



UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Facultat d'Informàtica de Barcelona



CREACIÓ D'UN GENERADOR D'HORARIS AUTOMÀTIC I PERSONALITZAT

MARC CERVILLA ROVIRA

Director/a: JOAQUIM DEULOFEU AYMAR (Departament d'Organització d'Empreses)

Titulació: Grau en Enginyeria Informàtica

Especialitat: Computació

Memòria del projecte

Facultat d'Informàtica de Barcelona (FIB)

Universitat Politècnica de Catalunya (UPC) - BarcelonaTech

Resum

Aquest document tractarà sobre un treball de fi de grau basat en el desenvolupament d'un generador d'horaris. L'objectiu principal del projecte és crear una eina capaç de generar els horaris dels treballadors d'una empresa de manera automàtica i personalitzada, utilitzant tecnologies com *JavaScript*, *NodeJs* i les bases de dades de *MS SQL Server*.

El projecte intenta donar solució al problema que pateixen moltes empreses respecte a la infinitat de possibilitats existents en un horari de treball i tenint en compte totes les restriccions que comporta. Nosaltres proposem una funcionalitat en una pàgina web per tal que l'usuari pugui realitzar l'assignació horària de manera senzilla i eficient.

El treball es divideix en diverses etapes, començant per una introducció detallada, on definim l'abast i la naturalesa del projecte, així com tota la teoria relacionada. També mostrem l'etapa de planificació i gestió, que inclou un informe de sostenibilitat que analitza els aspectes ambientals, econòmics i socials del projecte. A més a més, s'estableix la planificació temporal i el pressupost d'aquest. Posteriorment, procedim a explicar la fase de desenvolupament, des del disseny del software fins a la implementació d'aquest. Addicionalment, indiquem pas a pas el procés d'implementació del codi i, un cop acabat, duem a terme certes proves per comprovar l'eficiència del sistema. En tot moment, es busca proporcionar informació clara i comprensible per al lector, intercalant figures i taules per visualitzar les explicacions. Finalment, es presenten les principals conclusions obtingudes al llarg del procés.

En definitiva, amb aquest treball de fi de grau, s'espera contribuir en l'àmbit de la planificació horària i proporcionar una eina per a les empreses interessades en gestionar eficientment els horaris dels seus treballadors.

Resumen

Este documento tratará sobre un trabajo de fin de grado basado en el desarrollo de un generador de horarios. El objetivo principal del proyecto es crear una herramienta capaz de generar los horarios de los trabajadores de una empresa de manera automática y personalizada, utilizando tecnologías como *JavaScript*, *Node.js* y bases de datos de *MS SQL Server*.

El proyecto busca dar solución al problema que muchas empresas enfrentan debido a las infinitas posibilidades existentes en la planificación horaria, teniendo en cuenta todas las restricciones que conlleva. Proponemos una funcionalidad en una página web para que el usuario pueda realizar la asignación horaria de manera sencilla y eficiente.

El trabajo se divide en varias etapas, comenzando con una introducción detallada donde se define el alcance y la naturaleza del proyecto, así como toda la teoría relacionada. También se muestra la etapa de planificación y gestión, que incluye un informe de sostenibilidad que analiza los aspectos ambientales, económicos y sociales del proyecto. Además, se establece la planificación temporal y el presupuesto del mismo. Posteriormente, se procede a explicar la fase de desarrollo, desde el diseño del software hasta su implementación. Adicionalmente, se detalla paso a paso el proceso de implementación del código y, una vez finalizado, se realizan ciertas pruebas para verificar la eficiencia del sistema. En todo momento, se busca proporcionar información clara y comprensible para el lector, intercalando figuras y tablas para visualizar las explicaciones. Finalmente, se presentan las principales conclusiones obtenidas a lo largo del proceso.

En resumen, con este trabajo de fin de grado, se espera contribuir en el ámbito de la planificación horaria y proporcionar una herramienta para las empresas interesadas en gestionar de manera eficiente los horarios de sus trabajadores.

Abstract

This document will be about a bachelor's thesis project focused on the development of a schedule generator. The main objective of the project is to create a tool capable of automatically and personalized generating schedules for employees of a company, using technologies such as JavaScript, Node.js, and MS SQL Server databases.

The project aims to provide a solution to the problem that many companies face regarding the countless possibilities involved in schedule planning, considering all the associated constraints. We propose a functionality on a web page to allow users to easily and efficiently perform schedule assignments.

The work is divided into several stages, starting with a detailed introduction where the scope and nature of the project are defined, along with all the related theory. The planning and management phase is also presented, including a sustainability report that analyzes the environmental, economic, and social aspects of the project. Additionally, the project timeline and budget are established. Subsequently, the development phase is explained, from software design to implementation. Furthermore, a step-by-step description of the code implementation process is provided, followed by certain tests to verify the system's efficiency. Throughout, the aim is to provide clear and understandable information for the reader, incorporating figures and tables to enhance explanations. Finally, the main conclusions derived from the entire process are presented.

In summary, with this bachelor's thesis, we aim to contribute to the field of schedule planning and provide a tool for companies interested in efficiently managing their workers' schedules.

Índex

1. Introducció i context	5
1.1. Introducció	5
1.2. Descripció inicial	5
1.3. Justificació	5
1.4. Abast	10
1.5. Estat de la qüestió	15
1.6. Solució proposada	27
1.7. Metodologia	29
2. Planificació	35
2.1. Introducció a la planificació	35
2.2. Fases i planificació temporal	35
2.3. Gestió dels riscos	41
2.4. Pressupost	42
2.5. Informe de sostenibilitat	47
2.6. Integració de coneixements	51
2.7. Identificació de lleis i regulacions	52
2.8. Conclusions planificació	53
3. Desenvolupament	54
3.1. Introducció	54
3.2. Disseny	54
3.3. Implementació	74
3.4. Proves	105
4. Conclusions	110
4.1. Objectius del projecte i resultats obtinguts	110
4.2. Avaluació de les limitacions i possibles millores	111
4.3. Conclusió personal i agraïments	113
5. Referències	114
6. Índex de figures	118
7. Índex de taules	119
8. Annexos	120

1. Introducció i context

1.1. Introducció

Per començar, en aquesta secció es proporcionarà una introducció completa i detallada sobre la gestió i el desenvolupament d'un Treball de Fi de Grau en l'especialitat de computació, impartit per la Facultat d'Informàtica de Barcelona.

En aquest context, s'explicarà tot el que envolta el treball des d'un inici, començant per una breu descripció inicial del projecte, els motius que van portar a la seva elecció i la rellevància del problema tractat. A continuació, s'abordarà l'abast del projecte, definint els seus objectius, requisits i els obstacles i riscos que cal contemplar per aconseguir-los. Posteriorment, es farà una revisió de l'estat de l'art i del context actual relacionat amb el treball. Finalment, s'exposarà la solució proposada i s'explicaran els detalls rellevants per al seu desenvolupament i la metodologia usada.

1.2. Descripció inicial

Abans d'inscriure el projecte, vam haver de crear una descripció breu d'aquest, es mostra a continuació:

La idea principal és desenvolupar un sistema capaç de generar un horari de treball òptim de manera automàtica i personalitzada. L'objectiu d'aquest projecte serà que qualsevol empresa, a partir d'una sèrie d'entrades i d'especificacions senzilles, pugui generar automàticament l'horari ideal per als treballadors de manera continuada en el temps i personalitzada per a les necessitats d'aquella empresa.

1.3. Justificació

1.3.1. Motivació

Des de ja fa bastants anys, com tots sabem, estem vivint en una etapa de digitalització extrema, la qual va en augment, degut als grans avenços tecnològics i a la globalització. Els programes informàtics s'han tornat una eina essencial en el nostre dia a dia, sigui quin sigui l'àmbit. Cada

vegada més persones els utilitzen per a accomplir diferents tasques, sigui per obtenir una ajuda o directament per automatitzar-les completament.

A on vull arribar amb aquest discurs? Ja fa un temps, em vaig adonar que una de les moltes tasques que desenvolupa la meva mare a la seva feina és la de gestionar els horaris de treball dels treballadors. Em va explicar que de totes les tasques a desenvolupar, la de crear els horaris de manera continuada en el temps era la que més esforç requeria, ja que era molt complicat i li havia de dedicar molt temps, "fer els horaris és com fer sudokus" em deia i amb raó.

Aquest fet em va sorprendre bastant, pel fet que després d'uns anys en una carrera d'informàtica i veient com moltes persones ja usen les eines digitals per a fer la vida una mica més fàcil, vaig pensar que realitzar els horaris d'una empresa de manera manual podia ser una gran pèrdua de temps, per culpa de l'alta complexitat. En canvi, aquella mateixa tasca, amb la tecnologia de la qual disposem avui dia, la podria resoldre un ordinador per nosaltres o, com a mínim, obtenir una gran ajuda.

En definitiva, si nosaltres poguéssim construir un sistema capaç de resoldre aquesta feina, podríem facilitar el treball de la meva mare i també de totes les persones que es trobin en la mateixa situació. A més a més, penso que aquest problema té una gran rellevància, la qual explicaré a continuació i, juntament amb el meu interès a assolir coneixements en desenvolupar algorismes d'aquest tipus en l'àmbit de la computació, han estat els motius principals per als quals he decidit escollir aquesta temàtica de projecte.

1.3.2. Problema

A continuació explicarem més en detall de què tracta el problema que intentem resoldre i els motius per als quals és rellevant.

Nurse scheduling problem

El problema de programar horaris a les infermeres, més conegut com a *nurse scheduling problem* és un problema genèric d'optimització que es basa en trobar la millor manera d'organitzar els horaris de les infermeres, assignant-les a certs torns a partir d'un conjunt de restriccions. [50]

Aquestes restriccions poden ser de dos tipus diferents:

- Restricció forta: és obligatori que es compleixi, en cas que no es pugui complir, el resultat no és una solució.

- Restricció suau: no és estrictament necessari que es compleixi i, per tant, es pot donar una solució sense que es compleixi la restricció. Tot i això, és desitjable que es compleixi i se li pot donar certa importància.

Adicionalment, aquestes restriccions poden ser molt variades i depenen sobretot de l'àmbit de treball on ens trobem i, entre altres exemples, poden incloure:

➤ **Requisits de personal:**

Depenent de la càrrega de treball, cada torn pot tenir un nombre mínim o màxim de persones. Per exemple, en un torn en hora punta on la càrrega de treball és màxima, podem restringir que com a mínim hi hagi 5 persones treballant en aquell torn.

➤ **Requisits d'habilitats:**

Un torn en específic podria ser més complex que altres i necessitar uns certs coneixements o habilitats i, en conseqüència, que només certs treballadors puguin realitzar aquesta feina. Per exemple, només la Marta o la Paula tenen coneixements en cert temari, per tant, aquest torn només el poden acatar elles.

➤ **Requisits de temps:**

Pot haver-hi certes hores o dies que el personal tingui dies festius o vacances. A més a més, hi ha restriccions d'hores màximes o hores mínimes a treballar, entre altres. Per exemple, cada persona ha de treballar com a màxim 40 hores a la setmana.

➤ **Preferències del personal:**

Ja relacionat amb la vida personal, cada persona pot tenir unes preferències específiques i totalment diferents als altres. Per exemple, la Marta prefereix no treballar els dimarts a la tarda.

En definitiva, l'objectiu d'aquest problema es tracta de trobar una solució que satisfaci totes les restriccions fortes i optimitzar que també satisfaci les màximes restriccions suaus possibles. Aquest problema genèric és el que més s'assembla al nostre problema específic que tractem en aquest projecte i del qual detallarem les seves característiques, restriccions i casos concrets més endavant, però existeix una gran varietat de problemes semblants al *nurse scheduling problem*, amb la mateixa temàtica, però amb enfocaments diferents, on varia l'entrada de dades, l'objectiu o la sortida en forma de solució. Un exemple és el *Optimal job scheduling*, el qual s'enfoca en l'assignació de treballs a una o més màquines amb certes condicions. És a dir, donats una llista de feines, normalment indivisibles, amb un temps i una data límit específics, i una llista de màquines, l'objectiu és realitzar l'assignació de manera que es compleixin els terminis i es minimitzi el temps total del procés. [52]

Com explicarem a continuació, aquesta classe de problemes tenen una gran rellevància i han sigut objectiu d'estudi durant anys, en conseqüència, s'han desenvolupat diferents mètodes i

tècniques per a resoldre'ls, des de programació lineal fins a mètodes heurístics basats en cerca local, o fins i tot d'intel·ligència artificial, entre molts d'altres. [44]

L'objectiu d'aquestes tècniques pot ser trobar una solució al problema que satisfaci totes les restriccions, o inclús, trobar la solució òptima en funció d'algun criteri concret, com la satisfacció del personal o la minimització de costos. El que està clar és que estem parlant d'un problema força complex, denominat com *NP-hard* i, en conseqüència, en la majoria dels casos no serà gens fàcil trobar una solució i molt menys l'òptima.

