a l'API, i que podria facilitar la paginació i filtrat de dades, però es va desestimar al complicar l'estructura de directoris, el tractament de les dades i l'execució de la mateixa aplicació (sumant Flask i Celery, seria necessari tres terminals per a tindre l'aplicació funcional). A més a més, altres facilitats que Angular proporcionava, com el filtratge per canonades i el tractament de formularis ja venia amb Flask.

Django també es va desestimar per la seua complexitat, encara que té prou semblances respecte a Flask.

4.2.2. Base de Dades

Pel que fa a base de dades, s'ha optat finalment per dos de tipus no relacional: MongoDB i Redis.

Les raons obeeixen aquesta vegada que la informació a emmagatzemar no té relacions complexes i al fet que aquest tipus de base de dades sol ser més fàcil de gestionar. No tindrà relacions complexes perquè les col·leccions de documents es basen en els distints rols de l'aplicació: alumnes, professors i empreses, que alhora són usuaris d'aquesta. Per a establir la relació entre l'usuari i el rol, el nom del primer apareixeria en la mateixa col·lecció que a la que fa referència el rol. Per altra part, serà més fàcil de gestionar per la similitud de la seua estructura amb el d'un objecte. En el cas de *MongoDB*, per exemple, ens pot permetre amb l'ajuda de l'extensió *flask-wtforms* crear-ne formularis de forma automàtica a partir dels atributs d'una col·lecció i poblar els camps a partir de les dades dels documents que la integren.

També es troba l'avantatge de l'escalabilitat, que pot vindre prou a mà en el cas de les institucions que tinguen interés en tindre un històric dels cursos.

A més a més, MongoEngine, la llibreria emprada per al suport de MongoDB, compta internament amb un mode no persistent anomenat MockDB. Aquest permet realitzar les mateixes operacions que es poden realitzar en el mode producció sobre la base de dades, amb l'avantatge que la informació serà emmagatzemada durant el temps que dure la sessió, és a dir, fins que el procés de Flask siga tancat. Això ens permet realitzar proves sobre l'aplicació i correccions d'errors sense mesclar les dades reals ja existents en la base de dades amb dades falses, complint per tant amb el requisit de fer ús d'una base de dades de pega per al testeig de l'aplicació.

Per altra part, tant Celery com Flask-Session tenen opció per a fer ús de Redis per a l'emmagatzematge de dades temporals. En el cas de Celery, aquestes consisteixen en les coles de missatges manats als workers (els processos que executen les tasques en segon pla) pel client i en els resultats de dits processos. En el cas de Flask-Session, emmagatzemaria el càlcul de les distàncies realitzat durant el procés d'assignació però, a causa del fet que Flask-Session realitza peticions HTTP per a l'emmagatzematge d'aquestes dades, i Celery no admet l'execució de dites peticions, s'ha d'emmagatzemar les distàncies en una base de dades Redis i, quan finalitze el procés d'assignació, emmagatzemar-la en la variable de sessió. Encara que dit procés sembla redundant, es compensa per l'avantatge de Flask-Session que les dades emmagatzemades es comparteixen entre dispositius sempre que l'usuari estiga connectat al mateix compte del navegador. Per tant, si el coordinador realitza el procés d'assignació des del seu despatx, i vol tornar-ho a repetir-ho des de casa, el càlcul de les distàncies s'ometrà perquè el navegador ha associat al seu compte el càlcul de les distàncies.

Siga tot dit, es procedeix a presentar l'estructura de la base de dades.

4.3. Estructura de la Base de Dades

En aquest apartat es presentarà l'estructura de la base de dades. Aquesta, com ja s'ha esmentat en el anterior apartat, és no relacional i, per tant, les dades es troben organitzades de forma que simplifique el seu accés.

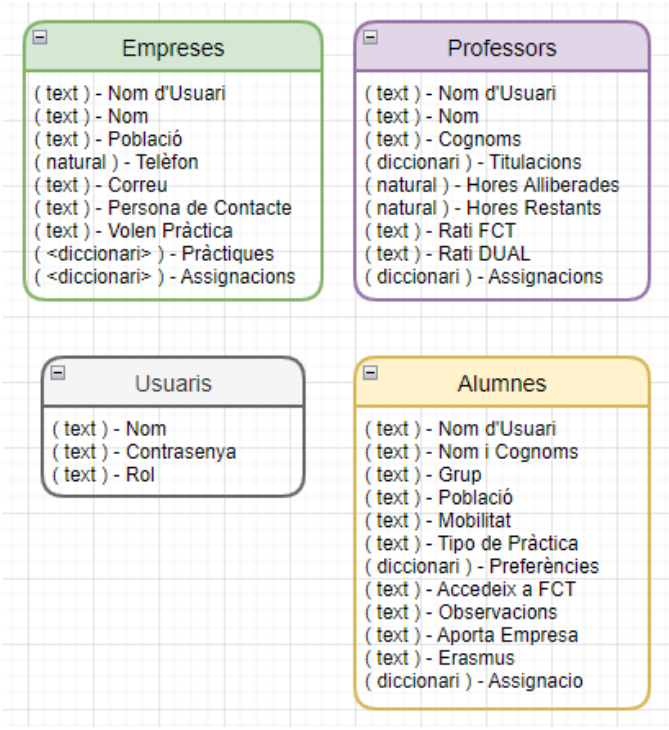

Figura 1: Estructura de la Base de Dades.

Font: Elaboració Pròpia.

En la Figura 1, es poden observar les 4 col·leccions de documents que componen la base de dades: usuaris, que inicien sessió en l'aplicació i, d'acord amb el seu rol, disposen d'accés a la part que els pertany; alumnes, professors i empreses, que contenen informació proporcionada per aquests o per el coordinador com a usuaris de l'aplicació a través dels formularis corresponents, així com les pràctiques que tinguen assignades.

A excepció de les dades de contacte de l'empresa, tota la informació emmagatzemada influeix d'alguna forma en el procés d'assignació, ja siga per la selecció automàtica d'acord amb si l'alumne ja ha aportat empresa o de la distància entre la seua llar i la de l'empresa on va a fer la pràctica, o per factors subjectius en una selecció manual, com puga ser el si l'alumne és d'Erasmus i les observacions donades per aquest.

A causa de la delicadesa d'algunes dades, i que el registre el fa el mateix coordinador, les tres parts deuran donar el seu consentiment. A finals de cada curs, la informació emmagatzemada es purgarà per a donar pas a la importació d'informació més actual.

4.3. Disseny de l'Aplicació

En aquesta secció es tractarà tot el referent al disseny de l'aplicació. Es començarà mostrant l'arquitectura de la informació per una millor visualització de la distribució de pàgines (4.3.1.), seguidament es mostraran els esquemes de pàgina i maquetes (4.3.2. i 4.3.3.) per a mostrar com l'usuari veurà representada la informació i, finalment, es mostrarà el diagrama de flux, on es podrà apreciar la interacció de l'aplicació amb la base de dades per a la mostra i actualització de les dades (4.3.4.).

4.3.1. Arquitectura de l'Aplicació

En aquest apartat és mostra un mapa tecnològic de l'aplicació, on es poden apreciar les tecnologies emprades i la relació que hi ha entre aquestes.

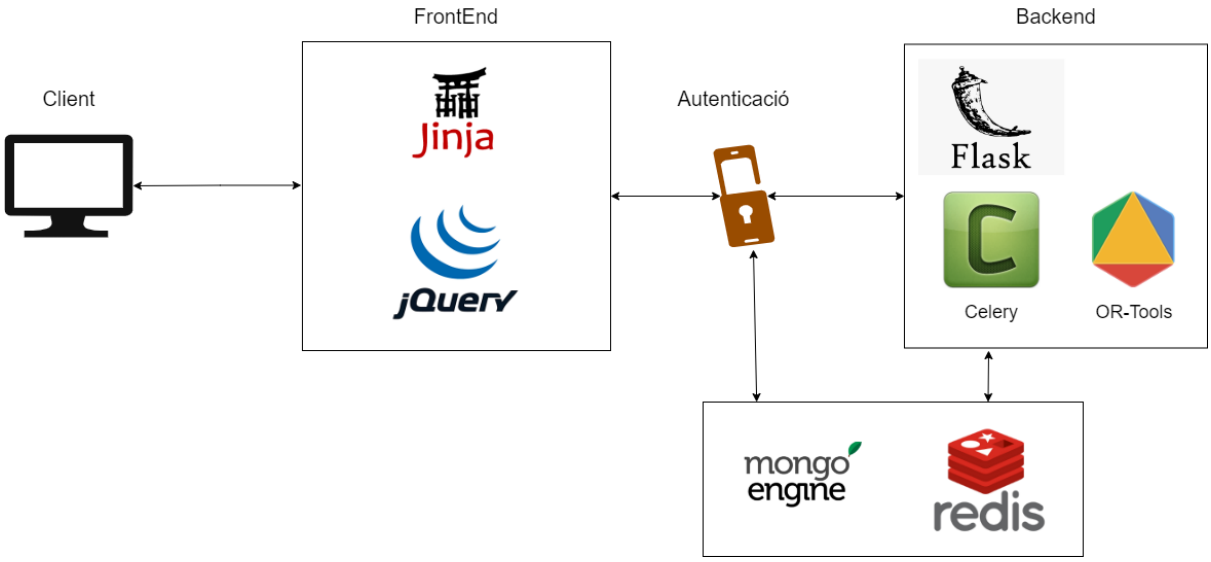

Figura 2: Mapa de Tecnologies

Font: Elaboració Pròpia

En la part *FrontEnd* disposem de Jinja2, un motor de plantilles centrat en la presentació dinàmica de les dades de cara a l'usuari. Les seues limitacions són compensades amb codi *JavaScript*, i solen girar entorn de la realització de peticions AJAX complexes i al filtratge de dades en taules. Per a accedir a aquestes dades, l'usuari ha d'autenticar-se.