NP-hard:

Els problemes *NP-hard* són aquells que pertanyen a la classe NP (*Non-deterministic Polynomial time*), per als quals, encara no s'ha trobat una solució algorísmica eficient que pugui resoldre'ls. És a dir, avui dia no existeix cap algorisme que pugui resoldre aquests problemes en un temps polinòmic, mentre que si es pot verificar la solució en temps polinòmic. [54]

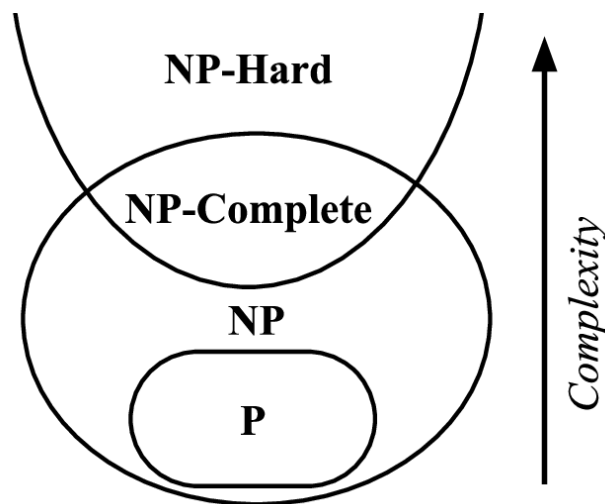


Figure 1. Diagrama d'Euler de la complexitat dels problemes en el cas $P \neq NP$

Per tant, el problema d'assignar treballadors que estem tractant en aquest projecte és *NP-hard*, ja que la seva complexitat creix exponencialment a mesura que augmenta la mida de l'entrada. Així doncs, a mesura que augmenta el nombre de treballadors i, sobretot, el nombre de torns existeixen moltes més combinacions possibles que han de ser explorades i, en un suposat cas en què l'entrada sigui un nombre elevat de torns, és molt probable que el problema no pugui acabar de trobar la solució en un temps raonable.

En termes generals i analitzant específicament el nostre problema, la complexitat és de $O(N^M)$, on N és la quantitat de torns i M el nombre de treballadors. Això és a causa del fet que, per cada torn, és possible assignar qualsevol treballador i, com a resultat, hi ha N^M

possibles solucions. Tot i això, és important tenir en compte que la complexitat real del problema serà diferent, ja que pot variar en funció de les restriccions específiques.

1.3.3. Rellevància

L'assignació d'horaris és un problema complex i comú al qual s'enfronten moltes empreses, sigui quin sigui l'àmbit. Com ja hem explicat anteriorment, es tracta d'una tasca molt complicada degut al gran nombre de restriccions i factors a tenir en compte. A qualsevol empresa o organització ens trobarem amb aquest problema o amb algun de similar i, poder trobar la solució òptima que satisfaci les necessitats tant de l'empresa com del personal, aportarà un gran benefici a les dues parts. De fet, és molt complicat trobar l'equilibri en què tothom surti guanyant, ja que no sempre es podran complir totes les condicions que desitgem i quan s'intenta prioritzar alguns criteris, és probable que d'altres es vegin perjudicats. En conseqüència, una mala assignació d'horaris podria portar conseqüències terribles, com per exemple una mala atenció al client, una disminució de la qualitat de feina o pèrdues econòmiques, entre altres.

A més a més, un cop trobes la solució a l'assignació d'horaris no et pots oblidar del tema, ja que és un problema que es presenta de manera recurrent, normalment setmanal o mensualment. Així doncs, li has de dedicar una gran quantitat de temps i esforç. Per tots aquests motius és tan complicat trobar la metodologia òptima per a resoldre aquest problema i tan important la seva investigació.

Adicionalment, tot i que no siguin característiques idèntiques, aquest problema es pot traspassar a molts àmbits i contextos diferents, a continuació descriure alguns exemples:

➤ **Assignació de vehicles:**

Aquest problema tracta d'assignar un conjunt de vehicles a diferents tasques de transport, tenint en compte restriccions com la capacitat dels vehicles, la distància a recórrer, etc.

➤ **Assignació de pacients:**

Aquest problema tracta d'assignar els pacients de qualsevol centre mèdic o residència a habitacions o llits. S'ha de tenir en compte coses com per exemple la disponibilitat dels llits o les necessitats específiques de cada pacient.

➤ **Assignació de tripulació:**

En aquest cas, es tracta d'assignar un conjunt de tripulacions a un conjunt de vols, tenint en compte la disponibilitat dels tripulants, el temps de vol, la disponibilitat de vols o les habilitats necessàries de cadascun.

En resum, el problema que tractem nosaltres en aquest projecte té moltes variants, però en general, tots aquests problemes tenen en comú assignar un seguit de recursos limitats (treballadors, vehicles, màquines, etc.) a tasques específiques tenint en compte les diferents restriccions.

Existeix un ventall molt ampli de possibilitats de com tractar aquests problemes i, per tant, considero que realitzar un software complet que intenti solucionar els diferents problemes passa a ser inclús ineficient, ja que en els diferents contextos es necessiten coses específiques diferents als altres. El que està queda clara és la importància que té, sent així un camp digne d'estudi.

1.4. Abast

1.4.1. Objectius

L'objectiu principal d'aquest projecte és implementar un sistema que permeti solucionar el problema dels horaris dels treballadors explicat anteriorment, generant una possible solució en forma d'horari de manera totalment automàtica. El que volem és que, a partir d'una entrada de dades i especificacions senzilles d'una empresa puguem aportar el millor horari possible per aquesta. A més a més, també volem que aquest sistema es pugui executar de manera continuada en el temps, ja que, com ja hem explicat, és molt probable que la tasca de generar els horaris la vulguem realitzar setmanalment o mensual.

En definitiva, el sistema que volem dissenyar haurà de complir sobretot tres característiques fonamentals per complir l'objectiu principal d'aquest projecte, que són les següents:

- Automàtic
- Personalitzat
- Continuat

Addicionalment, com a objectiu secundari complementari al principal, però també important, m'agradaria analitzar diferents algorismes i testejar diferents heurístiques per tal de trobar la millor opció en termes d'eficiència i resultats. Així doncs, podríem anar millorant el sistema per tal d'obtenir un millor rendiment o horaris més bons per als criteris de l'empresa.

Competències tècniques

En aquest treball de final de grau també es treballen unes competències tècniques específiques i escollides a l'inici del projecte. Aquestes competències agrupen els coneixements adquirits durant els estudis en Enginyeria Informàtica, en concret, de l'especialitat de computació i són també un objectiu primordial a assolir en un projecte d'aquesta naturalesa. [13]

A continuació, mostrem les competències tècniques associades:

➤ **[En profunditat] CCO2.1:**

Demostrar coneixement dels fonaments, dels paradigmes i de les tècniques pròpies dels sistemes intel·ligents, i analitzar, dissenyar i construir sistemes, serveis i aplicacions informàtiques que utilitzin aquestes tècniques en qualsevol àmbit d'aplicació.

➤ **[Bastant] CCO1.1:**

Avaluar la complexitat computacional d'un problema, conèixer estratègies algorísmiques que puguin dur a la seva resolució, i recomanar, desenvolupar i implementar la que garanteixi el millor rendiment d'acord amb els requisits establerts.

➤ **[Una mica] CCO2.2:**

Capacitat per a adquirir, obtenir, formalitzar i representar el coneixement humà d'una forma computable per a la resolució de problemes mitjançant un sistema informàtic en qualsevol àmbit d'aplicació, particularment en els que estan relacionats amb aspectes de computació, percepció i actuació en ambients o entorns intel·ligents.

1.4.2. Requisits

La definició dels requisits és una part essencial en qualsevol projecte de software. Poder analitzar i identificar quins són els requisits d'aquest projecte ens permetrà establir les característiques que ha de complir el software que volem desenvolupar i, així, assegurar que més endavant compleixi les expectatives. Hi ha dos tipus de requisits, els funcionals i els no funcionals i, a continuació, detallarem els requisits corresponents al desenvolupament del programa generador d'horaris que proposem. [18]

Requisits funcionals:

Els requisits funcionals són aquells que descriuen les funcionalitats que, si o si, el programa ha de complir. Si no és així i el sistema no ho compleix, aquest probablement fallarà. Aquests requisits normalment són definits en termes d'entrades, processos i sortides del programa per tal d'aconseguir els objectius.

En primer lloc, comentarem les funcionalitats fonamentals que ha de tenir el nostre sistema, les quals ja hem comentat als objectius i considerem que són les més importants. Les afegeixo també en aquest apartat de requisits funcionals, ja que són aspectes que el nostre software ha de realitzar obligatòriament, però també podrien col·locar-se en l'apartat de requisits no funcionals, perquè depèn de com s'enfoqui també podrien considerar-se com propietats que ha de tenir el programa.

➤ **Automàtic.**

Un cop ja s'ha realitzat l'entrada de dades, el programa ha de ser capaç d'executar-se automàticament només prement un botó, sense que l'usuari hagi de fer cap tasca manual, a part de la ja comentada entrada de dades.

➤ **Personalitzat.**

Cada empresa aportarà unes dades diferents segons els horaris que vulgui obtenir i, per tant, el sistema crearà un horari personalitzat per a aquella empresa.

➤ **Continuat.**

Com ja hem comentat anteriorment, aquesta tasca de creació de l'horari de treball s'haurà de fer de manera continuada en el temps, en conseqüència, una funcionalitat molt important que ha de tenir aquest sistema, és que es pugui executar de manera senzilla i continuada en el temps, per exemple, setmanalment.

Aquests són els altres requisits funcionals que hem de considerar en el nostre projecte d'assignació d'horaris:

➤ **Entrada dels treballadors.**

El software ha de ser capaç de rebre la informació dels treballadors de manera ordenada i consistent. Aquesta informació és la necessària per a poder realitzar l'assignació de torns correctament, així com el nom, el tipus de torns, les hores setmanals, etc.

➤ **Entrada dels torns de treball.**

El software ha de ser capaç també d'obtenir certa informació dels torns de treball, per poder obtenir quines seran les necessitats de l'empresa respecte a les hores o dies treballats i la quantitat de recursos necessaris.

➤ **Sortida final de l'horari.**

Finalment, el software ha de proporcionar un horari complet a partir dels torns de treball que havíem obtingut a l'entrada i amb els treballadors assignats als diferents torns segons es consideri a l'algorisme.

➤ **Gestió de la informació.**

També és molt important, ja que el programa s'utilitzarà de manera continuada, que puguem gestionar tota la informació relativa als treballadors i als torns, podent modificar-la quan sigui necessari i emmagatzemant-la per a una millor comoditat i eficiència.

➤ **Restriccions.**

El software ha de ser capaç d'aplicar un conjunt de restriccions per tal de poder generar un horari que sigui compatible amb les necessitats del negoci.

➤ **Visualització de l'horari**

El software ha de proporcionar una visualització en format de taula setmanal per tal que l'usuari pugui interaccionar-hi de manera intuïtiva.

Requisits no funcionals:

També és important definir els requisits no funcionals del projecte, que són aquells que descriuen algunes característiques que no estan relacionades amb les funcionalitats o serveis que ha de realitzar l'aplicació, sinó amb aspectes genèrics, propietats o qualitats amb les quals el producte final ha de comptar.

A continuació, mostrem els requisits no funcionals que hem de tenir en compte en aquest projecte:

➤ **Eficiència.**

En aquest projecte estem tractant amb un problema *NP-Hard* que, com hem explicat anteriorment, no es pot solucionar en un temps polinòmic. Per tant, un dels requisits primordials és aconseguir que el software tingui una complexitat computacional el més eficient possible, permetent així, obtenir resultats en un temps raonable.

➤ **Seguretat.**

El nostre sistema ha de ser segur, ja que tractarem amb informació privada dels treballadors d'una empresa i, en conseqüència, és molt rellevant protegir aquesta informació i prevenir que no pugui ser accedida.

➤ **Utilitat.**

L'objectiu principal del projecte és que l'usuari pugui solucionar el problema dels horaris de la seva empresa i, per tant, volem que sigui el més útil possible per aquest.

➤ **Usabilitat**

El sistema també ha de ser fàcil d'utilitzar i d'entendre per a l'usuari i, a través d'una interfície intuïtiva i no molt complicada, ha de ser senzill interactuar amb l'aplicació.

➤ **Disponibilitat**

També volem que el sistema estigui disponible en qualsevol moment, les 24 hores del dia, perquè l'usuari el pugui utilitzar sempre que vulgui.

➤ **Escalabilitat**

El software que volem dissenyar, amb el pas del temps, s'anirà desenvolupant i millorant, ja que, realment, pot tenir un gran ventall de possibilitats pel que fa a la complexitat, des d'un algorisme senzill fins a un gran sistema molt més complicat. Per tant, volem que sigui un sistema escalable, de fàcil adaptació en cas que en un futur volem afegir millores i crear noves versions.

1.4.3. Obstacles i riscos

Durant la realització del projecte ens trobem en diferents situacions i és molt probable que ens trobem amb alguns obstacles o prenguem alguns riscos. Així doncs, des d'un inici, a part dels objectius i els requisits, també vam definir uns obstacles i riscos que ens podíem trobar pel camí, els expliquem a continuació:

Obstacles:

➤ **La inexperiència**

Com a autor d'aquest treball, en un inici no tinc massa coneixements en l'àmbit de la planificació d'horaris i, per tant, això em pot obstaculitzar i, al principi, tenir certes dificultats per comprendre les restriccions específiques necessàries per a poder generar els horaris òptims. A més a més, tot i tenir certa experiència en programació en els estudis, mai he utilitzat la programació web o certs llenguatges de programació, que en cas d'utilitzar-los, hauré d'aprendre i obtenir aquests coneixements posant-hi un esforç suplementari respecte altres persones que ja podrien tenir experiència en aquests àmbits.

➤ **Gestió del temps**

Gestionar el temps en qualsevol treball és tan important com difícil, ja que sempre poden passar imprevistos o tenir tasques que s'expandeixen més del que pensaves en un principi, generant així més obstacles dels previstos.