La part *BackEnd* es troba impulsada per *Flask*, fent servir com a base de dades persistent *MongoDB* y com a base de dades de cau Redis. Una altra tecnologia emprada en aquesta part és *Celery*, que s'encarrega de l'execució en segon pla de la tasca d'assignació, gestionada alhora per OR-Tools, que és la més computacionalment laboriosa i que també fa ús de Redis per a emmagatzemar-ne els missatges manats i els resultats de la tasca.

4.3.1. Navegació

En aquest apartat es mostrarà un esquema on es reflecteix el camí que l'usuari deurà seguir per a trobar-se amb les distintes seccions de l'aplicació.

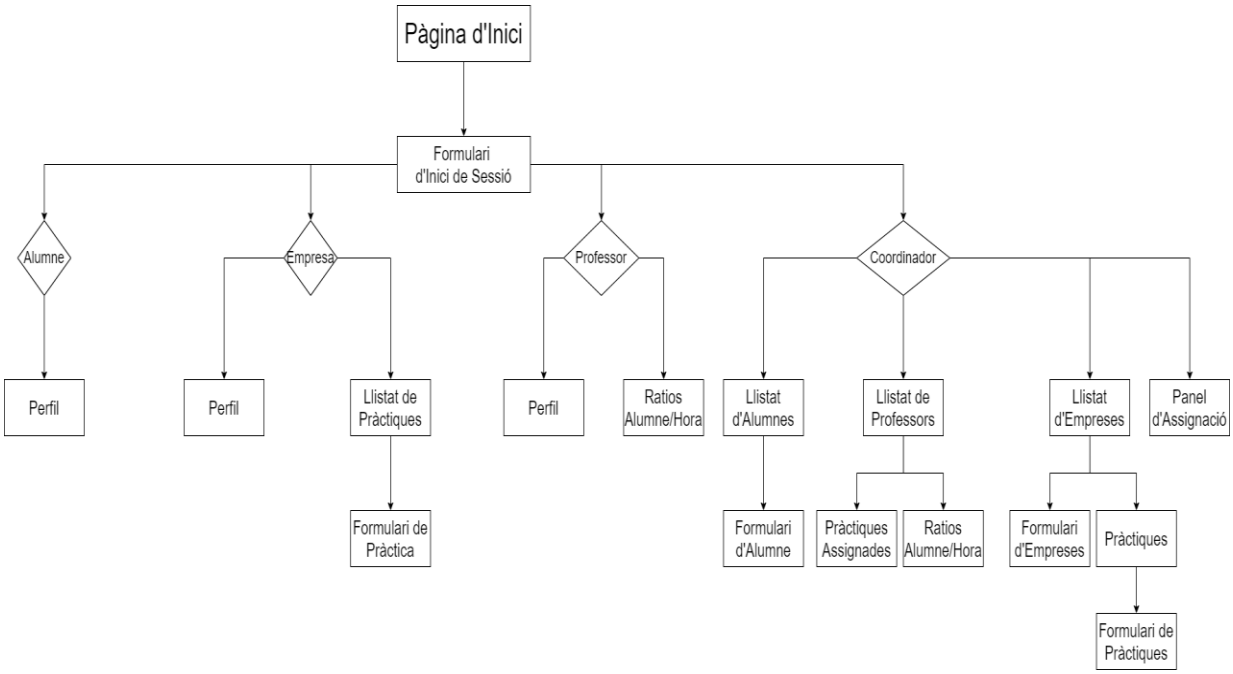

Figura 3:Arquitectura de la Informació.

Font: Elaboració Pròpia

Com s'aplega a apreciar en la Figura 3, tots els usuaris començaran en la pàgina d'inici de la seua institució i tindran l'opció d'iniciar sessió. En cas de fer-ho, solament veurà la informació que resulte important per al seu rol.

- **Alumnes:** solament podran actualitzar el seu perfil, amb informació que es tindrà en compte durant el procés d'assignació.
- **Professors:** podran actualitzar el seu perfil i les ràtios necessàries per a traure el percentatge de disponibilitat, que es tindrà en compte durant l'assignació.
- **Empreses:** podran actualitzar el seu perfil i veure un llistat de les seues pràctiques. En cas de faltar-ne, tindran l'opció d'afegir una així com actualitzar i/o esborrar les ja existents.
- **Coordinador:** aquest tindrà accés a les dades de tots els alumnes, professors i empreses, a través de llistats. De no existir dades d'alguna de les parts podrà importar-ne o afegir-ne manualment, amb excepció dels professors amb els quals vindrà ja carregada l'aplicació. Addicionalment podrà veure les assignacions i disponibilitat d'empreses i professors, així com inserir pràctiques o ajustar les ràtios de disponibilitat. Tot això per a tindre-ho tot preparat per a l'assignació.
  Quan ho tinga tot preparat, podrà realitzar l'assignació automàtica en el panel d'assignació, per a després fer les rectificacions que veja convenient.

4.3.2. Esquemes de Pàgina

En aquest apartat es mostraran els esquemes que s'han realitzat de l'aplicació per a tindre una referència de com es mostrarà la informació.

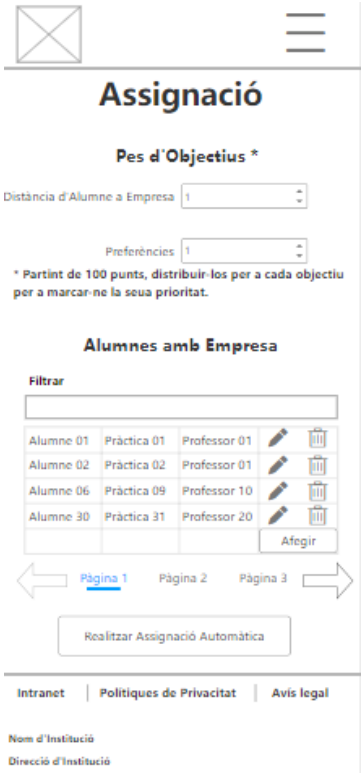

Figura 4: Esquema de la Pàgina d'Assignació.

Font: Elaboració Pròpia.

Com s'aplega a apreciar en la Figura 4, l'aplicació disposarà d'un encapçalat on apareixerà el logotip de la institució, el qual podrà ser fàcilment canviat substituint el que ve per defecte en l'aplicació. A la dreta es troba el botó d'iniciar sessió que per a l'usuari autenticat serà substituït per una icona de barres que obrirà el menú al fer clic sobre aquesta.

Pel que fa al peu de pàgina, disposarà d'enllaços a pàgines d'interés de la Institució, per a afavorir la interconnexió amb aquesta.

La part dinàmica es trobarà en el cos de la pàgina que, o bé disposarà d'un formulari que l'usuari haurà d'emplenar, o d'una taula que llista un tipus determinat de dades (e.g., alumnes, professors, assignacions i empreses), així com les distintes opcions amb les quals interactuar amb aquestes.

La resta d'esquemes, es poden consultar en l'annex A.

4.3.3. Maquetes

En aquest apartat es mostraran les maquetes que s'han realitzat de l'aplicació per a tindre una referència de la paleta de colors emprada.

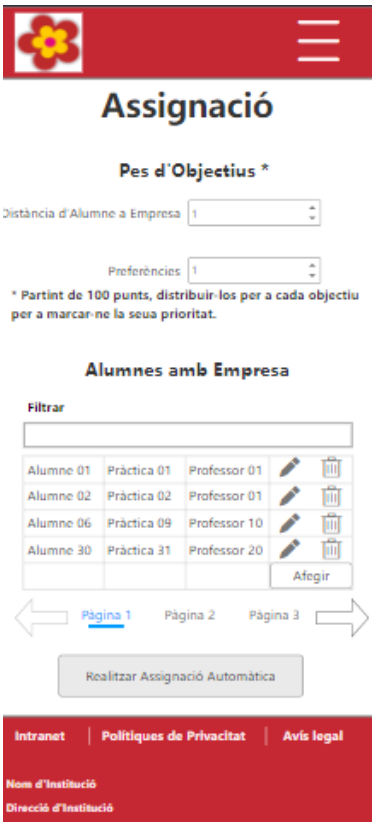

Figura 5: Maqueta de la pàgina d'assignació.

Font: Elaboració Pròpia.

La paleta, així com el logotip de la Figura 5, corresponen al de l'aplicació base, amb qualsevol institució podent adaptar-ho a la seua marca corporativa. Això s'aconseguirà amb variables SASS i substituint el logotip pel seu.

Quant al cos, es fa ús simplement d'un color de fons blanc sobre el qual es posara la informació. L'objectiu és tindre una aplicació bonica però sense distraccions, amb tal que l'usuari se senta còmode entrant en ella i puga realitzar les seues tasques sense complicació.

La resta d'esquemes, es poden consultar en l'annex B.

4.3.4. Diagrama de Flux

En aquest apartat es mostrarà un diagrama de flux a través del qual es podrà apreciar les interaccions de l'aplicació amb la base de dades per a fer fluir la informació.

L'usuari (siga alumne, professor, empresa o coordinador), accedeix a la pàgina d'inici del lloc web de la seua institució. Li dona al botó d'iniciar sessió i se l'apareix un formulari per a iniciar sessió.

Deguda la naturalesa restringida de l'aplicació, no té l'opció de registrar-se. El compte es crea en el moment en què donen consentiment a formar part del programa de pràctiques de la institució, i les credencials se les entrega pel mig elegit per la institució.

Per temes de seguretat, tampoc hi ha opció de recuperar contrasenya. De tindre problemes per accedir deurà contactar amb la institució.

Una volta iniciada la sessió, pot passar 4 coses en funció del rol que exerceix l'usuari:

- **Alumne:** se l'apareixerà un formulari on podrà actualitzar les seues dades.

- **Empresa:** se l'apareixerà una pàgina amb una taula que conté informació de les distintes pràctiques que ofereix, amb opció d'afegir més pràctiques o actualitzar-ne les existents.

Alternativament, pot modificar la informació de l'empresa accedint a la corresponent opció del menú.

- **Professor:** se l'apareixerà un formulari on pot actualitzar les seues dades. A partir del menú podrà accedir a una secció on podrà ajustar la ràtio d'alumnes d'FCT/DUAL per tal hores alliberades.

- **Coordinador:** se l'apareixerà la pàgina on poder fer les assignacions, siguen manuals o automàtiques. Alternativament, a partir del menú podrà accedir al llistat d'alumnes, professors i empreses i modificar les dades de qualsevol d'aquests.

Per a més informació, consultar l'annex C.

4.3. Estructura del Projecte

En aquesta secció es farà un recorregut per l'estructura de directoris del projecte, posant èmfasis en la importància que cada fitxer té en l'exerció de les funcions de l'aplicació.

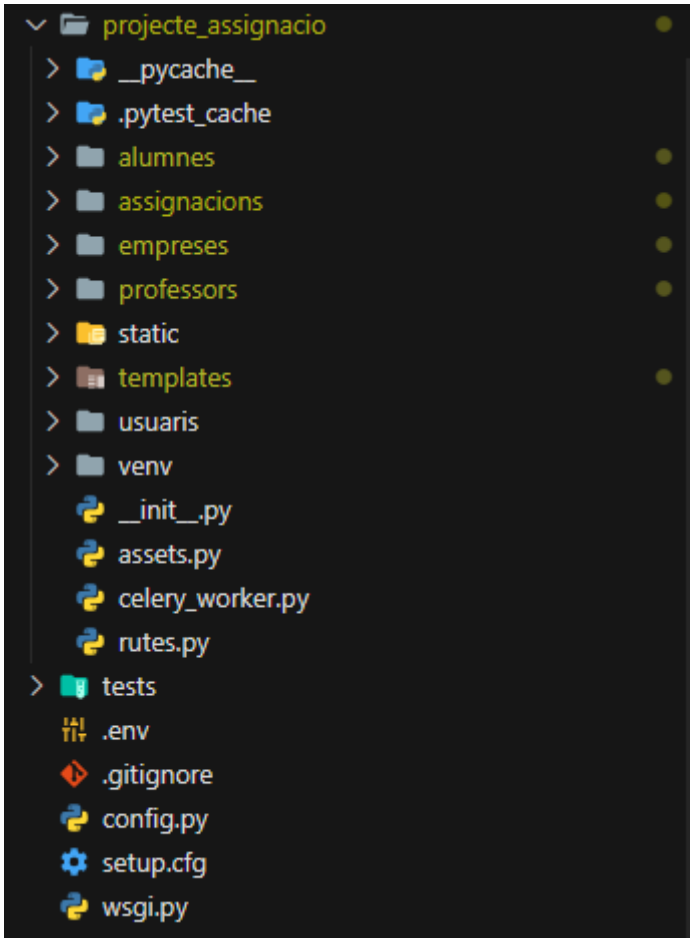

Figura 6: Estructura del Projecte

Font: Elaboració Pròpia

El projecte ha sigut creat seguint el patró Fàbrica de Flask, que busca resoldre el dilema de la importació circular que te lloc en aquest marc de treball quan es busca crear una aplicació modular.

Aquest problema consisteix en el fet que, a l'executar l'aplicació, s'importen totes les llibreries de *Flask* declarades en el fitxer d'engegada, i s'instància un objecte que representa l'aplicació, a partir del qual es configura l'aplicació i es declaren les rutes. Aquesta forma de treballar no ocasiona cap inconvenient si totes les rutes i funcions és troben dins d'aquest fitxer però, tindreu tot dins d'aquest no és factible quan l'aplicació està en continu creixement. És a partir d'ací que es busca modularitzar l'aplicació però, com es depén d'aquest objecte d'aplicació, per cada mòdul s'ha d'importar el fitxer d'engegada per a recollir dit objecte i, a l'hora, el

fitxer d'engegada ha d'importar el mòdul per a fer-lo accessible i visible dins de l'aplicació., el que acaba provocant el cercle viciós d'importacions que dona nom al problema.

El patró Fàbrica resol aquest problema a partir del context d'aplicació de *Flask*, que en resumits comptes ve a significar que tots els fitxers i mòduls que conformen l'aplicació són tractats com si foren un sol i, per tant, poden vores i interactuar els uns amb els altres, com si d'una fàbrica es tractara. Aquest context es crea en el fitxer d'engegada on, al crear-ne l'aplicació, s'indica a *Flask* que mòduls i fitxers existeixen.

L'ús d'aquest patró porta a l'estructura de directoris vista en la Figura 6, dels amb els fitxers d'execució i configuració en l'arrel, mentre que els que contenen les funcionalitats es troben en la carpeta del projecte. A la mateixa altura que la carpeta de projecte, es troba la que conté els tests.

4.3.1. Arrel

Dels fitxers en l'arrel ens trobem amb:

- ***wsgi.py*****:** actua d'executable de l'aplicació. A causa del patró fàbrica, tota l'aplicació està continguda en una funció dins del fitxer__init__.py de la carpeta del projecte, la qual retorna l'aplicació ja compilada. Aquesta és arreplegada i executada per aquest fitxer en la direcció especificada o en totes les direccions que puga trobar.

- ***setup.cfg*****:** indica a *pytest* (la ferramenta per a provar el funcionament del codi) on està la carpeta amb els tests, mentre que a *coverage* (la ferramenta per a analitzar el codi font i saber quant d'aquest ha sigut provat) li indica on està la carpeta amb el codi font.

- ***config.py*****:** conté les variables de configuració, agrupades en 3 classes: una de desenvolupament de la que solament es fa ús quan es vol fer les tasques de desenvolupament i manteniment del codi o quan s'executen els tests, una de producció quan l'aplicació es troba desplegada en un servidor i una base, amb les configuracions compartides en ambos casos.

Quan el projecte es troba en mode de desenvolupament, els errors són mostrats per terminal o dins de la mateixa aplicació a partir d'una pantalla personalitzada de *Flask,* i *mongoengine* apunta a una base de dades de pega que guarda internament i cull contingut es perd al tancar la sessió, mentre que quan està en mode producció, la base de dades a la que apunta és la real i les dades perduren.

La configuració base compta amb la clau secreta que serà emprada per al xifrat de les contrasenyes, la direcció cap a les carpetes *static* i *templates* base del projecte (que contenen els actius de l'aplicació), així com les comandes a executar per comprimir els fitxers d'estils *.less*.

```python
class DevConfig(Config):
    """Mode Desenvolupament.

    Args:
        Config (class): Configuració Base.
    """
    FLASK_ENV = 'development'
    DEBUG = True
    ASSETS_DEBUG = True
    TESTING = True

    ###################################
    ## Base de Dades no Persistent. ##
    ###################################
    MONGODB_SETTINGS = {
        "db": environ.get('DEV_MONGO_DB'),
        "host": environ.get('DEV_MONGO_URI')
        }
    ###################################
    ###################################

    ############################
    ## Execució en Segón Pla. ##
    ##       (Worker)         ##
    ############################
    CELERY_CONFIG={
        'broker_url': 'redis://localhost:6379/0', ## Base de dades per a emmagatzemar els missatges.
        'result_backend': 'redis://localhost:6379/0', ## Base de dades on s'emmatgatzemen els resultats.
    }       David F.F, el mes pasado • Worker d'assignació implementat. …
    ############################
    ############################
```