➤ **Informació poc precisa**

Per la naturalesa d'aquest projecte, es necessita que des d'un inici s'aconsegueixi informació detallada i a la vegada que sigui molt senzilla d'introduir sobre els treballadors o els torns. Si aquesta informació no és del tot precisa, serà molt difícil crear un bon horari de treball.

➤ **Complexitat del problema**

Com ja hem comentat amb anterioritat, el problema de generació dels horaris és *NP-hard*. Això suposa un obstacle extra, ja que és molt probable trobar-nos amb casos on el problema no té solució o tarda molt temps a resoldre's. En conseqüència, si no es troba un algorisme eficient en termes computacionals, no podrem obtenir la solució desitjada en un temps raonable.

➤ **La quantitat de dades**

El fet que puguem treballar amb la quantitat de dades que vulguem es pot veure com un avantatge o com un desavantatge, ja que com més dades, més alta és la probabilitat de fer un sistema el màxim d'eficient, però a la vegada, hi haurà moltes més possibilitats a triar i, inclús, podria generar ineficiència del sistema per un ús excessiu d'aquestes,

així que s'haurà de tenir en compte si volem usar poques dades de manera senzilla o utilitzar una gran quantitat d'aquestes.

Riscos:

➤ **Regulacions laborals**

Els horaris d'una empresa té un seguit de regulacions que s'han de complir, per tant, en aquest projecte tenim el risc de l'incompliment d'aquestes regulacions laborals, la qual cosa portaria, en conseqüència, multes o sancions econòmiques per a l'empresa.

➤ **Insatisfacció dels treballadors**

Realitzar un horari equilibrat intentant prioritzar diferents aspectes és molt complicat, ja que si intentes prioritzar certa característica, altres en sortiran perjudicades. Per tant, tenim un risc molt important en què podem generar horaris que siguin injustos o poc convenients per a certs treballadors, perquè tots ells tenen unes preferències i és difícil que es puguin acatar totes. Si es produeix aquesta insatisfacció, podria portar a una disminució de la productivitat, cosa que perjudicaria tant als empleats com a l'empresa.

➤ **Problemes tècnics**

No podem descartar que el software pugui fallar en un cert moment i haurem de córrer el risc que puguin ocórrer interrupcions en el sistema, com errors de connexió o de servidor, o el risc de la pèrdua de dades. Per tant, serà necessari prendre mesures de seguretat per a minimitzar aquests riscos.

1.5. Estat de la qüestió

L'objectiu d'aquest apartat consisteix a obtenir informació sobre l'estat de la qüestió, més conegut en anglès com *state of the art*, que és una part crucial de qualsevol projecte i on es tracta d'investigar i analitzar la situació de la temàtica d'aquest, incloent-hi els antecedents i el context històric, les tecnologies i mètodes que existeixen en l'actualitat, així com els mercats similars. Això ens ajudarà a identificar possibles solucions al problema que tractem i a assolir una visió completa de les tecnologies i les possibilitats que existeixen en l'àmbit de la realització d'horaris.

1.5.1. Antecedents i context històric

Realment, les empreses porten existint des de fa molt temps i, per tant, l'assignació dels seus treballadors als torns de feina, també. Tot i això, i depenent molt del sector, hi ha empreses que

aquest problema no els ha afectat tant, o bé perquè l'assignació de torns és molt sistemàtica i sempre és la mateixa, o bé perquè no els hi preocupa tant les preferències dels treballadors, etc. En canvi, n'hi ha d'altres, que per altres raons com per exemple la falta de recursos o tenir uns torns de treball complexos, és un tema rellevant que es vol tenir en compte i, sumat al creixement exponencial del poder computacional, ha agafat una gran importància en els darrers anys. Els encarregats de recursos humans cada vegada es preocupen més per tenir uns horaris òptims, prioritzant tant als treballadors com a l'empresa per tal d'obtenir el màxim benefici.

En definitiva, el problema de l'elaboració i planificació d'horaris, en un sentit molt més ampli que el d'aquest projecte, ha estat objecte d'investigació durant dècades i, com veurem a continuació, els avenços en tecnologia i informàtica han permès el desenvolupament de propostes i solucions ben variades.

Des de la dècada de 1970, s'han publicat diversos estudis centrats en casos pràctics en l'elaboració dels horaris i l'anàlisi sistemàtic d'aquests. Aquest és el cas de l'article d'investigació '*Nurse staffing, scheduling, and reallocation in the hospital*' publicat en 1976 per D. Warner. [8] En aquest article, defineix diferents criteris per a tenir en compte en l'assignació horària:

- Cobertura: en què es diferencien el nombre de persones requerides per una tasca al nombre de persones assignades a aquesta?
- Qualitat: quina és la durada de la jornada laboral? Els horaris són justos?
- Estabilitat: com està dissenyat l'horari quant a la consistència i als dies festius o caps de setmana?
- Flexibilitat: està l'horari ben adaptat en cas que hi hagi algun canvi o imprevist?
- Cost: quins són els recursos, humans o materials, que s'utilitzen?

En les dècades posteriors fins a l'actualitat, el domini de totes aquestes investigacions s'ha ampliat, ja que les empreses s'han interessat en aquesta capacitat de crear solucions horàries més eficients i personalitzades per tal de millorar la productivitat i reduir costos. En aquests últims anys s'ha desenvolupat un camp d'investigació en què, utilitzant casos de prova reals, sobretot en el cas de les infermeres, han aparegut noves tècniques d'optimització i metaheurístiques¹.

A continuació mostro alguns exemples d'aquestes investigacions més recents:

- '*An electromagnetic meta-heuristic for the nurse scheduling problem*' (Maenhout i Vanhoucke, 2007) [26]
- '*A shift sequence based approach for nurse scheduling and a new benchmark dataset*' (Brucker et al., 2010) [7]

¹ metaheurística: procediment que busca trobar solucions aproximades per a problemes computacionals generals, que són difícils de resoldre amb algorismes exactes. A diferència d'altres mètodes, no garanteixen trobar de la millor solució, però ofereixen solucions pròximes a la òptima en un temps raonable. [metaheurística]

- ‘A systematic two phase approach for the nurse rostering problem’ (Valouxis et al., 2012) [48]

La majoria d’aquestes investigacions parlen de mètodes matemàtics que treballen amb una funció objectiu, és a dir, optimitzen un cert criteri. En els primers articles, els investigadors intenten desenvolupar models lineals per a solucionar el problema, però s’adonen que l’assignació d’horaris es pot tornar un tema molt complex segons el cas real que li passis. En conseqüència, més endavant es comencen a utilitzar diferents mètodes heurístics per a solucionar aquests problemes de casos reals més complicats, on ja es comença a parlar d’intel·ligència artificial, *constraint programming* (programació de restriccions) i de sistemes experts.

Està clar que hi ha hagut grans avenços i noves propostes per tractar de solucionar el problema dels horaris, tot i això, encara queden molts conceptes a abordar, com per exemple la complexitat de les dades, la variabilitat dels requisits per part de treballadors o empreses o la necessitat d’obtenir solucions personalitzades per un gran ventall de sectors diferenciats. Per això pensem que és important continuar investigant i desenvolupant solucions per als problemes actuals i futurs en aquest context.

1.5.2. Mètodes i tecnologies

Actualment, existeixen diversos mètodes i tecnologies per a solucionar el problema d’assignació d’horaris en diferents contextos. Com ja hem vist, existeixen gran varietat d’articles científics que estudien aquest àmbit i proposen diverses solucions per optimitzar aquest procés de planificació horària, cadascuna amb els seus avantatges i inconvenients. A continuació, us mostrem un resum de les tècniques més conegudes, no entrarem molt en detall, ja que cada una té una àmplia àrea d’estudi. [49]

Optimització matemàtica:

L’optimització matemàtica és un mètode molt popular per resoldre problemes matemàtics que permet trobar la solució òptima minimitzant o maximitzant una funció objectiu, o funció d’avaluació, a partir d’uns possibles valors d’entrada. És a dir, l’optimització matemàtica intenta donar resposta a problemes amb el següent esquema:

$$\begin{aligned} & \max(\min) f(x) \\ & x \in \Omega \subseteq \mathbb{R}^n \end{aligned}$$

On la x equival a un vector, l’expressió $f(x)$ és la funció objectiu que volem optimitzar i la x ha d’estar dins de les restriccions que dona el problema, o bé, pertànyer al conjunt de decisions factibles (Ω). [55]

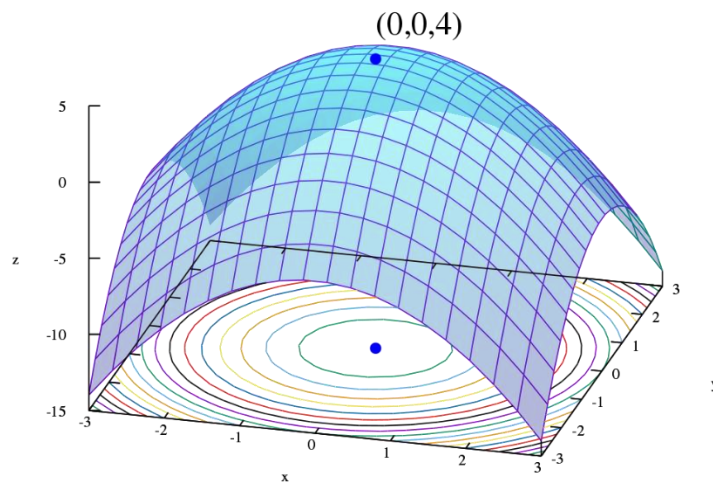


Figure 2. Gràfic d'una funció objectiu amb un màxim global en el punt $(0,0,4)$

Així doncs, aquest mètode també pot ser utilitzat en el problema dels horaris, on podem definir objectius com minimitzar el cost total dels horaris, maximitzar l'eficiència o altres opcions que nosaltres vulguem. Per exemple, en el cas més habitual on l'objectiu sigui minimitzar el cost, en primer lloc, s'haurà de formular el problema que tractem en un model matemàtic, definint les variables, la funció objectiu i les restriccions. En un cas senzill podríem definir les variables com el nombre d'hores treballades per cada treballador, l'assignació de torns de cada treballador i el cost total de l'horari, el qual voldrem minimitzar a partir de la funció objectiu, tenint en compte restriccions com la disponibilitat dels treballadors.

Un cop definit el model matemàtic, existeixen diversos algorismes d'optimització matemàtica, com per exemple, la programació lineal, utilitzada quan les variables són contínues, o la programació lineal entera, que s'utilitza quan les variables són valors discrets.

En definitiva, l'optimització matemàtica pot ser un recurs molt important en certs problemes, ja que ens proporcionarà una solució òptima. Això té grans avantatges, però requereix una formulació matemàtica correcta i, en casos de problemes més complexos, pot tenir un cost computacional molt elevat.

Hem donat només alguns exemples dels més senzills, però hi ha moltes altres subàrees que també usen l'optimització matemàtica, com són els mètodes heurístics o la programació de restriccions, les quals considero molt rellevants en l'àmbit que estem tractant i explicarem més en detall a continuació.

Heurístiques:

Una heurística és un terme molt ampli que no només s'utilitza en la informàtica, però igualment ens agradaria mostrar la seva definició, ja que explica molt bé l'objectiu. Segons la Viquipèdia es defineix de la següent manera:

“Una heurística, o tècnica heurística, és qualsevol proposta per resoldre problemes que utilitza un mètode pràctic, tot i que no està garantit que sigui òptim, perfecte o racional, però és suficient per assolir un objectiu immediat, a curt termini o una aproximació.” [56]

En definitiva, i ja en el context on ens trobem nosaltres, es poden utilitzar molts mètodes heurístics diferents per al problema dels horaris, ja que la majoria de casos són problemes complicats que no tenim clar si tenen una solució òptima, o ni tan sols si tenen una solució. Aquests mètodes són intuïtius i pràctics i n'hi ha de ben variats segons els objectius que tinguem, però en general, tots intenten trobar una solució relativament acceptable entre totes les possibles, ja que s'usen sobretot en casos on hi ha moltes possibles combinacions.

En general, s'utilitzen algorismes de cerca, on tindrem un conjunt de nodes o estats, els quals poden ser una possible solució parcial o total al problema (també pot ser que no hi hagi solució). Aquests nodes s'hauran d'explorar per mirar de trobar la millor solució possible seguint certs criteris, minimitzant-los o maximitzant-los en la funció objectiu, que comentàvem en l'apartat anterior.

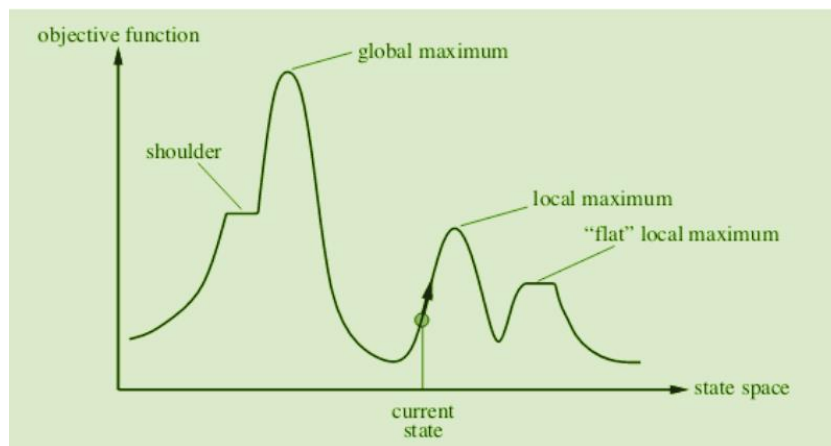


Figure 3. Gràfic on podem veure els diferents estats segons una funció objectiu

Com s'observa a la figura 3 i com hem comentant, la millor solució seria el màxim global, però si l'espai de tots els estats és molt gran, potser és millor quedar-nos amb una possible solució que és el màxim local. A continuació, mostrem algunes de les heurístiques més utilitzades:

En primer lloc, tenim els algorismes d'exploració més senzills, que no utilitzen cap informació específica i simplement recorren l'espai de solucions de manera sistemàtica i són la cerca en

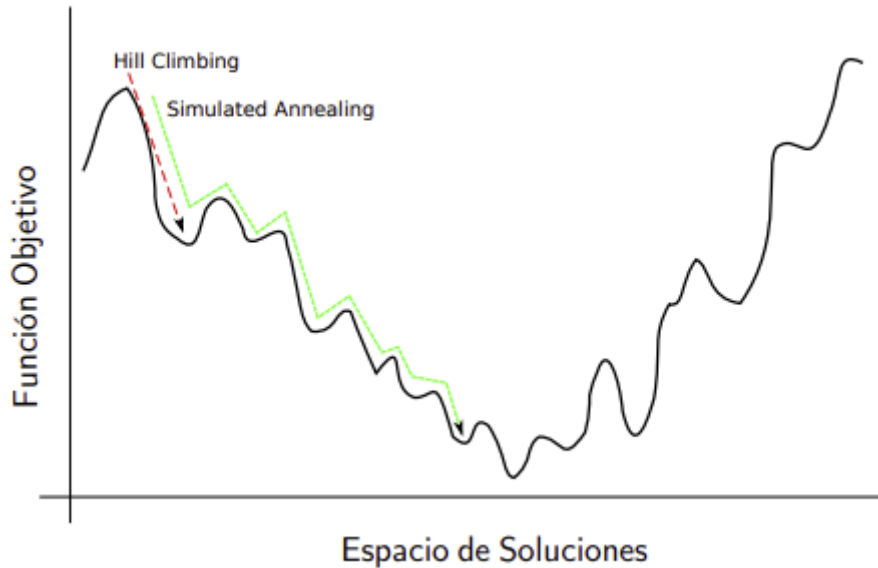


Figure 5. Gràfic per visualitzar com actuen en l'espai de solucions els algorismes de 'Hill Climbing' i 'Simulated Annealing'

Aquests algorismes que hem vist són només alguns exemples explicats de manera molt breu, però n'hi ha de molts altres, cadascun amb les seves fortaleses i debilitats. L'objectiu serà trobar un algorisme que satisfaci els nostres desitjos pel que fa a la qualitat de la solució, però també quant al temps computacional.

Programació de restriccions:

La programació de restriccions, més conegut en anglès com *constraint programming*, és una tècnica de programació utilitzada per resoldre problemes que involucren restriccions, limitacions o regles i trobar una solució que les satisfaci. Així doncs, aquests problemes s'anomenen *CSPs* i normalment es defineixen de la següent manera:

Donats:

- un conjunt X de variables: $X = \{x_1, x_2, \dots, x_n\}$
- un conjunt D de dominis: $D = \{d_1, d_2, \dots, d_n\}$
- un conjunt C de restriccions: $C = \{c_1, c_2, \dots, c_n\}$

Haurem de trobar una assignació de valors ($x_1 \mapsto v_1, \dots, x_n \mapsto v_n$), que compleixi els dominis $v_i \in D$ i satisfaci totes les restriccions de C . [39]

En aquest tipus de problema, doncs, existeixen un gran nombre de combinacions possibles, com també és el cas del problema d'assignació d'horaris amb moltes restriccions que s'han de complir, per tant, aquest mètode ens pot ser molt útil per tal de trobar les millors solucions. El

cas bàsic que es fa servir sempre per comprendre i exemplificar aquesta classe de problemes és el Sudoku.

En la majoria dels casos, els *CSPs* es converteixen en problemes de satisfacció booleana, anomenats SAT, que pertanyen a la classe NP-complet. En aquesta conversió, les variables passen a tenir un domini binari, amb valor cert o fals. Per explicar-ho de manera resumida, aquests problemes es solucionen a través de *SAT Solvers*, els quals reben una gran fórmula booleana constituïda per algunes més petites, cadascuna de les quals representa una restricció del problema original. Llavors, utilitzant altres tècniques com el *backtracking* o la propagació, el *SAT Solver* explora diferents combinacions de valors fins que troba una combinació que satisfà la fórmula booleana gran. En el cas que no es pugui trobar una combinació que satisfaci la fórmula, aleshores el problema és insatisfactible. En tot cas, aquests ens ofereixen trobar una possible solució en un temps bastant raonable tot i tenir moltes variables i restriccions, així que l'haurém de tenir bastant en compte en aquest projecte.

Intel·ligència artificial:

La intel·ligència artificial és tot un món a part, ja que porta molts anys investigant-se en molts sectors i ocupa una gran quantitat de temari que no acabariem mai, tot i això, encara és un sector molt recent i li queda un gran recorregut en el futur. La intel·ligència artificial (IA) és una branca de la informàtica que tracta d'imitar la intel·ligència humana perquè una màquina pugui desenvolupar tasques complicades de manera autònoma. La idea és que aquests sistemes de IA puguin prendre decisions i aprendre a mesura que interactuen amb un entorn. La IA té un gran nombre de tècniques i aplicacions, però en aquest cas ens centrarem en la seva utilització per a la planificació d'horaris. [53]

Així doncs, la IA és una eina molt útil per a resoldre problemes complexos que els humans tenen dificultats per resoldre, com podria ser l'assignació dels treballadors. Un popular mètode que s'utilitza en aquest cas pot ser l'aprenentatge automàtic, més conegut en anglès com *machine learning*. Els algorismes d'aquest s'usen per realitzar prediccions a través d'una gran quantitat de dades per tal d'entrenar el sistema i es podrien utilitzar, per exemple, per predir quins treballadors haurien de dur a terme certes tasques, quines tasques necessiten més o menys recursos, en quin moment del dia és millor realitzar certa activitat, i un llarg etcètera.

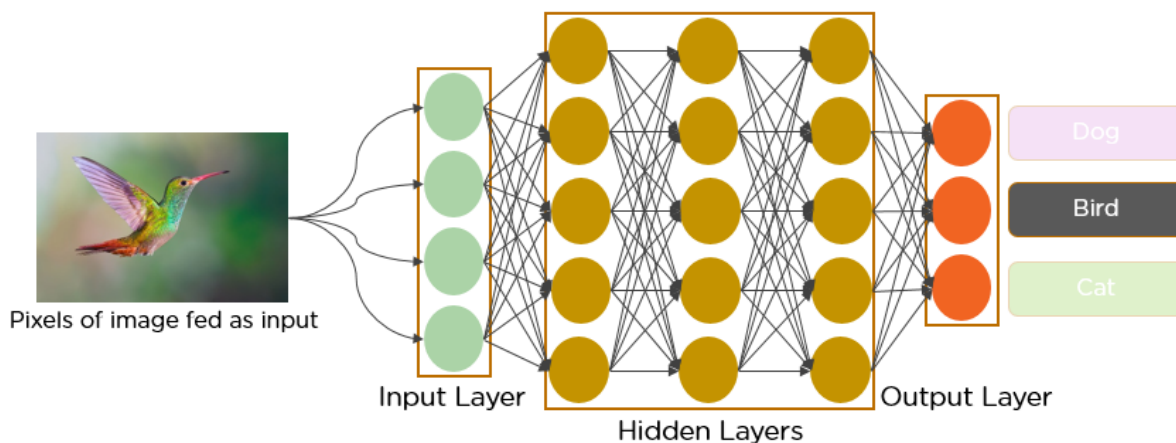


Figure 6. Model simple d'una xarxa neuronal utilitzada en l'aprenentatge automàtic de reconeixement d'animals

L'altre mètode popular més utilitzat són els sistemes experts. Aquests són uns sistemes complexos basats en una sèrie de regles integrades i, a partir de les quals, prenen decisions sobre un problema en concret, a més a més, van guanyant experiència. En el cas de l'assignació horària, a partir dels sistemes experts podríem determinar per exemple quins treballadors estan disponibles per treballar o quines tasques s'han de realitzar primer.

Així doncs, existeixen moltes maneres de resoldre aquest problema, inclús es pot combinar la IA amb les tècniques d'optimització matemàtica. Hi ha un gran ventall de possibilitats, des de la més bàsica a la més complexa i cadascuna amb els seus avantatges i inconvenients.

1.5.3. Mercats similars

Com ja hem vist, l'assignació d'horaris ha estat objecte d'estudi de molts investigadors al llarg dels anys i té una gran rellevància a escala mundial. En conseqüència, sembla lògic pensar que ja existiran moltes solucions al mercat actual per a tractar amb aquest problema. En aquest apartat, investigarem l'existència d'alternatives i solucions per al problema que fem referència. En molts casos, aquests softwares no només treballen el nostre problema, sinó que tenen diverses funcionalitats per a tenir més exposició al mercat i demanda, a continuació, mostrarem alguns softwares especialitzats en la generació d'horaris i quines característiques tenen.

Softwares de gestió d'horaris:

En primer lloc, quan busques informació respecte a l'assignació horària, ens trobem que la majoria de les solucions del mercat referent a aquest tema, ens aporten certes ajudes a l'hora de realitzar els horaris, però no ens solucionen el problema proposat en aquest projecte, ja que els

horaris o bé s'han de continuar fent de manera manual, o bé no ofereixen un ús pràctic de les restriccions.

Els casos més senzills els trobem amb gent que utilitza l'*Excel* o *Google Calendar* per a crear els horaris, que l'única ajuda que ens suposa és en proporcionar eines interactives per a fer-los de manera fàcil, però tot continua sent manual i no ens aporta cap solució interessant.

Altres solucions com, per exemple, *Trello*, *Asana* o *Monday*, ofereixen funcionalitats per planificar tasques i assignar-les a membres de l'equip, però no estan especialitzades en la programació d'horaris amb restriccions.

En definitiva, existeixen multitud d'aplicacions que, tot i que és cert que aporten ajudes per a construir els horaris d'una empresa, com per exemple, monitorar les hores treballades dels empleats, assignar grups de treball segons les categories o planificar quins recursos materials, humans i temporals tenen les diferents tasques, continuen sense aportar-nos una solució especialitzada a resoldre de manera eficient i automàtica el problema tractat en aquest projecte.

Skello:

El software més interessant que he trobat és *Skello*, també he trobat altres de similars, com *Connecteam* o *Wrike*, però aquest em sembla el més complet, ja que ofereix moltes alternatives i ajudes perquè les empreses puguin gestionar els horaris dels treballadors.

Skello és un software de planificació de recursos humans en format web que ofereix moltes alternatives i ajudes a les empreses perquè puguin gestionar els seus equips, crear els horaris dels treballadors i comunicar-se amb ells, a més d'altres funcionalitats de gestió del negoci, però que ja no tenen a veure amb el problema que plantegem nosaltres, sinó que són un extra. [43]

Alguns exemples d'aquestes funcionalitats, en relació amb el nostre problema són:

- Creació i visualització d'horaris
- Conservar els torns més freqüents
- Duplicar horaris
- Comptabilitzar les hores treballades, dies de vacances, etc.
- Comunicar els horaris als treballadors

Tenen diferents plans amb diferents funcionalitats, des del bàsic fins al prèmium. En el seu millor pla, el prèmium, parlen d'una solució molt interessant: una tecnologia automatitzada anomenada *Smart Planner* que, a partir dels teus recursos i necessitats, t'assisteix per a crear els horaris en un sol clic. Aquest sistema funciona amb intel·ligència artificial i comenten que tenint en compte les regles de productivitat del negoci i els contractes legals dels treballadors, *Smart Planner* t'optimitza els horaris de manera que s'adaptin a les teves necessitats. [47] Per

altra banda, sembla que aquesta funcionalitat està limitada a 10 horaris, com expliquen en la comparativa dels plans.