Figura 7: Configuració del Mode Desenvolupament

Font: Elaboració pròpia

- ***.env*:** en el cas de les variables de configuració més delicades en *config.py* (com puguen ser la clau secreta), el seu contingut està allotjat en aquest fitxer que, que apareixerà com ocult en el servidor i que no serà pujat al repositori git al ser especificat la seua restricció en el fitxer *.gitignore*.

```
## Clau secreta per al xifrat de contrasenyes.
SECRET_KEY = "_b=X)Jn&hFHp5RfKPY@W(8|L~oey3C4i,

## Variables de Sessió.
SESSION_TYPE = "redis"
SESSION_REDIS = "redis://localhost:6379"

## Base de dades emprada en producció.
PROD_MONGO_DB = "FP"
PROD_MONGO_URI = 'localhost:27017'

## Base de dades de pega per a tests.       Davi
DEV_MONGO_DB = "test"
DEV_MONGO_URI = 'mongomock://localhost'
```

Figura 8: Contingut del fitxer .env

Font: Elaboració pròpia

4.3.2. Carpeta del Projecte

Dins de la carpeta del projecte ens trobem amb:

- ***__init.py__:*** carrega les variables de configuració i les llibreries i, dins d'un mateix context (la fàbrica que dona nom al patró), registra els distints mòduls (coneguts com a *Blueprints*) i compila els actius. La variable retornada conté tota la informació de l'aplicació, que és executada pel fitxer *wsgi.py* en l'arrel del projecte.

```
with app.app_context():
    ######################
    # Inclusió de Rutes #
    ######################
    from .usuaris import sessio, rutes_usuaris as usuaris
    from .alumnes import rutes_alumnes as alumnes
    from .professors import rutes_professors as professors
    from .empreses import rutes_empreses as empreses
    from .assignacions import rutes_assignacions as assignacions
    from .assets import compilar_actius_estatics
    from . import rutes
    ##################################################################
    ##################################################################

    ####################
    # Registrar Mòduls #
    ####################
    app.register_blueprint(usuaris.usuaris_bp)
    app.register_blueprint(alumnes.alumnes_bp)
    app.register_blueprint(professors.professors_bp)
    app.register_blueprint(empreses.empreses_bp)
    app.register_blueprint(assignacions.assignacions_bp)
    ######################################################
    ######################################################

    compilar_actius_estatics(assets)
    return app
## with
```

Figura 9: Context de l'aplicació

Font: Elaboració Pròpia

- ***assets.py:*** comprimeix els fitxers js i less generals de l'aplicació i els específics de cada mòdul per a una càrrega més ràpida dels estils i del codi JavaScript.

- **celery_worker:** és el fitxer que permet l'execució de *Celery* en l'aplicació *Flask* a través de la comanda *celery -A projecte_assignacio.celery_worker.celery worker --loglevel=debug*. A causa del funcionament de Flask i a l'ús del patró Fàbrica, quan s'executa aquesta ferramenta

s'ha d'assegurar que actue sobre el mateix context de l'aplicació *Flask* a fi d'evitar la importació circular. Amb tal d'aconseguir això, aquest fitxer inicialitza de nou l'aplicació *Flask* i fusiona el seu context amb el context global.

```
import os
from . import celery, init_app

app = init_app('config.ProdConfig')
app.app_context().push()
```

Figura 10: Execució de Celery

Font: Elaboració pròpia

- **rutes.py:** conte una ruta per emprada per a provar el funcionament de Flask en els tests i la que s'encarrega de mostrar la pàgina d'inici. Fitxers de nom paregut apareixen en cada mòdul amb les rutes específiques d'aquests.

Pel que fa a les carpetes, es destaquen:

- ***venv:*** fa referència a l'entorn virtual de *Python* creat per a instal·lar les llibreries de les quals fa ús el projecte.
- ***static*:** conté els fitxers estàtics de l'aplicació, com puguen ser imatges, fonts, fitxers i estils i codi *JavaScript*. Aquests dos últims s'agrupen en dues carpetes: *src*, que conté els fitxers originals i sense comprimir, i *dist*, que conté la versió comprimida:

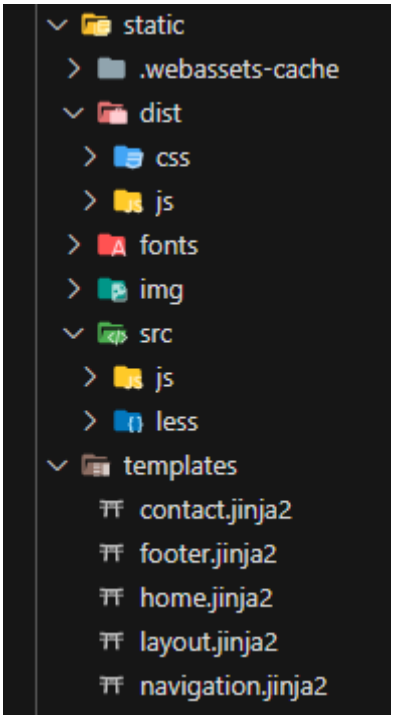

Figura 11: Carpetes d'Actius

Font: Elaboració Pròpia

- ***templates*:** conte els fitxers jinja2, que fan ús del motor de plantilles del mateix nom, la qual els dota de la capacitat d'interpretar i filtrar les variables d'estil Python passades i d'incrustar

altres fitxers de tipus HTML. Aquests fitxers són renderitzats per Flask en funció de la ruta accedida.

4.3.3. Blueprints

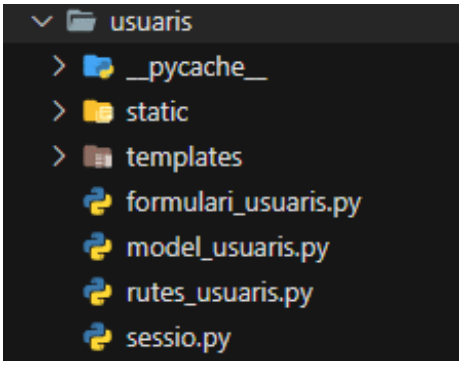

Figura 12: Carpeta Blueprint

Font: Elaboració Pròpia

La resta de carpetes fan referència als *Blueprints* registrats i contenen:

- **Carpetes *static* i *templates*:** contenen els recursos específics d'aquest *Blueprint*.

- **Rutes:** conté una sèrie de funcions que s'activen a l'accedir a certa ruta amb els tipus de petició y paràmetres admesos. Les respostes retornades per les rutes varien en funció dels resultats de l'operació feta a la base de dades i la forma en la qual aquests es mostren a l'usuari varia en funció del mode en la que es troba executada l'aplicació: en mode test de desenvolupament es representen com una resposta JSON, mentre que en producció redirigeixen a l'usuari a una pàgina on el resultat de l'operació és mostrat a través d'un missatge emergent. També retornen la pàgina a ser mostrada a l'usuari en cas que la ruta estiga destinada a recuperar una pàgina.

- **Model:** la classe que representa a la col·lecció de documents a la que fa referència el *Blueprint*. A efectes pràctics, dona forma a l'objecte sobre el qual fer les operacions, definint totes les seues propietats.

- **Formulari:** : una classe que conté els camps, requisits i forma de validar d'un formulari. Al crear una instància d'aquesta, es pot renderitzar els camps desitjats de el dit formulari en el document .jinja2.

Addicionalment, el *Blueprint* usuaris disposa del fitxer *sessio.py* amb les funcions centrades en la gestió de la sessió, com el manteniment de la sessió cada volta que l'usuari accedeix a una ruta o la restricció d'usuaris invitats a les rutes en les quals necessiten estar autenticats.

El *Blueprint* assignacions també disposa d'un fitxer únic anomenat *model_de_optimització.py* que contes les funcions relacionades amb els distints processos de l'assignació, com el càlcul de distàncies, la definició de variables, la definició de restriccions i la definició de la funció objectiu.

4.3.4. Carpeta Tests

Pel que fa a la carpeta de tests, segueix la següent distribució:

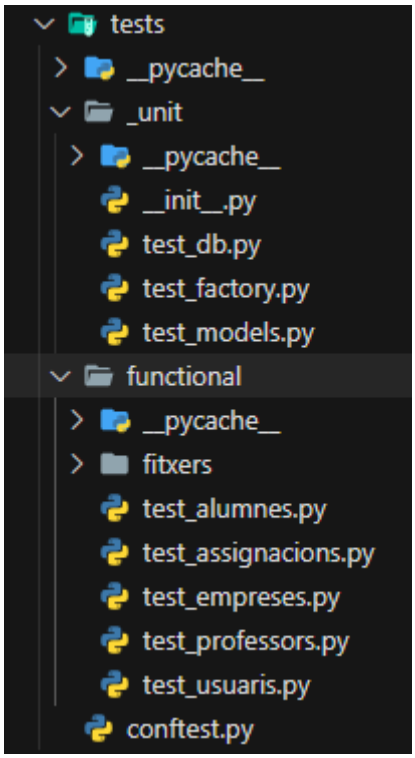

Figura 12: Carpeta Tests

Font: Elaboració Pròpia

- ***conftest.py*:** conté les funcions de configuració del test, que inicialitzen el client Flask i Mongo i instancien les classes necessàries sobre les quals executar els tests.

- **Carpeta *unit*:** conté tests unitaris, és a dir, que posen a prova un procés que forma part d'una de les funcionalitats de l'aplicació, com puga ser la inicialització i tancament del client Flask o de la base de dades i la creació d'una instància de model.

- **Carpeta *functional*:** conté els tests de les diferents funcionalitats de l'aplicació. Cada fitxer dins d'aquesta carpeta se centra en un dels *Blueprints* de l'aplicació, de tal forma que es puga comprovar fàcilment l'estat de cada mòdul.

4.4. Funcionalitats

En aquest apartat es fa un recorregut per les distintes funcionalitats de l'aplicació i es detalla el seu funcionament.

4.4.1. Usuaris

En el Blueprint d'usuaris ens trobem amb les següents funcionalitats:

- **Registre:** s'executa durant el procés d'importació d'alumnes, professors i empreses, o de forma manual a través de *peticions a l'API* o d'alguna aplicació *Mongo*. Es comprova que no existisca un usuari amb el mateix nom. De no existir, encripta la contrasenya passada, el crea i inicia la seua sessió. En cas contrari retorna un avís que es mostrarà en la interfície web a partir d'un missatge emergent.

Algoritme 1	Registre
1	`usuari_existent = Usuari.get(nom_usuari);`
2	`if usuari_existent is None then:`
3	`    usuari = new Usuari(nom, pwd, rol);`
4	`    usuari.xifrar_contrassenya();`
5	`    usuari.save();`
6	`end`

Figura 13: Algoritme 1 (Registre)

Font: Elaboració Pròpia.

- **Inici de Sessió:** comprova que la versió encriptada de la contrasenya passada siga idèntica a l'encriptada en la base de dades. De ser cert inicia la sessió de l'usuari, redirigir-lo a la pàgina corresponent i retornant un missatge avisant de l'èxit de l'operació.

Algoritme 2	Inici de Sessió
1	usuari = Usuari.get(nom_usuari);
2	**if** usuari **AND** contrasenya_valida **then**:
3	login_user(usuari);
4	**if** usuari.rol == "Alumne" **then**:
5	redirigir-lo al seu perfil;
6	**elif** usuari.rol == "Professor" **then**:
7	redirigir-lo a ajustos de ratis;
8	**elif** usuari.rol == "Empresa" **then**:
9	redirigir-lo a vista pràctiques;
10	**elif** usuari.rol == "Coordinador":
12	redirigir-lo a panel assignació;
13	**end**
14	**end**

Figura 14: Algoritme 2 (Inici de Sessió)

Font: Elaboració Pròpia

- **Carrega d'Usuari:** per la forma en què funciona *Flask*, cada volta que l'usuari accedeix a una ruta de l'aplicació, s'ha de comprovar que la sessió continue activa i retornar-la.

Algoritme 3	Carrega d'Usuari
1	**if** id_de_usuari **then**:
2	**return** usuari;
3	**else**
4	**return** None;
5	**end**

Figura 15: Algoritme 3 (Carrega d'Usuari)

Font: Elaboració Pròpia

- **Tancar Sessió:** simplement tanca la sessió de l'usuari cridant a la funció interna *logout_user()*.

4.4.2. Alumnes/Professors/Empreses

En els Blueprints d'alumnes, professors i empreses ens trobem amb les següents funcionalitats:

- **Importació:** a partir d'un full de càlcul o csv, s'importen les dades d'aquests a la base de dades, no sense abans comprovar si ja existeixen. Al final de la importació, apareix un avís de quants registres s'hi han inserit i quants dels no inserits ja es trobaven dins de la base de dades.

Algoritme 4	Importació
1	`open arxiu csv:`
2	`    files = llistat de files de l'arxiu;`
3	`    for fila in files do:`
4	`        objecte = new Objecte(fila);`
5	`        registrar(objecte);`
6	`        objecte.save();`
7	`    end`
8	`end`

Figura 16: Algoritme 4 (Importació)

Font: Elaboració Pròpia

- **Exportació:** les dades dins de la col·lecció s'exporten en un full de càlcul.

Algoritme 5	Exportació
1	`objectes = Objectes.getAll();`
2	`llista_diccionaris = [];`
3	`for objecte in objectes do:`
4	`    diccionari = objecte.toDict();`
5	`    llista_diccionaris.append(diccionari);`
6	`end`
7	`df = Dataframe_fromDict(llista_diccionaris);`
8	`df.toExcel();`

Figura 17: Algoritme 5 (Exportació)

Font: Elaboració Pròpia

- **CRUD:** comprèn les funcionalitats bàsiques d'inserció, actualització, recuperació i eliminació de registres.

4.4.3. Algoritme d'Optimització

Aquest algoritme va ser al principi provat en un entorn segur de Google Collab, amb el pas d'aquest a l'aplicació romanent quasi inalterat.

El procés que seguix aquest algoritme, explicat de la forma més breu possible, es el següent:

Algoritme 6	Assignació
1	`alumnes = Alumne.getAll();`
2	`professors = Professor.getAll();`
3	`empreses = Empresa.getAll();`
4	`if not redis.exists("Distancies") then:`
5	`    calcular_distancia(redis, alumnes, empreses);`
6	`end`
7	`variables = definir_variables(alumnes, empreses, professors)`
8	`restriccions = definir_restriccions(alumnes, professors, Yki, Xji)`
9	`funcio = definir_funcio_objectiu(alumnes, professors, Yki, Xji)`
10	`for alumne in alumnes do:`
11	`    if alumne.aporta_empresa == "No" and alumne.accedeix_a_fct == "Sí":`
12	`        for assignacio in funcio[2][alumne.nom] do:`
13	`            alumne.update(assignacio);`
14	`            professor.update(assignacio):`
15	`            empresa.update(assignacio);`
16	`        end`
17	`    end`
18	`end`

Figura 18: Algoritme 6 (Assignació)

Font: Elaboració Pròpia

En primer lloc, s'obtenen els alumnes, professors i empreses emmagatzemats en la base de dades i, de no existir cap distància en cau, es calculen aquestes. Seguidament es defineixen les variables, restriccions i funció del model d'optimització i s'assignen les pràctiques als alumnes i professors, així com s'actualitza la propietat d'assignació d'aquests i de les empreses per a fer de més fàcil el accés a les pràctiques se les han sigut assignades

Tant el càlcul de les distàncies, com la definició de les variables, restriccions i funció objectiu, es realitzen en funcions separades per a facilitar la lectura del codi. En el cas del càlcul de les distàncies, es realitza el següent:

Algoritme 7	Calcular Distància

```
1  distancies = {};
2  for alumne in alumnes do:
3      if alumne.aporta_empresa == "No" and alumne.accedeix_a_fct == "Sí" then:
4          coordinades_alumne = obtindre_coordinades(alumne.poblacio);
5          for empresa in empreses do:
6              perfil == false;
7              for practica in empresa.practiques do:
8                  if practica["Titulacio"] == alumne.grup:
9                      perfil == true;
10                 end
11             end
12             if perfil then:
13                 coordinades_empresa = obtindre_coordinades(empresa.poblacion);
14                 if(coordinades_alumne and coordinades_empresa) is not None then:
15                     transport = "";
16                     if alumne.mobilitat == "Sí" then:
17                         transport = "car";
18                     else:
19                         transport = "foot";
20                     end
21                     distancia = obtindre_distancia(coordinades_alumne, coordinades_empresa);
22                     redis.hset("Distancies", distancia/1000);
23                     distancies = redis.hgetall("Distancies");
24                 end
25             end
26         end
27     end
28 end
```

Figura 19: Algoritme 7 (Calcular Distància)

Font: Elaboració Pròpia

Per cada alumne, es verifica que complisca amb les condicions per a fer pràctiques i que no haja aportat empresa i, de poder fer-ne es recorren les empreses candidates i es comprova que coincidisquen amb el perfil de l'alumne. De fer-ho, s'obtenen les coordinades de les seues

respectives ciutats i s'obté la distància entre aquestes en funció de si l'alumne disposa o no de cotxe. Seguidament, es guarden totes les distancies de l'alumne en cau.

Respecte a la definició de les variables, simplement es creen llistes amb les combinacions de cada alumnes i pràctica que representen la variable Y_{ki} i de cada professor i pràctica, que representen la variable X_{ji}, *sempre i quant complisquen els prerequisits de qualificació i titulació.*

Algoritme 8	Definir Variables
1	solver = "SCIP";
2	variable_alumnes = [];
3	variable_professors = [];
4	**for** alumne **in** alumnes **do**:
5	**if** alumne.aporta_empresa == "No" **and** alumne.accedeix_a_fct == "Sí" **then**:
6	**for** empresa **in** empreses **do**:
7	X = len(empreses.practiques);
8	**while** X > 0 **do**:
9	
10	Y_{ki} = alumne.nom+"-"+empresa.nom+"(Pràctica"+X+")";
11	variable_alumnes.append(Y_{ki});
12	X = X-1;
13	**end**
14	**end**
15	**end**
16	**end**
17	**end**
18	**for** professor **in** professores **do**:
19	**for** empresa **in** empreses **do**:
20	X = len(empreses.practiques);
21	**while** X > 0 **do**:
22	X_{ji} = professor.nom+"-"+empresa.nom+"(Pràctica"+X+")";
23	variable_professors.append(X_{ji});
24	X = X-1
25	**end**
26	**end**
27	**end**
28	**return** [variable_alumnes, variable_professrors, solver];

Figura 20: Algoritme 8 (Definir Variables)

Font: Elaboració Pròpia

La definició de les restriccions es realitza recorrent aquestes llistes i configurant-les per a que solament puga hi haure una activa, en el cas dels alumnes, o tantes actives com hores tinguen alliberades, com és el cas dels professors. També s'assegura que en una mateixa pràctica hi haja tant un professor com un alumne assignats.