Com es pot observar a la figura 7, aquesta aplicació ofereix una interfície d'usuari bastant senzilla i fàcil d'entendre, moltes funcionalitats de gran utilitat i, en general, un software bastant complet de gestió de personal. Tot i això, sembla ser un dels software més cars, amb un cost de 89.00 € al mes, preu que depèn sobretot de quin pla i quines característiques hagi demanat. [41]

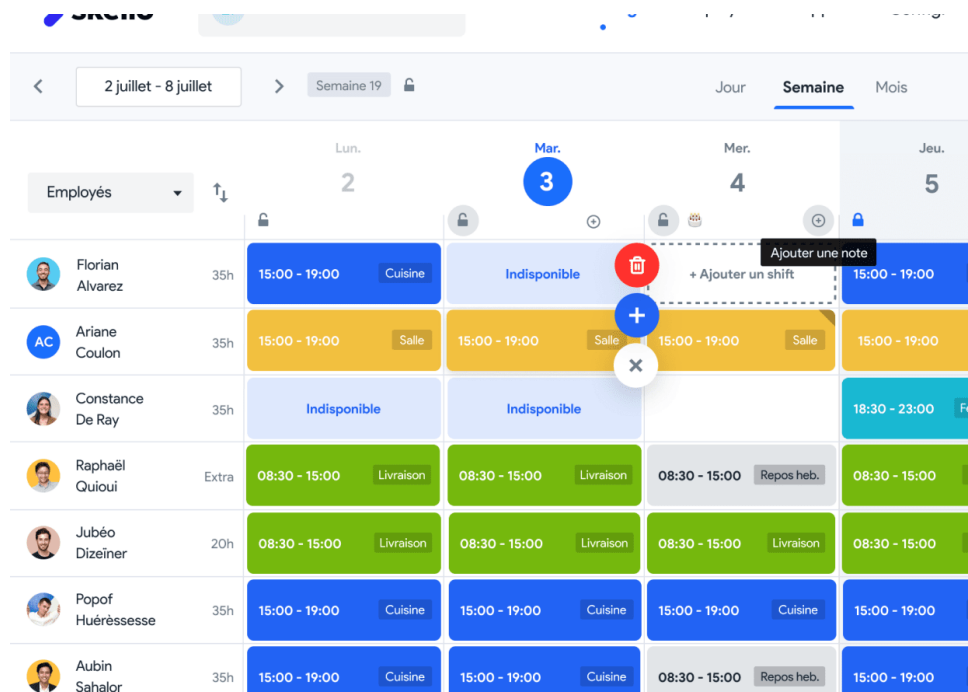


Figure 7. Interfície gràfica d'un horari en el software Skello

Rotageek:

També ens agradaria mencionar el programari que ofereix l'empresa *Rotageek* que, tot i no ser tan completa com l'anterior proposta, també ofereix solucions de gran utilitat per gestionar als treballadors. La funció principal és justament l'objectiu d'aquest projecte, un generador d'horaris automatitzat (*autoscheduling*) que, a través d'algorismes avançats d'intel·ligència artificial, aconsegueix proporcionar una solució per optimitzar els horaris dels treballadors i cobrir les necessitats de personal de l'empresa. [1]

El seu funcionament de cara a l'usuari és simple, en primer lloc, s'ha d'ingressar informació sobre els treballadors, les seves habilitats i preferències, així com la seva disponibilitat i, a continuació, com es pot veure a la seva pàgina web, a través d'un botó podrem generar un horari de forma automàtica que s'ajusti a les necessitats d'empresa i treballadors.



Figure 8. Característiques que té en compte Rotageek per realitzar horaris automàtics.

Adicionalment, altres característiques que ofereix l'aplicació són:

- Flexibilitat: els treballadors poden ingressar la seva pròpia disponibilitat i preferències.
- Comunicació: responsables i treballadors es poden comunicar fàcilment respecte als canvis que hi hagi als horaris.
- Anàlisi de dades: ofereix informes i anàlisis per tal d'ajudar a les empreses millorar l'eficiència.

Respecte als preus, *Rotageek* ofereix diferents solucions personalitzades segons les necessitats de l'empresa, però no mostra els preus. Així doncs, l'empresa interessada haurà de contactar directament amb ells per a obtenir informació dels costos.

Per concloure aquest apartat, aquest és un petit recull d'algunes aplicacions molt útils per solucionar el nostre problema, però n'hi ha d'altres com per exemple *Humanity* o *When I Work* que aporten funcionalitats i solucions similars a les anteriors i, per tant, ja no ens estendrem en detall.

En tot cas, en aquest apartat s'ha redactat la teoria i la informació que ens proporcionen aquestes empreses, és a dir, per treure conclusions finals satisfactòries, s'haurien de reclamar demostracions a aquestes empreses i veure si de veritat aporten solucions en casos reals i pràctics.

1.6. Solució proposada

1.6.1. Solució i context

Un cop vistes la descripció i justificació del projecte, el seu abast i l'estat de la qüestió, ja tenim tota la informació necessària per comprendre la naturalesa del problema, així que, finalment, presentarem la nostra proposta de solució.

Com s'ha comentat en [l'apartat 1.5.3](#), ja existeixen solucions al mercat molt competitives i completes, sistemes complexos de gestió de recursos humans realitzats per empreses amb alts pressupostos i personal especialitzat, amb les quals serà impossible competir per part nostre.

Tornant al context inicial, la motivació principal d'aquest projecte és poder ajudar a la meua mare o a persones en situacions similars. Així doncs, des d'un principi em vaig posar en contacte amb diverses persones i vam dur a terme una reunió per tractar la viabilitat d'aquest projecte. En aquesta reunió, a part de mi i la meua mare, hi havia també en Toni Vila, l'encarregat de l'empresa on treballa, *Cal Forner*, que es tracta d'un forn de pa, pastisseria i cafeteria, i en Jordi, l'encarregat de *Hit Systems*, l'empresa que li dona suport informàtic. Aquesta empresa es dedica a aportar solucions informàtiques a empreses d'àmbit comercial com forns de pa i cafeteries i, com que una d'aquestes solucions era la de realitzar el control de personal i el nostre objectiu era comú, vam decidir col·laborar durant la realització d'aquest projecte. Actualment, les empreses del sector de cafeteries i forns de pa ja tenen un sistema existent que els hi aporta diverses funcionalitats per a la gestió dels recursos humans. De fet, en la web de Hit Systems, ja hi ha un apartat on es pot dur a terme els horaris de l'empresa, però de manera manual i sense cap altre ajuda.

En definitiva, la proposta final d'aquest projecte tracta en implementar i adaptar el generador automàtic d'horaris del qual hem discutit, dins del sistema ja existent que ens ofereix Hit Systems, permetent així una nova funcionalitat per a poder generar els horaris del personal d'aquestes empreses de manera eficient i automàtica. Així doncs, per tal de solucionar i millorar aquest problema, aquest projecte es desenvolupa en l'entorn d'aquesta empresa, utilitzant les tecnologies *JavaScript* i *NodeJs* per a fer l'algorisme, ja que són la manera més senzilla i compatible per treballar en aquest entorn. Els beneficis d'utilitzar aquest entorn de treball són notoris, perquè podem treballar en una web i amb una base de dades ja dissenyades en un context molt específic. En comparació amb els complexos sistemes del mercat actual, nosaltres tindrem l'avantatge d'usar grans quantitats de dades que ells no disposen i treballar en un sector molt específic, permetent que l'algorisme prioritzi petits detalls que beneficiïn aquestes empreses del sector, ja que com més àmbits intentes agrupar, més complicat és de solucionar el mateix problema per tothom. Finalment, si tot funciona correctament podrem afegir la funcionalitat desitjada a la seva pàgina web.

1.6.2. Actors implicats

En aquest projecte hi ha diversos actors implicats, els quals detallem a continuació:

➤ **Director/s.**

És l'encarregat d'assessorar a l'estudiant i fer el seguiment del TFG, així com avaluar les fites que li corresponguin. En aquest cas és en Joaquim Deulofeu Aymar, del Departament d'Organització d'Empreses de la UPC.

➤ **Desenvolupador/programador.**

Aquest projecte disposa únicament d'un desenvolupador, que a la vegada és el programador. És l'autor d'aquest treball i treballa en la planificació, investigació, documentació, codi i tot lo referent al projecte. En aquest cas és en Marc Cervilla Rovira, estudiant de Computació a la FIB.

➤ **Empresa 1. (usuària)**

És aquella empresa o empreses que li interessa gestionar de la forma més eficient possible als seus treballadors per tal d'obtenir màxim benefici. En aquest cas és Cal Forner, junt amb el seu encarregat, en Toni Vila.

➤ **Persones de RRHH.**

És la persona o persones que treballa en l'empresa 1 i l'encarregada de gestionar els Recursos Humans d'aquesta. És l'usuari i client final del producte que proposem en aquest projecte, la persona que utilitzarà el sistema per a realitzar els horaris dels treballadors de l'empresa 1. En aquest cas és la meva mare, Rosa Rovira, però en un cas general, pot ser qualsevol persona de RRHH.

➤ **Empresa 2. (software)**

És aquella empresa que ofereix els recursos i solucions informàtiques a diferents negocis com l'empresa 1, proposant un software amb varies funcionalitats per a la gestió d'aquestes. En aquest cas és Hit Systems, junt amb el seu encarregat, en Jordi Bosch, també actuant com a director del projecte ajudant-me a dirigir el rumb del treball.

1.7. Metodologia

1.7.1. Descripció general

Aquest projecte s'ha desenvolupat majoritàriament des de casa en el meu ordinador de sobretaula i dedicant el màxim d'hores lliures per tal d'obtenir els millors resultats. Tot i això, alguns dies en específic he treballat fora de casa, per exemple en l'oficina de Hit Systems, on he utilitzat el meu ordinador portàtil per també aprofitar i poder avançar quan estava fora de casa.

Respecte a la metodologia, he usat el mètode de cascada o *waterfall*. Aquest és un mètode de gestió de projectes, que es divideix en diferents fases i s'enfoca en realitzar cadascuna d'aquestes de manera seqüencial i lineal, completant cada etapa del projecte abans d'avançar a la següent. [46]

Per tant, cadascuna d'aquestes fases en les quals es divideix té els seus propis objectius i resultats. En primer lloc, es realitza una anàlisi exhaustiva dels requisits del projecte, incloent-hi els objectius, els requisits funcionals i no funcionals i els obstacles i riscos, els quals ja hem mostrat en [l'apartat 1.4](#). A continuació, es procedeix a dissenyar una solució que s'adeqüi als requisits anteriors i, una vegada dissenyat, es comença a implementar la solució utilitzant les tecnologies adequades. Finalment, arribem a l'última etapa obligatòria que tindrà aquest projecte, on es realitzaran les proves necessàries per a garantir que la solució ja implementada compleixi amb els requisits inicials. Si durant aquestes proves es troben problemes, s'ha de tornar enrere fins a la fase de disseny i fer els canvis que facin falta. Aquestes serien les fases que contemplem en aquest projecte, tot i això, en la metodologia cascada existeixen dues fases més, la d'implantació i la de manteniment, on el producte ja sortiria al mercat per a l'ús real. Aquestes fases, actualment, no les contemplem en aquest projecte i s'haurà de veure si, un cop finalitzat i si tot va bé, l'empresa de Hit Systems les vol aplicar al mercat real.

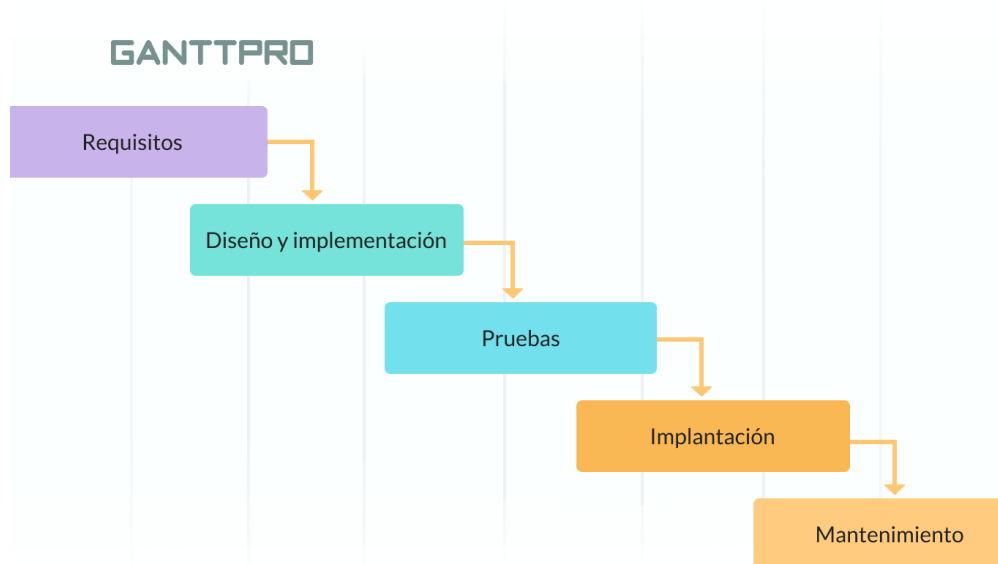


Figure 9. Diagrama de les fases de la metodologia cascada

Així doncs, aquesta metodologia té els seus avantatges i inconvenients. Els principals avantatges són que és un model molt senzill on és fàcil fer-te una idea de com evoluciona el projecte. És un sistema de fases molt rígid i específic i, en conseqüència, saps perfectament quines tasques has de realitzar i et pots marcar objectius fàcilment. Per tant, és un mètode que s'ha d'utilitzar sempre i quan els requisits estiguin ben definits i no estiguin subjectes a canvis, és aquí on trobem el seu desavantatge, en cas que s'hagin de canviar els requisits del projecte, s'haurà de tornar a començar de nou tot el procés.

En aquesta metodologia, doncs, es poden patir canvis importants que t'obliguin a tornar al pas anterior. En el cas d'aquest projecte, en un principi es va realitzar un disseny en concret. La idea de disseny original era dur a terme una web completa que complís l'objectiu principal de fer els horaris automàticament i s'assolissin els diferents requisits funcionals i no funcionals, com per exemple, que et demanés les dades dels treballadors, et mostres visualment els horaris, etc. Més endavant, durant la fase d'implementació, que és la més complexa i llarga, he utilitzat una metodologia més específica que es basa a ficar-se petits objectius en la implementació de codi de manera realista. També a mesura que s'avançava i havíem de prendre decisions rellevants, feia reunions amb en Jordi, l'encarregat de Hit Systems i decidíem el camí a recórrer.

En definitiva, en una d'aquestes reunions es va decidir canviar el disseny inicial. Com ja he comentat, la idea inicial era fer tant el *frontend* com el *backend* d'una web amb *HTML*, *CSS* i *NodeJS*. També s'usava *VSCode* per a l'edició de codi i *MongoDB Compass* per a tenir una base de dades no relacional de prova, a part de molts paquets i *frameworks* diferents, com seria el cas de *Bootstrap*. No entraré molt en detall d'aquestes tecnologies, ja que finalment es va canviar el disseny i, en el següent apartat, detallarem les que s'han utilitzat per al projecte final. Més enllà d'això, els objectius i requisits seguien sent els mateixos i només vam haver de repetir la fase de disseny i implementació. El nou disseny, doncs, s'ha basat a implementar la

funcionalitat dels horaris a la web ja existent de Hit Systems, com hem detallat en [l'apartat 1.6](#) de la solució proposada, usant un nou entorn de treball del qual descriurem les eines i tecnologies utilitzades en el següent apartat.