Algoritme 9	Definir Restriccions
1	`for alumne in alumnes do:`
2	`  if alumne.aporta_empresa == "No" and alumne.accedeix_a_fct == "Sí" then:`
3	`    c = restricció 1 a 1;`
4	`      for variable in variable_alumnes do:`
5	`        c.establir_coeficient(variable, 1)`
6	`      end`
7	`  end`
8	`end`
9	`for professor in professors do:`
10	`  c = restricció 0 a hores alliberades`
11	`  for variable in variable_professors do:`
12	`    c.establir_coeficient(variable, 1)`
13	`  end`
14	`end`
15	`for practica in practiques do:`
16	`  c_alumne = restricció 0 a 1;`
17	`  c_professor = restricció 0 a 1;`
18	`    for v in variable_alumnes do:`
19	`      c.establir_coeficient(v, 1);`
20	`    end`
21	`    for v in variable_professors do:`
22	`      c.establir_coeficient(v, 1);`
23	`    end`
24	`end`
25	`return [solver, variable_alumnes, variable_professors];`

Figura 21: Algoritme 9 (Definir Restriccions)

Font: Elaboració Pròpia

La definició de la funció objectiu es encara més simple, establint com objectiu la minimització del valor i com a coeficient la distància d'eixa pràctica. Totes les combinacions d'alumne i pràctica es van recorrent i l'algoritme es queda amb la que menys distància de trasllat requereix.

Algoritme 10	Definir Funció Objectiu

```
1   objectiu = minimització;
2   nombre_de_alumnes = 0;
3   for alumne in alumnes do:
4       if alumne.aporta_empresa == "No" and alumne.accedeix_a_fct == "Sí" then:
5           contador = 0;
6           for distancia in redis.get("Distancia") do:
7               nombre_de_practiques = distancia["Nombre de Practiques"];
8               while nombre_de_practiques > 0 do:
9                   objectiu.establir_coeficient(variable_alumne[alumne.nom][contador], distancia);
10              end
11          end
12      end
13  end
14  return [solver, objectiu, variable_alumnes, variable_professors];
```

Figura 22: Algoritme 10 (Definir Funció Objectiu)

Font: Elaboració Pròpia

4.5. Testege de Funcionalitats

En aquest apartat es farà un recorregut pels tests desenvolupats. Aquests són importants perquè ens permeten comprovar en un moment, a través de l'execució d'una comanda, si l'aplicació continua sent funcional i, de no ser-ho, que és el que exactament falla, estalviant temps en la depuració i manteniment del codi.

Com la base de dades emprada en els tests és de pega, podem executar tantes voltes el test com vulguem, sense preocupar-nos d'esborrar les dades reals, independentment de si l'aplicació es trobe en producció, el que resulta convenient atenent a la importància que aplega a tindre les dades durant el procés d'assignació.

El directori *test*, on es troba tot el relacionat amb els tests, disposa de tres part importants: el fitxer *conftest.py* i els directoris *unit* i *funcional*.

4.5.1. conftest.py

En aquest fitxer part es troben les funcions que preparen l'entorn per al test.

- ***test_client()*:** inicialitza el client de l'aplicació en modo test i el retorna per a provar les seues funcionalitats.

- ***test_mongo()*:** inicialitza de forma manual el client *Mongo* i retorna la seua instància per a provar l'apertura i tancament d'aquesta.

- ***nou_usuari()*:** crea un objecte usuari i el retorna per a fer proves.

- ***nou_alumne()*:** crea un objecte alumne i el retorna per a fer proves.

- *nou_professor()*: crea un objecte professor i el retorna per a fer proves.

- *nova_empresa()*: crea un objecte empresa i el retorna per a fer proves.

4.5.2. Tests Unitaris

En aquest directori es troben les funcions que proven el funcionament de les tecnologies de l'aplicació i de processos que es repeteixen al llarg de les funcionalitats. Els fitxers que conté son els següents:

- ***test_mongo*:** conté dos funcions: *test_iniciar_db()* i *test_tancar_db()*, que comproven respectivament si la base de dades s'ha connectat o desconnectat amb èxit.

- *test_factory*: conté dos funcions: *test_acces_valid_a_rutes()* i *test_acces_invalid_a_rutes()*, que comproven respectivament si al accedir a una ruta amb un mètode vàlid retorna el valor esperat o si es realitza amb un mètode invàlid retorna un codi d'error.

- **test_models:** comprova que es puga accedir a les propietats dels objectes passats i de que els seues valors siguen els esperats. Aquests objectes son: usuaris, alumnes, professors i empreses.

4.5.3. Tests Funcionals.py

En aquest directori es troben les funcions que posen a prova les funcionalitats de l'aplicació. Aquestes funcionalitats estan agrupades en fitxers que representen cadascú dels mòduls de l'aplicació: assignació, alumnes, empreses, professors i usuaris, e involucren la interacció amb la base de dades per a inserir, actualitzar, esborrar i obtindre dades:

- ***Inserir*:** comprova que el valor inserit estiga en la base de dades.

- *Actualitzar*: comprova el valor retornat per l'operació d'actualització és 1, que significa que no ha ocorregut cap problema en l'actualització. Seguidament es proba a actualitzar un registre inexistent i es comprova que el valor retornat siga 0.

- **Esborrar:** es comprova que el valor esborrat no estiga en la base de dades.

- **Obtindre Dades:** comprovar que els valors obtinguts siguen els esperats.

4.6. Resultat Final

En aquest apartat es realitzarà una breu descripció de l'estat final de l'aplicació, juntament amb captures de pantalla sobre aquesta.

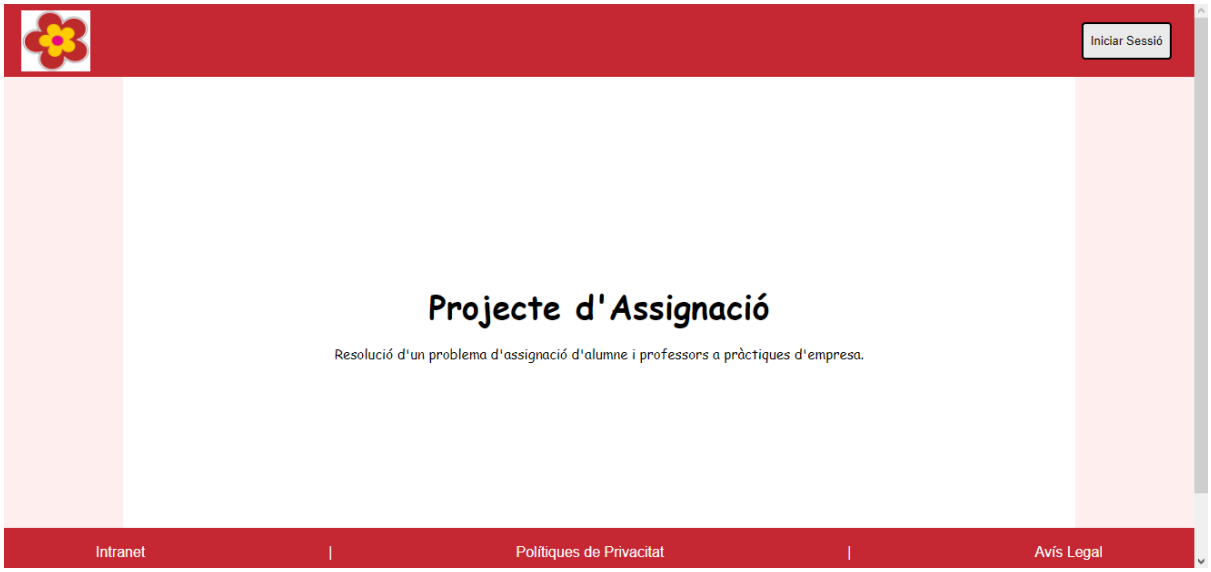

Figura 18: Pantalla d'Inici de l'Aplicació

Font: Elaboració Pròpia

L'aplicació actua com un complement al lloc web de la institució, on es concentra la informació necessària perquè el coordinador de pràctiques puga realitzar les assignacions. Aquesta informació s'entrega de forma voluntària per alumnes, empreses i professors, que tenen accés als formularis on poden introduir aquestes dades.

En base a les seves necessitats principals, les distintes parts son redirigides, al iniciar sessió, a la pàgina que siga més del seu interés. En el cas dels alumnes, son redirigits al formulari per a actualitzar el seu perfil, que és la única secció a la que tenen accés:

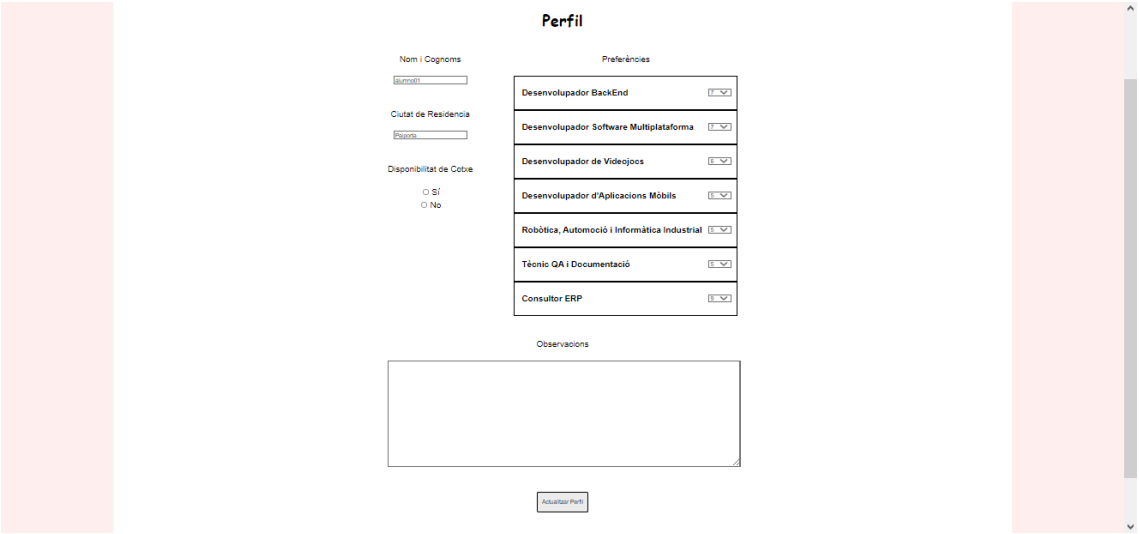

Figura 19: Perfil d'Alumne

Font: Elaboració Pròpia

Els professors ja tindran per defecte la seua informació introduïda, així que per seran redirigits a la pàgina d'Ajustos, on podran ajustar les ràtios de nombre d'alumnes per hores alliberades:

Figura 25: Pàgina d'Ajustos

Font: Elaboració Pròpia

Les empreses, per la seua part, seran redirigides al llistat de pràctiques, on poden afegir noves pràctiques o actualitzar alguna de les existents:

Figura 26: Llistat de Pràctiques

Font: Elaboració Pròpia

Finalment, els coordinadors es veuran per defecte en el panel d'assignació, perquè puguen assignar de forma manual o automàtica als alumnes i professors a pràctiques d'empresa:

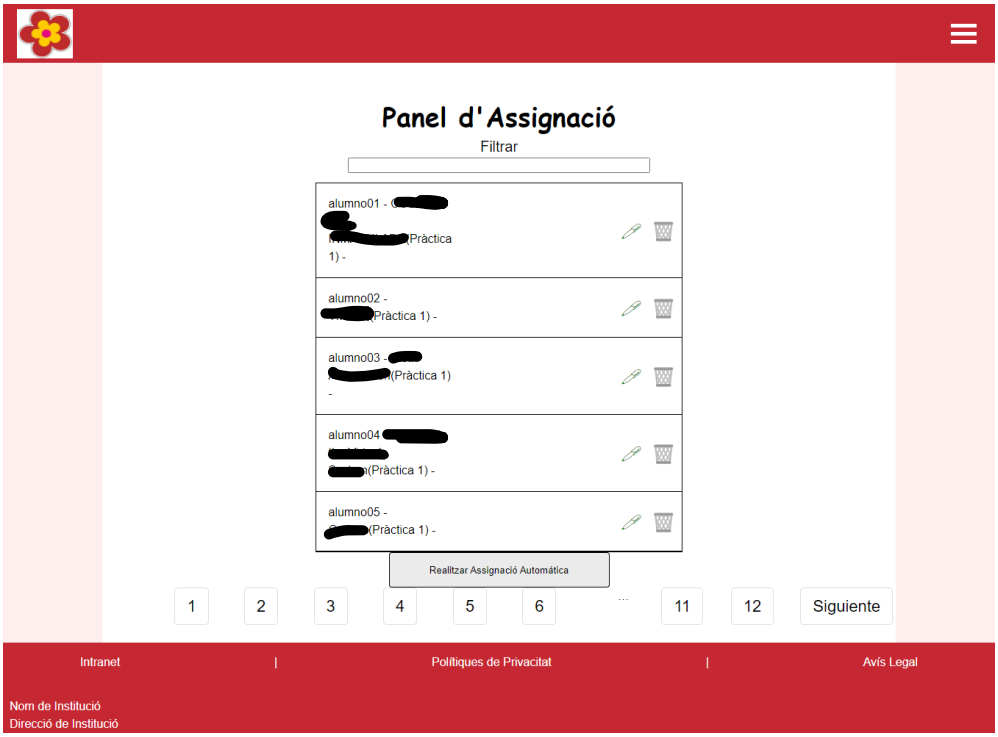

Figura 20: Panel d'Assignació

Font: Elaboració Pròpia

Addicionalment, podran consultar els llistats d'alumnes, professors i empreses i així comprovar que dispose de les dades necessàries per a realitzar l'assignació i fer les modificacions que veja convenients.

5. Validació

En aquest apartat es realitzarà una anàlisi del temps que porta l'algoritme en executar-se, desglossant-ho en: càlcul de distància, definició de variables, definició de restriccions i definició de funcions, i també s'analitzarà la solució obtinguda, tant en el que respecta la eficàcia en la minimització de distàncies com en la coherència de les assignacions.

Per a la anàlisi del temps, es prova inicialment a realitzar l'assignació de 15 alumnes, i es va incrementant de 15 a 15 per a comprovar la influencia de la quantitat d'alumnes en el temps tardat.

- **15 alumnes:** 12 minuts en general: 11 minuts per a calcular les distàncies, 0,097 segons quant a la definició de variables, 0,107 segons pel que fa a la definició de restriccions i 0,289 segons respecte a la definició de funció objectiu.

- **30 alumnes:** 25 minuts en general: 24 minuts per a calcular les distàncies, 0,250 segons quant a la definició de variables, 0,204 segons pel que fa a la definició de restriccions i 0,7 segons respecte a la definició de funció objectiu.

- **45 alumnes:** 37 minuts en general: 35 minuts per a calcular les distàncies, 0,465 segons quant a la definició de variables, 0,347 segons pel que fa a la definició de restriccions i 1 minut i 32 segons respecte a la definició de funció objectiu.

Per tant, tenim que, per cada 15 alumnes tractats, el càlcul de les distàncies puja 11 minuts i, per tant, és imprescindible l'emmagatzematge d'aquestes en cau per a agilitzar futures assignacions.

La solució del problema de minimització de distància amb el conjunt de dades emprat, és la següent:

```
alumno01-empresa01(Pràctica 1)          ->          [Paiporta - Paiporta] Distància 0.0 km
alumno02-empresa02(Pràctica 1)          ->          [València - València] Distància 0.0 km
alumno03-empresa03(Pràctica 1)          ->          [Algemesí - Algemesí] Distància 0.0 km
alumno04-empresa04(Pràctica 1)          ->          [Alaquas - Alaquas] Distància 0.0 km
alumno05-empresa05(Pràctica 1)          ->          [València - València] Distància 0.0 km
alumno06-empresa06(Pràctica 1)          ->          [Requena - Riba Roja] Distància 59.6629 km
alumno07-empresa07(Pràctica 1)          ->          [Algemesí - Fortaleny] Distància 12.055 km
alumno08-empresa08(Pràctica 4)          ->          [València - València] Distància 0.0 km
alumno09-empresa09(Pràctica 1)          ->          [València - València] Distància 0.0km
alumno10-empresa10(Pràctica 1)          ->          [Benetusser - Sedaví] Distància 2.6135 km
alumno11-empresa11(Pràctica 2)          ->          [Silla - Silla] Distància 0.0 km
alumno12-empresa12(Pràctica 1)          ->          [Benetusser - Benetusser] Distància 0.0 km
alumno13-empresa13(Pràctica 1)          ->          [Alginet - L'Alcúdia] Distància 9.9133 km
alumno14-empresa14(Pràctica 1)          ->          [Alginet - Picassent] Distància 15.867700000000001 km
alumno15-empresa15(Pràctica 2)          ->          [Catarroja - Massanassa] Distància 1.2002000000000002 km
alumno16-empresa16(Pràctica 1)          ->          [Picassent - Picassent] Distància 0.0 km
alumno17-empresa17(Pràctica 1)          ->          [Catarroja - Catarroja] Distància 0.0 km
alumno18-empresa18(Pràctica 4)          ->          [València - València] Distància 0.0 km
alumno19-empresa19(Pràctica 2)          ->          [Albal - Catarroja] Distància 2.3773 km
alumno20-empresa20(Pràctica 1)          ->          [Sollana - Almussafes] Distància 5.4293000000000005 km
alumno21-empresa21(Pràctica 3)          ->          [Carcaixent - Fortaleny] Distància 18.9595 km
alumno22-empresa22(Pràctica 2)          ->          [Albal - Alfafar] Distància 5.59 km
alumno23-empresa23(Pràctica 3)          ->          [Valencia - Valencia] Distància 0.0 km
alumno24-empresa24(Pràctica 1)          ->          [Picassent - Picassent] Distància 0.0 km
alumno25-empresa25(Pràctica 1)          ->          [València - València] Distància 0.0 km
alumno26-empresa26(Pràctica 1)          ->          [Albal - Alfafar] Distància 5.59 km
alumno27-empresa27(Pràctica 1)          ->          [Silla - Silla] Distància 0.0 km
alumno28-empresa08(Pràctica 5)          ->          [València - València] Distància 0.0 km
alumno29-empresa28(Pràctica 2)          ->          [Benetusser - Alfafar] Distància 1.8020999999999998 km
alumno30-empresa29(Pràctica 1)          ->          [Catarroja - Catarroja] Distància 0.0 km
```

Figura 21: Assignacions Resultants

Font: Elaboració Pròpia.