1.7.2. Eines i tecnologies

AWS Cloud9

Com diuen des de la mateixa web d'*Amazon: Amazon Web Services (AWS)* és la plataforma en el núvol més adoptada i completa del món, la qual ofereix més de 200 serveis integrals de centres de dades en l'àmbit global. [35] És a dir, *Amazon* proporciona un ventall molt ampli de serveis allotjats a internet per a empreses i programadors. Alguns exemples d'aquests serveis poden ser emmagatzematge en bases de dades, algorismes d'anàlisis o intel·ligència artificial, però hi ha molts d'altres. En la figura 10 es poden observar alguns dels serveis més importants que ofereixen, que sobretot van destinats a empreses.



Figure 10. Conjunt de serveis més importants que ofereix l'AWS

Concretament, el servei que hem elegit per aquest projecte és Cloud9, un entorn de desenvolupament integrat (IDE) basat en el núvol que permet escriure, executar i depurar codi a través del navegador. [20] En un principi vaig començar utilitzant el *Visual Studio Code*, però els motius per als quals finalment hem elegit aquesta plataforma de desenvolupament són variats, ja que ofereix molts avantatges. En primer lloc, és un entorn de treball molt fàcil d'utilitzar i en el qual tens diferents eines per tal d'escriure, testejar i depurar codi, a través dels seus interpretes de llenguatge, depuradors i terminals. A més a més, en ser un sistema basat en el núvol i no en un entorn local, es pot accedir des de qualsevol lloc a través d'un navegador, cosa que m'ha facilitat poder treballar tant des de casa com des de fora amb l'ordinador portàtil.

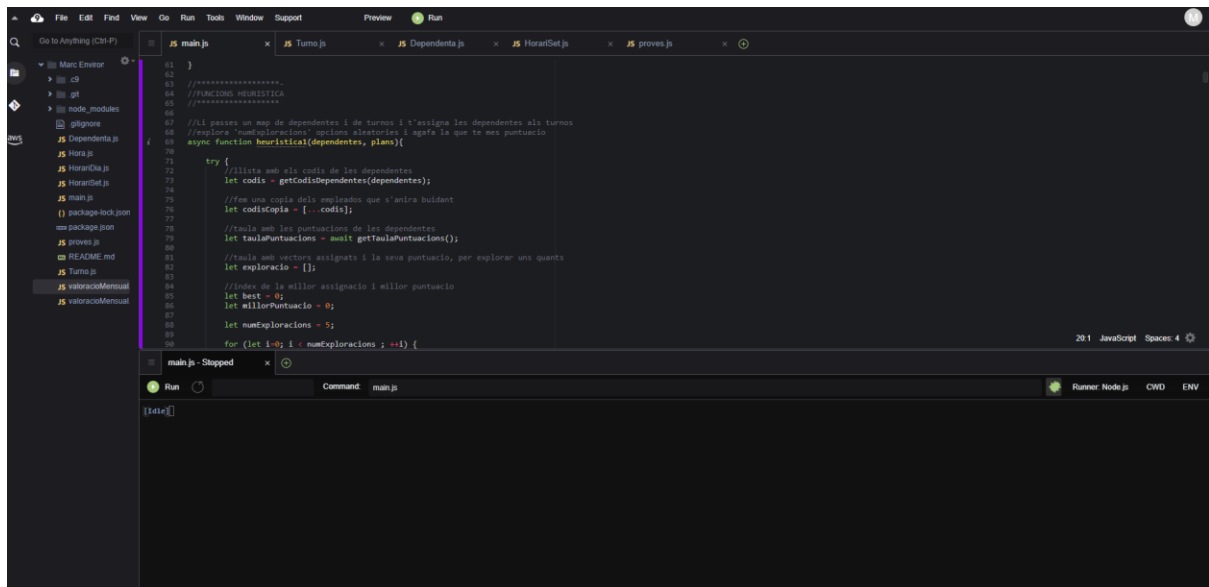


Figure 11. Captura de pantalla del entorn de treball on s'ha desenvolupat aquest projecte.

Per últim, el factor més important per al qual hem utilitzat aquest IDE és que ens ha permès treballar amb la base de dades de l'empresa col·laboradora Hit Systems i, en conseqüència, hem pogut desplegar una solució més eficient i efectiva per complir els nostres objectius.

JavaScript

El llenguatge de programació utilitzat per a la implementació de codi d'aquest projecte és *JavaScript*. Aquest és un llenguatge de programació interpretat molt popular actualment que s'utilitza sobretot en el desenvolupament d'aplicacions web i del costat del client, tot i que cada vegada s'utilitza més també al costat del servidor. [36] Per tant, una de les principals característiques és que permet implementar funcions complexes en pàgines web, que és precisament la solució que hem proposat per aquest projecte i el motiu, però no l'únic, per al qual hem triat aquest llenguatge.

Durant la fase d'implementació es va dubtar entre dos llenguatges, JavaScript o Python, dels quals no tenia experiència prèvia en cap dels dos, però són els dos llenguatges més utilitzats en la programació web i els més preferibles per aquesta classe de projecte. Tot i això, si tenia experiència en llenguatges orientats a objectes, com per exemple C++, i tant JavaScript com Python són llenguatges de script amb diversos paradigmes de programació com ara l'orientació a objectes, cosa que ens facilita la creació i manteniment del codi. Ambdós llenguatges són molt populars i hi ha una gran quantitat de recursos, documentació i llibreries per a la utilització i aprenentatge.

Python també té els seus avantatges, ja que també es pot utilitzar tant al costat del client o del servidor. El seu avantatge més important, però, és la seva corba d'aprenentatge, ja que és un

llenguatge molt simple i fàcil d'aprendre que ofereix aquesta simplicitat per a realitzar i llegir codis complexos, cosa que altres llenguatges no poden oferir. [40]

Respecte a la velocitat i rendiment, Python és la millor solució per processar grans quantitats de dades, mentre que JavaScript millora en projectes on l'usuari interactua amb el programa a temps real. Pel que fa a la popularitat, JavaScript és el llenguatge més utilitzat i el 98% de webs l'utilitzen, però també és cert que en els darrers anys, com mostren alguns estudis, Python està modificant la tendència i en un futur pot ser el més utilitzat. [28]

Així doncs, un cop analitzades els seus avantatges, desavantatges i característiques principals, finalment vam decidir utilitzar JavaScript, sobretot per al fet que l'objectiu principal del projecte és que l'usuari pugui interaccionar i aconseguir l'horari a través de la web, per exemple amb l'entrada de dades o a partir d'un botó, i aquest és un àmbit on surt clarament guanyador. A més a més, JavaScript ens ofereix aquesta flexibilitat per treballar tant pel costat del client com del servidor, gràcies a NodeJS que explicarem a continuació, cosa que necessitem per a obtenir les dades de la base de dades. Tot i això, cal dir que les dues opcions eren bones i haguessin sigut igual de vàlides.

NodeJS

NodeJS és un entorn d'execució de JavaScript basat en el motor V8 de Google Chrome. Això significa que permet executar codi de JavaScript fora del navegador, pel costat del servidor, per la qual cosa és una opció molt popular per a la creació de servidors web. NodeJS es basa en una arquitectura orientada a objectes i no bloquejant, és a dir, el codi es segueix executant en cas que rebí sol·licituds com per exemple l'entrada i sortida de dades i permet que el codi sigui més escalable i eficient. [22]

Com ja hem comentat, nosaltres en aquest projecte requerim de certes funcionalitats que apliquen tant pel costat del client com el de servidor, ja que volem que s'executi una funció al prémer un botó, però també volem accedir a la base de dades del servidor. En conseqüència la utilització de NodeJS és estrictament necessària.

Amb tot això, NodeJS també compta amb un sistema de gestió de paquets conegut per npm (Node Package Manager). Aquest ofereix moltes eines i biblioteques de codi obert que faciliten la creació d'aplicacions per als programadors. En concret, nosaltres hem utilitzat els següents paquets: 'mssql', una llibreria per a poder utilitzar Microsoft SQL Server en NodeJS [30] i 'logic-solver', una llibreria que t'aporta funcions per a poder implementar un problema de satisfacció booleana i les seves restriccions. [29]

Microsoft SQL Server

Com ja hem comentat, hem necessitat la instal·lació d'un paquet npm anomenat 'mssql' per a poder fer servir Microsoft SQL Server. Aquest és un sistema de base de dades relacional desenvolupat per l'empresa Microsoft i s'utilitza per administrar i emmagatzemar dades d'aplicacions o a nivell empresarial. [11]

Respecte al nostre projecte, hem usat MS SQL Server degut a la ja existent base de dades de Hit Systems, que utilitza aquest sistema i ens ha donat permisos, fet que ens ha facilitat poder aplicar el projecte en una estructura empresarial real. Així doncs, hem pogut realitzar les consultes a la base de dades a través del popular llenguatge de consulta estructurada SQL, el llenguatge estàndard utilitzat per a accedir i manipular les bases de dades, sobretot nosaltres l'utilitzarem per llegir i modificar-les.

Git i GitHub

Altres tecnologies que hem usat per al desenvolupament d'aquest projecte, concretament per la part d'implementació del codi, han estat Git i GitHub, les quals van molt relacionades. Per una banda, tenim git, un sistema de control de versions (VCS) utilitzat per guardar diferents versions d'un projecte o document. Això facilita la gestió del projecte, ja que a través dels anomenats repositoris pots tornar enrere i recuperar una versió anterior o també pots visualitzar els canvis de les noves versions respecte a les anteriors. Per altra banda tenim a GitHub, la plataforma web més popular on els usuaris poden allotjar i visualitzar els seus repositoris de Git. [10]

És cert que aquesta tecnologia normalment s'utilitza en projectes de més gran escala on hi ha diversos col·laboradors i no és el cas d'aquest projecte amb només un desenvolupador. Tot i això, Git i GitHub ens han servit de gran ajuda en aquest treball, ja que hem pogut mantenir un control més organitzat respecte les versions del projecte, sabent en tot moment quins canvis proporcionen cada versió, permetent-nos tornar a versions anteriors en cas que fos necessari i també mantenint el projecte al núvol i no només en local, augmentant la protecció i seguretat d'aquest.

Microsoft Word

La última tecnologia utilitzada, però no menys important, és Microsoft Word, el conegut software de processament de text, que ens ha permès la redacció i emmagatzematge de diferents documents essencials per al desenvolupament d'aquest projecte. Concretament, en la part més teòrica del projecte, com és la redacció de la gestió del projecte, el disseny inicial, el seguiment dels objectius i apunts necessaris durant la implementació del codi i, per últim i més rellevant, la construcció d'aquesta memòria final.

Altres

Per últim, també hem utilitzat altres tecnologies diverses com per exemple, PowerPoint per a realitzar la presentació o el navegador Google Chrome, per tota la cerca d'informació a internet.

2. Planificació

2.1. Introducció a la planificació

Un cop ja hem vist tot l'apartat d'introducció, ara ja entenem de què tracta i quin és l'abast i el context d'aquest projecte de manera detallada. En aquesta nova secció, veurem tota la informació referent a la planificació del treball. Aquesta planificació es va dur a terme en els inicis i era una part molt important a l'hora de gestionar tot el projecte, tot i això, és probable que degut a les diferents situacions que et trobes, aquesta s'hagi vist alterada.

A continuació, mostrarem les diferents etapes del projecte, la descripció de les tasques, la planificació temporal d'aquestes i altres aspectes relacionats amb la preparació.

2.2. Fases i planificació temporal

2.2.1. Extensió temporal

Per tal de realitzar la planificació temporal d'aquest projecte, en primer lloc, hem de calcular les hores que li haurem de dedicar segons els crèdits ECTS. El projecte consta d'un total de 18 ECTS i es calcula que cada crèdit és, aproximadament, una dedicació de 25 hores. Dels 18 crèdits, 3 es destinen a la gestió de projectes (GEP). És a dir, les hores que li dedicaré al projecte seran unes 75 hores a la gestió de projectes i 375 hores al treball en si.

El projecte es durà a terme durant un període d'aproximadament cinc mesos, començant a l'octubre i fins al febrer i durant aquest temps es realitzaran totes les tasques i objectius.

També cal tenir en compte que aquest projecte té un caire bastant ampli que s'haurà d'anar concretant amb el temps, és a dir, és molt probable que la planificació que faci en un inici es

vegi trastocada. En conseqüència, durant les reunions i amb el pas del temps, caldrà analitzar com s'està complint la planificació i plantejar si s'han de fer les modificacions respectives.

2.2.2. Fases del projecte

En primer lloc, definirem les diferents fases genèriques que tindrà el projecte. Com ja hem dit, utilitzarem una metodologia de cascada, en la qual hi haurà diferents dependències i no es començarà una fase sense haver finalitzat l'anterior. Tot i això, veurem que hi ha algunes tasques més independents que si es poden dur a terme paral·lelament.