D'aquesta descobrim que la distància màxima que un alumne ha de recórrer per a aplegar al seu lloc de pràctiques és de quasi 60 km i la menor és de 0 km.

En el cas de l'alumne amb màxima distància, resideix en Requena, mentres que la pràctica té lloc en Riba Roja. La distància entre aquestes ciutats a peu si es va per la CV-388 es de justament 59,7 km d'acord amb Google Maps, i l'alumne 6 no disposa de cotxe, així que el càlcul de les distàncies es correcte.

Els casos amb 0 km és perquè els alumnes coincideixen en lloc de residència amb el de la empresa que ofereix la pràctica. Per exemple, l'alumne 01 i l'empresa a la que s'ha assignat resideixen en Paiporta.

Dels 30 alumnes visibles en la figura, un total de 18 han sigut assignats a una pràctica del seu lloc de residència, mentres que solament 4 han sigut assignats a una pràctica que supera els 10km de recorregut. Per tant, tenim que les solucions son factibles i que l'operació de minimització aconsegueix bons resultats.

6. Conclusions

Al llarg d'aquest document, s'ha vist com resoldre un problema d'assignació automàtica d'estudiants i professors a pràctiques d'empresa fent ús de la ferramenta OR-Tools i com, a través del micromarc de treball Flask, es pot desenvolupar una aplicació web capaç d'executar-ne de forma intuïtiva aquesta solució i d'adaptar-se als requeriments de diferents institucions. Però això, és solament el començament d'un viatge de millora perpetua.

Els algoritmes d'optimització no deixen de ser un reflex de l'afany humà d'autosuperar-se i de sentir-se realitzat, fixant-se un objectiu i lluitant fins a trobar la millor manera d'aconseguir-ho, així que, encara que els resultats hagen alcançat el nivell de qualitat esperat, quan s'afronta un problema d'aquest tipus, sempre hi ha marge de millora. S'ha reduït la càrrega de treball del coordinador de pràctiques però, al fer-ho, se l'haurà donat més marge de temps per a reflexionar sobre com aconseguir millorar la satisfacció de totes les parts involucrades i augmentar així el prestigi de la seua institució i, per tant, es més que probable que sorgisca la necessitat d'afegir més funcionalitats i d'adaptar l'algoritme a aquestes necessitats.

Per exemple, s'ha aconseguit minimitzar el desplaçament d'alumnes a les pràctiques que l'han sigut assignades, però no s'ha tingut en compte altres aspectes importants i de les que disposem informació com les preferències dels alumnes i les ferramentes amb els que aquests treballaran en les pràctiques, així com altres aspectes que l'alumne pot valorar com, per exemple, l'amabilitat dels treballadors, millors salaris o més llibertat a l'hora de desenvolupar les seues pròpies idees. Si es tenen en compte aquests detalls, es pot mirar d'assignar-li'ls a pràctiques que continguen allò que més els apassione.

Tampoc es pot deixar de costat als professors, que al ser assignats a més d'un alumne necessiten seguir una ruta per a poder atendre'ls a tots. Una bona idea seria que el model d'optimització tinga en compte aquest recorregut i que les assignacions les faça de forma que siga el més curt i directe possible.

I no solament es tracta de l'algoritme d'optimització: l'aplicació web té més a oferir per a totes les parts. Per exemple, per a incentivar als alumnes l'ús de mitjans de transport sostenibles, pot incloure aquests entre les opcions de transport i mostrar la diferència de costos i de km/h.

També pot mostrar tant a alumnes com a professors un mapa amb el recorregut que han de fer per a aplegar al seu lloc de pràctiques i donar al coordinador de pràctiques l'opció de donar pes als distints objectius implementats, així com de veure un històric d'assignacions i poder manar enquestes de satisfacció perquè dispose de la suficient informació per a pensar diverses formes de millora de cara als següents cursos.

Al ser pensada com una aplicació que pot ser emprada per distintes institucions, i tenint en compte el suposat anterior, cal esperar que es retroalimente amb les seues aportacions fins a convertir-se en una ferramenta poderosa que ajude als centres d'FP a alcançar-ne la seva meta.

7. Bibliografia

Alberola, J. M., del Val, E., Sanchez-Anguix, V., Palomares, A., & Teruel, M. D. (2016). An artificial intelligence tool for heterogeneous team formation in the classroom. *Knowledge-Based Systems*, *101*, 1–14. https://doi.org/10.1016/j.knosys.2016.02.010

Apache Software Foundation. (n.d.). Apache CouchDB. Retrieved September 7, 2022, from https://couchdb.apache.org/

Belbin, R. M. (2012). *Team roles at work*. Routledge.

Cechlárová, K., Fleiner, T., Manlove, D. F., McBride, I., & Potpinková, E. (2014). Modelling practical placement of trainee teachers to schools. *Central European Journal of Operations Research*, *23*(3), 547–562. https://doi.org/10.1007/s10100-014-0356-5

Django Software Foundation. (n.d.). Django. Retrieved August 27, 2022, from https://www.djangoproject.com/

Eiselt, H. A., & Sandblom, C.-L. (2012). *Operations research a model-based approach*. Springer Berlin Heidelberg.

Free Software Foundation (FSF). (n.d.). *GLPK (GNU linear programming kit)*. GNU Project. Retrieved August 27, 2022, from https://www.gnu.org/software/glpk/

Google. (n.d.). Angular. Retrieved August 27, 2022, from https://angular.io/

Google. (n.d.). Cloud bigtable: Nosql database service | google cloud. Google. Retrieved September 7, 2022, from https://cloud.google.com/bigtable/

Google. (n.d.). *OR-tools | google developers*. Google. Retrieved August 27, 2022, from https://developers.google.com/optimization/

Harper, P. R., de Senna, V., Vieira, I. T., & Shahani, A. K. (2005). A genetic algorithm for the project assignment problem. *Computers & Operations Research*, *32*(5), 1255–1265. https://doi.org/10.1016/j.cor.2003.11.003

IBM. (n.d.). *Cplex optimizer*. IBM. Retrieved August 27, 2022, from https://www.ibm.com/analytics/cplex-optimizer

MariaDB. (2019, November 13). *MariaDB foundation*. MariaDB.org. Retrieved September 4, 2022, from https://mariadb.org/

Microsoft. (n.d.). *Software Y aplicaciones de base de datos: Microsoft access*. Microsoft. Retrieved August 27, 2022, from https://www.microsoft.com/es-es/microsoft-365/access

MongoDB. (n.d.). *The developer Data Platform*. MongoDB. Retrieved August 27, 2022, from https://www.mongodb.com/

MySQL. (n.d.). Retrieved August 27, 2022, from https://www.mysql.com/

Pallets. (n.d.). *Bienvenido a flask¶*. Bienvenido a Flask - Documentación de Flask (2.2.x). Retrieved August 27, 2022, from https://flask-es.readthedocs.io/

Project Voldemort: A distributed database. Voldemort. (n.d.). Retrieved September 7, 2022, from https://www.project-voldemort.com/voldemort/

Pyomo. (n.d.). Retrieved September 4, 2022, from http://www.pyomo.org/

Ramos, A., Sánchez, P., Ferrer, J. M., Barquín, J., & Linares, P. (2010). Modelos matemáticos de optimización. Publicación Técnica, 1.

Rangel, D. (2015). *DynamoDB: Everything you need to know about Amazon Web Service's NoSQL database*. Amazon. Retrieved September 7, 2022, from https://aws.amazon.com/dynamodb/

Redis. (n.d.). Redis. Retrieved August 27, 2022, from https://redis.io/

Rivera, F. L. O. (2008). *Base de datos relacionales*. Itm.

Sanchez-Anguix, V., Chalumuri, R., Aydoğan, R., & Julian, V. (2019). A near pareto optimal approach to student–supervisor allocation with two sided preferences and workload balance. *Applied Soft Computing*, *76*, 1–15. https://doi.org/10.1016/j.asoc.2018.11.049

SCIP. (n.d.). Retrieved August 27, 2022, from https://www.scipopt.org/

Strauch, C., Sites, U. L. S., & Kriha, W. (2011). NoSQL databases. Lecture Notes, Stuttgart Media University, 20(24), 79.

Thiruvady, D., Morgan, K., Bedingfield, S., & Nazari, A. (2021). Allocating students to industry placements using integer programming and Ant Colony Optimisation. *Algorithms*, *14*(8), 219. https://doi.org/10.3390/a14080219

Valverde, V., Portalanza, N., & Mora, P. (2019). Análisis descriptivo de base de datos relacional y no relacional. *Revista Atlante: Cuadernos de Educación y Desarrollo*, *3*.