Les fases del projecte són les següents:

- Gestió de projectes
- Documentació treball
- Reunions de seguiment
- Estudi previ
- Disseny
- Implementació
- Proves

2.2.3. Descripció de les tasques

En aquest apartat, es concretarà de què tracta cada tasca dins de les fases creades anteriorment. També mencionarem, per cada tasca, quins són els recursos utilitzats respecte a les tecnologies i eines de la metodologia, esmentades en [l'apartat 1.7.2](#). Respecte als recursos humans i l'espai de treball seran els mateixos en totes les tasques i, per tant, no els especificaré, ja que aquest projecte el desenvoluparé jo sol i, majoritàriament, des del meu ordinador personal.

A continuació, us mostrem la llista de fases amb les seves corresponents tasques, abreviades en sigles per a identificar-les en els següents apartats:

GP – Gestió de projectes

- *GP1. Contextualització i abast*
Definir l'abast del projecte en el context del seu estudi.
Recursos: Microsoft Word
- *GP2. Planificació temporal*
Planificar les tasques que es desenvoluparan en el projecte i quin temps ocuparan.
Recursos: Microsoft Word

- *GP3. Pressupost i sostenibilitat*
Estimació del pressupost del projecte i fer l'informe de sostenibilitat del qual.
Recursos: Microsoft Word
- *GP4. Document final gestió*
Fer el document de síntesi final de la gestió del projecte agrupant les anteriors tasques.
Recursos: Microsoft Word

DT – Documentació treball

- *DT1. Memòria*
Realització de la memòria final del treball. Aquesta s'anirà desenvolupant de manera paral·lela a mesura que avança el projecte.
Recursos: Microsoft Word
- *DT2. Presentació*
Un cop finalitzat tot el projecte, faltirà preparar-se la presentació per exposar-lo al tribunal.
Recursos: PowerPoint

RS – Reunions seguiment

Aquesta fase del projecte és única i no es divideix en tasques. Seran un seguit de reunions de seguiment que es faran paral·lelament al projecte cada cert temps per tal d'observar el procés del projecte i marcar-se els següents objectius parcials. En aquesta tasca també participaran el director del TFG, en Joaquim, i en Jordi, l'encarregat de Hit Systems.

EP – Estudi previ

- *EP1. Estudi del problema i elecció dels objectius i requisits*
Estudiar detalladament quin és el problema que ens interessa solucionar, analitzant en quin estat d'investigació es troba i quines podrien ser les propostes de solució. També caldrà escollir quins són els objectius i requisits necessaris per a solucionar-lo.
Recursos: Google Chrome
- *EP2. Anàlisi de les possibles tècniques*
Una de les tasques més importants pel desenvolupament del projecte serà analitzar quines possibilitats reals existeixen per tal de crear el sistema desitjat de la manera més

eficient. És a dir, analitzar tots els softwares i llenguatges de programació possibles i escollir quin és el més adequat per assolir els nostres objectius.

Recursos: Google Chrome

- *EP3. Estudi funcionament tècnica escollida*

Aquesta tasca serà, òbviament, dependent de l'anterior, ja que un cop haguem analitzat i escollit amb quina tècnica treballarem, caldrà fer un estudi més profund d'aquesta per tal d'aprendre el funcionament i adquirir el coneixement necessari per desenvolupar el projecte.

Recursos: Google Chrome

DI – Disseny

- *DI1. Disseny sistema general*

En aquesta tasca caldrà dissenyar un sistema que compleixi amb els requisits i objectius proposats en l'estudi previ a través de la tecnologia escollida.

Recursos: Microsoft Word, JavaScript

- *DI2. Disseny entrada i sortida de dades*

Un cop tinguem un esquema general, pensarem quina és la millor manera de tractar les dades, les que haurà d'entrar l'usuari, les que haurà de treure el programa i les dades ja existents a la base de dades.

Recursos: Microsoft Word, JavaScript, MS SQL Server

- *DI3. Disseny heurístiques*

Per finalitzar la fase de disseny, caldrà pensar en els mètodes o heurístiques específiques que s'utilitzaran per a generar l'horari.

Recursos: Microsoft Word, JavaScript

IM – Implementació

- *IM1. Implementació del codi base*

Després del disseny dels diferents camps dels quals estarà format el sistema, caldrà generar el codi bàsic que ho fiqui tot en funcionament. Per tant, de seguida que tinguem fetes les tasques de disseny, procedirem a la implementació de codi, la tasca més complexa i que segurament ens ocuparà més temps.

Recursos: Cloud9, JavaScript, NodeJS, SQL Server, Git

- *IM2. Implementació algorismes i possibles millores*

Mentre realitzem la implementació del codi base, també haurem d'anar implementant les diferents heurístiques, com a mínim una, que treballaran la generació d'horaris prèviament dissenyades. A més a més, a mesura que avanci la creació de codi, anirem

pas a pas, complint els objectius parcials que ens fiquem i afegint totes les possibles millores que puguem.

Recursos: Cloud9, JavaScript, NodeJS, SQL Server, Git

PF – Proves i comprovacions finals

- *PF1. Depuració*

Un cop fet el disseny i la implementació del sistema ja estarem dins de la fase final del projecte, on caldrà comprovar el bon funcionament d'aquest. És a dir, serà una fase de test on ficarem a prova el nostre sistema per tal que no doni errors inesperats.

Recursos: Cloud9, JavaScript, NodeJS

- *PF2. Anàlisis i conclusions*

Un cop tinguem el nostre projecte acabat, caldrà veure si realment compleix tots els objectius que volíem en un principi i extreure les conclusions pertinents.

Recursos: Microsoft Word

2.2.4. Dependències

Les dependències entre les fases i les tasques individuals esmentades anteriorment són les següents:

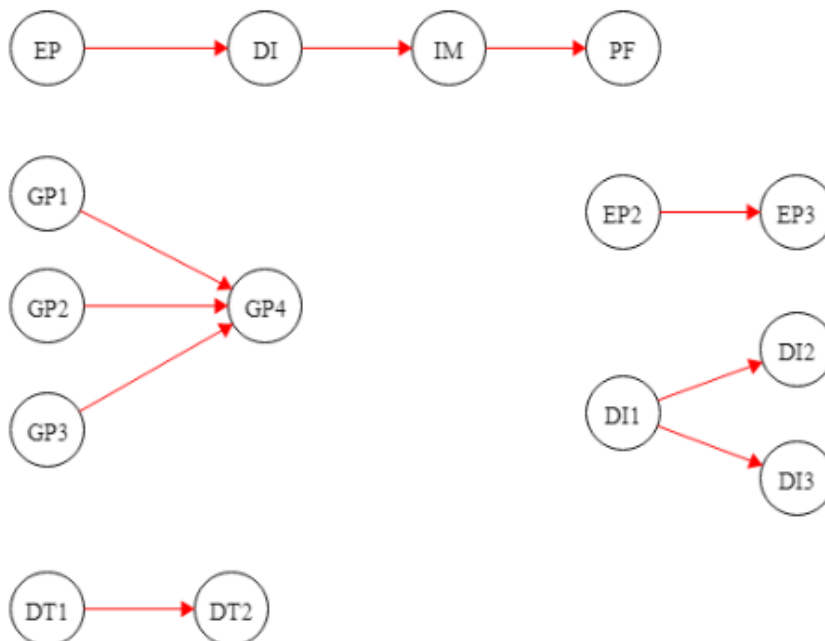


Figure 12. Dependències entre les tasques

2.2.5. Taula resum

A continuació, us mostrem una taula on poder veure de manera senzilla les diferents fases i tasques desglossades del treball amb el seu respectiu temps estimat. En primer lloc, hem calculat el temps en hores i, a partir d'aquest, hem calculat també el temps aproximat en dies, en un cas genèric que li dediquem unes 3 hores cada dia durant tot el període dels cinc mesos.

TASCA	TEMPS (hores)	TEMPS (dies)
GP	75	25
<ul style="list-style-type: none"> • GP1 • GP2 • GP3 • GP4 	20 15 15 25	6.6 5 5 8.3
DT	70	23.3
<ul style="list-style-type: none"> • DT1 • DT2 	50 20	16.6 6.6
RS	10	3.3
EP	40	13.3
<ul style="list-style-type: none"> • EP1 • EP2 • EP3 	10 10 20	3.3 3.3 6.6
DI	55	18.3
<ul style="list-style-type: none"> • DI1 • DI2 • DI3 	20 15 20	6.6 5 6.6
IM	170	56.6
<ul style="list-style-type: none"> • IM1 • IM2 	100 70	33.3 23.3
PF	30	10
<ul style="list-style-type: none"> • PF1 • PF2 	20 10	6.6 3.3
	TOTAL: 450	150

Taula 1. Distribució del temps per tasca

2.2.6. Diagrama de Gantt

El diagrama de Gantt representa de forma gràfica la planificació explicada anteriorment, on podem veure perfectament l'ús de la metodologia cascada. S'ha creat amb l'eina *Office Timeline* [31] i el podem observar en [l'apartat 8](#) d'annexos.

2.3. Gestió dels riscos

En aquest apartat, veurem quina ha estat la planificació del projecte respecte als riscos que ens puguem trobar pel camí, que ja hem identificat i analitzat en [l'apartat 1.4.3](#).

<i>Risc</i>	Probabilitat ocurrencia	Impacte
<i>Regulacions laborals</i>	Mitjana	Molt alt
<i>Insatisfacció treballadors</i>	Alta	Alt
<i>Problemes tècnics</i>	Baixa	Mitjà

Taula 2. Resum dels riscos amb la seva probabilitat d'ocurrencia i impacte.

➤ **Regulacions laborals**

Aquest risc es refereix a la possibilitat que el nostre sistema generador d'horaris, obtingui resultats que incompleixin les regulacions laborals de la mateixa empresa o les lleis establertes en el país on s'utilitzi.

La probabilitat d'ocurrencia és mitjana, tot i que depèn molt de l'abast del projecte i que tan complex s'acabi fent el sistema. Per altra banda, l'impacte que pot tenir si s'acaba donant el cas és molt elevat, ja que aquest incompliment podria resultar en multes o sancions econòmiques per a l'empresa.

Per tal d'evitar aquest risc, ens haurem de comunicar amb els responsables de recursos humans i que ens expliquin les restriccions i regulacions a tenir en compte amb l'objectiu de complir-les.

➤ **Insatisfacció dels treballadors**

En aquest cas, patim el risc que, un cop el nostre algorisme hagi generat un horari de manera automàtica, els treballadors es poden sentir insatisfets amb els resultats d'aquest, pel fet que no s'ajusta a les seves necessitats o preferències.

La probabilitat que passi això és alta, ja que en casos d'empreses on hi hagi molts treballadors o on hi hagi recursos limitats, és molt probable que no es puguin complir les necessitats de tots. Això també tindria un impacte elevat, ja que podria disminuir la productivitat i motivació dels treballadors, generant un mal ambient de treball.

Amb l'objectiu de minimitzar aquest risc, haurem de crear un algorisme que sigui equilibrat, en el que totes les preferències contin per igual i no sempre surtin els mateixos perjudicats i haurem d'obtenir de la manera més eficient possible les preferències dels diferents treballadors, per exemple preguntant quin dia volen tenir lliure o quin torn de treball prefereixen.

➤ **Problemes tècnics**

En qualsevol projecte de desenvolupament d'un software poden aparèixer els problemes tècnics, com errors de programació, errors del hardware o qualsevol caiguda de les tecnologies utilitzades.

Tot i que no es pot saber, crec que la probabilitat d'ocurrència és baixa, ja que en condicions normals tot hauria de funcionar correctament. Tot i això, mai se sap el que pot passar i suposaria un impacte mitjà, ja que, tot i que depèn de quin tipus de problema tècnic sigui, podria endarrerir el desenvolupament del projecte en cas ens afecti a nosaltres, o causar una gran insatisfacció, en cas que afecti a l'usuari final.

Alguns d'aquests problemes tècnics poden no dependre de nosaltres i, en aquest cas, no podrem fer res. Tot i això, perquè els problemes no siguin produïts pel nostre propi sistema, caldrà prendre les mesures de seguretat adequades durant la implementació i realitzar un bon control d'errors, per al qual ja he dedicat un temps en la planificació temporal.

2.4. Pressupost

2.4.1. Identificació dels recursos

Per aquest projecte, considerem que els recursos es poden classificar en les següents categories:

- Recursos humans
- Hardware
- Software
- Despeses generals
- Contingència i imprevistos

2.4.2. Estimació dels costos

Recursos humans

Per calcular el cost que tindran els recursos humans destinats a aquest projecte, calcularem els costos de les activitats a realitzar com a desenvolupador, vistes en l'apartat de planificació, simulant un preu de mercat. El sou mitjà d'un programador junior està al voltant dels 19.700 € bruts a l'any, tenint en compte els impostos. [14] Les hores hàbils d'una jornada laboral en un any són 1776 hores. [15] Per tant, tenint en compte que li dedicarem unes 450 hores, el pressupost que li haurem de dedicar als recursos humans és de 4991.5 €.

Hardware

Respecte al hardware que utilitzarem, com ja hem comentat, aquest projecte el desenvoluparem majoritàriament des de l'ordinador de sobretaula personal. El preu d'aquest és d'uns 1200 €, però també tindrem en compte els components complementaris, com el ratolí de 30 €, el teclat de 80 € i el monitor de 100 €. Intentarem calcular el cost d'aquest hardware proporcionalment a les hores que li dediquem al projecte, és a dir, ho calcularem tenint en compte que hisenda permet amortitzar un hardware com un ordinador en uns 4 anys i que les hores hàbils en les quals podem treballar en un any són 1776 hores, de les quals, dedicarem 450 hores al projecte. El valor residual d'un hardware ficarem que és un 20% del cost original. Amb totes aquestes dades ja podrem calcular l'amortització aproximada de la següent manera:

$$\text{valor residual} = 20\% \text{ cost inicial}$$

$$\text{amortització anual} = \frac{(\text{cost inicial} - \text{valor residual})}{\text{vida útil}}$$

$$\text{amortització projecte} = \frac{\text{hores projecte} \times \text{amortització anual}}{\text{hores treballades anuals}}$$

Per exemple, per a l'ordinador serà:

$$\text{valor residual} = 20\% 1200\text{€} = 240\text{€}$$

$$\text{amortització anual} = \frac{(1200\text{€} - 240\text{€})}{4 \text{ anys}} = 240\text{€}$$

$$\text{amortització projecte} = \frac{450\text{hores} \times 240\text{€}}{1776\text{hores}} = 60.8\text{€}$$

Recurs	Preu (€)	Vida útil	Amortització (€)
Ordinador	1200	4 anys	60.8
Ratolí	30	4 anys	1.5
Teclat	80	4 anys	4
Monitor	100	4 anys	5
Total:	1410	–	71.3

Taula 3. Pressupost hardware

Software

Pel que fa al software, calcularem l'amortització de la mateixa manera que hem fet amb el hardware, tenint en compte el preu d'aquest, la seva vida útil i el temps de 450 hores que li dedicarem al projecte. Aquí, però, tindrem la variació que un software no té valor residual, ja que un cop se t'acaba la llicència ja no el tens.

Recurs	Preu (€)	Vida útil	Amortització (€)
Windows 10 [17]	145	3 anys	12.24
Microsoft Office [27]	69	1 any	17.48
AWS [2]	9.2	1 mes	27.9
JavaScript i NodeJS	0	-	0
MS SQL Server [45]	989	10 anys	25
Git i Github	0	-	0
Altres	0	-	0
Total:	-	-	82.62

Taula 4. Pressupost software

Despeses generals

Hi ha moltes despeses generals que provenen directa o indirectament del desenvolupament del projecte. Les que considero més importants per tenir en compte en aquest treball són l'electricitat que gasta l'ordinador, l'espai de treball i el transport. Per calcular l'espai de treball he tingut en compte que treballo majoritàriament des d'un lloc petit com és la meva habitació d'uns 8 metres quadrats i, actualment, el preu del metre quadrat està al voltant de 15.2 €/m² mensuals. [21] Addicionalment, el preu del transport es basarà en una targeta de transport T-jove de 3 zones. Per últim, en referència a l'electricitat que gasta un ordinador de sobretaula amb tots els seus complements, aquest en una jornada laboral de 8 hores de treball gasta 2.2kWh [16] i el preu del kWh és de 0.18 €. [33] En definitiva, l'electricitat de l'ordinador cada hora ens costarà 0.0495 € que ho multiplicarem per les hores totals del projecte.

Recurs	Preu (€)
Espai de treball	608
Transport	40
Electricitat	22.2
Total:	670.2

Taula 5. Pressupost despeses generals

Contingència i imprevistos

Com ja hem vist, existeixen uns certs riscos que ens podrien generar imprevistos, els quals podrien provocar que s'allargués el projecte i necessitéssim dedicar més hores. A més a més, el pressupost que estem calculant utilitza dades molt genèriques i aproximades, és a dir, seria molt normal que el pressupost final sigui diferent del previst. Per aquest motiu, afegirem un 10% del pressupost total per contingència i tenir així un marge d'error, evitant que ens quedem curts de pressupost en cas de pujada de preus o errors de càlcul.

Pressupost final

En la següent taula es mostra el resum del pressupost final desglossant les diferents categories vistes anteriorment.

Recurs	Preu (€)
Recursos humans	4991.5
Hardware	71.3
Software	82.62
Despeses generals	670.2
Total	5815.62
Contingència (10%)	581.56
TOTAL FINAL:	6397.18

Taula 6. Pressupost total

2.4.3. Control de gestió

En definitiva, el pressupost total que requereix el projecte és de 6397.18 €. Aquest és un pressupost molt aproximat i ve donat en gran part per als recursos humans, el qual representa el 78% d'aquest pressupost final. Si a més a més li afegim que les amenaces i imprevistos més importants, com per exemple si al final li hem d'acabar dedicant més hores a la implementació de codi, venen relacionades també amb el cost dels recursos humans. Per altra banda, considero que les amortitzacions en hardware i software passen a tenir un paper més secundari en aquest control de gestió i el seu preu no es veurà afectat per cap imprevist. A més a més, per evitar sorpreses inesperades i poder gestionar bé el pressupost, hem dedicat un 10% extra a la contingència, sobretot per garantir l'èxit en cas que ens sobrepassi el pressupost dels recursos humans o de les despeses generals. Tot i això, durem a terme un mecanisme de control durant les reunions de seguiment i a mesura que avanci el projecte, per assegurar que el projecte es mantingui dins del pressupost, analitzarem les despeses generades fins aquell punt i les compararem amb despeses previstes vistes anteriorment, calculant la part proporcional segons les hores que portem de projecte i a través de les fórmules mostrades en l'apartat anterior. Així doncs, en cada moment sabrem si anem ben encaminats, sobretot amb els costos humans, i decidirem si fa falta algun canvi o es mantindrà igual.

2.5. Informe de sostenibilitat

2.5.1. Introducció

Avui en dia, qualsevol empresa que realitza un projecte d'enginyeria o de qualsevol altre àmbit, s'enfoca en obtenir la màxima rendibilitat i benefici econòmic, deixant de banda altres aspectes també molt importants com són l'impacte social, ambiental o econòmic. La falta d'atenció en aquests podria provocar conseqüències molt greus, tant per l'empresa com per la societat en general.

Com a persona que està començant un projecte d'informàtica, soc conscient que els beneficis personals d'un projecte són rellevants, però no són l'única cosa a tenir en compte ni molt menys, sinó que hem de tenir molt presents les afectacions que pot provocar aquest, sigui de manera positiva o negativa, a la vida de les altres persones o al medi ambient. Així doncs, per tal d'identificar les possibles afectacions i conseqüències d'aquest projecte utilitzarem la matriu de sostenibilitat.

	DP	VIDA ÚTIL	RISCOS
AMBIENTAL	Consum de disseny	Petjada ecològica	Ambientals
ECONÒMIC	Factura	Pla de viabilitat	Econòmics
SOCIAL	Impacte personal	Impacte social	Socials

Taula 7. Matriu de sostenibilitat

A continuació analitzarem la sostenibilitat del projecte en les respectives dimensions: ambiental, econòmica i social. Per cadascuna d'elles, tindrem en compte el desenvolupament del projecte (DP), des de la planificació a l'elaboració, la seva vida útil, que comença d'ençà que s'implanta el projecte en un cas real i, per últim, el seu risc en general. En la següent figura podem observar aquestes etapes de manera gràfica.

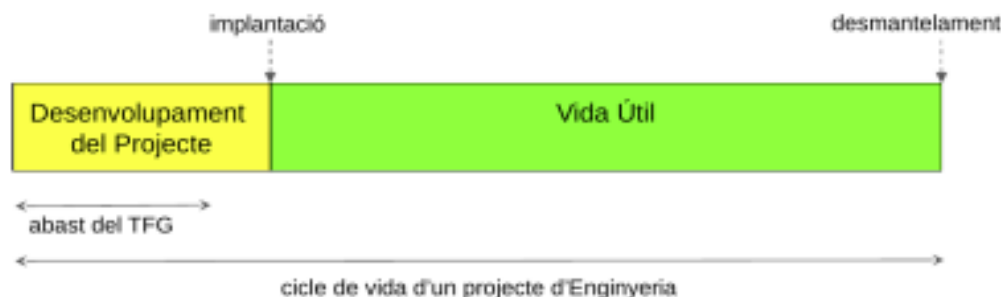


Figure 13. Abast del TFG

2.5.2. Dimensió ambiental

Durant el desenvolupament del projecte, un dels aspectes més importants a tenir en compte és el seu impacte ambiental per tal d'identificar-lo i fer lo possible per minimitzar-lo. En el nostre cas, considero que aquest projecte utilitza molt pocs recursos i, per tant, consumeix molt poca energia. Com ja hem comentat en l'apartat de pressupost i costos 2.4.2., la gran majoria dels recursos són humans i el cost ambiental ha estat molt baix. Tot i això, hem volgut analitzar i quantificar la despesa energètica que suposa realitzar aquest treball.

Per quantificar l'impacte ambiental, doncs, el mesurarem en kilowatts per hora (kWh). Així doncs, tindrem en compte l'energia consumida per el nostre ordinador personal durant la realització d'aquest projecte. Aquestes dades ja les hem vist en [l'apartat 2.4.2.](#), concretament en les despeses generals, quan calculàvem el preu de l'electricitat. Un ordinador de sobretaula encès durant una jornada laboral de 8 hores, tenint en compte també els diferents components, hem vist que gasta 2.2kWh [16]. En conclusió, calculant la despesa energètica d'aquest ordinador en les 450 hores que dura el projecte obtenim que l'impacte ambiental corresponent al desenvolupament d'aquest projecte és de 123.75 kWh. A més a més, en un procés de constitució d'un software és imprescindible l'ús de l'ordinador per a totes les fases. En conseqüència, considero que l'impacte ambiental és el mínim i el podem minimitzar de cap manera.

En un cas hipotètic en el qual s'acaba implantant el nostre projecte i la web de Hit Systems passa a tenir la nostra funcionalitat del generador d'horaris cal veure també quin serà el seu impacte ambiental. En aquest cas, ens és impossible calcular quin serà el resultat de l'energia consumida en el transcurs de la seva vida útil, ja que depèn molt dels anys que s'utilitzi, la quantitat de temps que l'utilitzin els usuaris i de moltes altres circumstàncies. No obstant això, penso que la implantació d'aquest projecte sí que aportaria beneficis al medi ambient i contribuiria a reduir la petjada ecològica pels següents motius:

Reducció del ús del paper:

El primer motiu és la solució que hi ha actualment per abordar aquest projecte, concretament la solució que utilitza la meva mare que, com hem explicat, és una possible usuària d'aquest software. Ella per crear els horaris ho fa manualment, usant un gran nombre de fulles de paper per a cada treballador i, a més a més, de forma setmanal. En conseqüència, passar a abordar aquest problema de manera digital a través del nostre software reduiria aquest ús constant de la impressora i el malbaratament de paper.

Hosting en el núvol:

Però aquesta no seria l'únic avantatge, ja que nosaltres implementariem aquesta funcionalitat en un servei de hosting al núvol, és a dir, a través d'AWS podríem utilitzar un servidor allotjat al núvol que una de les característiques que ofereix és que aquest servidor només funciona quan és estrictament necessari. Per exemple, el servidor només s'executa durant el temps que és necessari processar l'algorisme de generador d'horaris. Aquesta característica significaria que, comparat amb altres tecnologies actuals que solucionen aquest problema, la nostra produiria un impacte ambiental molt menor.

Respecte als riscos en relació a la dimensió ambiental, l'únic escenari que augmentaria la petjada ecològica seria en el cas que durant aquest projecte es requerís més temps i, per consegüent, augmentés la despesa energètica durant el desenvolupament del projecte, en canvi, durant la vida útil no existirà aquest risc, ja que com hem vist es reduiria aquesta petjada ecològica.

2.5.3. Dimensió econòmica

La dimensió econòmica de la fase de desenvolupament del projecte amb la seva estimació dels costos humans i materials l'hem estudiat en detall, junt amb la gestió dels costos, en l'[apartat 2.4](#). Finalment, contant els recursos humans, el software i hardware i altres despeses, ens ha donat un pressupost total de 6397.18 €. Aquests costos són aproximacions i amortitzacions, on per exemple, hem calculat el cost del personal en el cas hipotètic i, per tant, en aquest TFG, no tindria sentit realitzar una quantificació dels costos reals que ha suposat. A més a més, s'han acabat utilitzant tots els recursos estimats en un inici, així que seria molt complicat reduir el cost estimat al pressupost.

Durant la vida útil d'aquest software en el cas que s'acabi implantant, està clar que, a part de beneficis ambientals, portaria sobretot beneficis econòmics tant en la generació d'ingressos com en la reducció de costos. En primer lloc, des del punt de vista de l'empresa que ofereix solucions informàtiques, com podria ser Hit Systems, aquesta podria oferir una nova funcionalitat implementada a la seva web, el generador d'horaris i generar un benefici econòmic a canvi d'oferir aquesta nova solució informàtica de gestió de recursos humans. Per altra banda, des del punt de vista de l'empresa que utilitza aquest software, com podria ser per exemple Cal Forner, també podria obtenir aquests beneficis, ja que si obté un horari òptim per als seus treballadors podria augmentar la seva satisfacció i eficiència de cara al treball. Addicionalment, aquesta empresa veuria reflectida aquest avantatge en la reducció de costos, perquè els encarregats RRHH no haurien de dedicar tanta estona a la generació d'horaris. Aquí també podríem tenir en compte la reducció dels costos ambientals que hem vist en la secció anterior, com per exemple l'ús de grans quantitats de paper.

Per tant, considero que aquest projecte és viable econòmicament tant en el desenvolupament com en la seva vida útil, ja que l'únic cost addicional que suposa la implantació d'aquest software és el cost de manteniment. És a dir, caldria afegir un cost extra per a que una persona realitzés un manteniment del software en cas que aquest doni errors tècnics o s'hagin de realitzar actualitzacions. Això, però, no seria quelcom negatiu, ja que aquesta persona també podria seguir investigant el treball i afegir-hi millores.

En quant als riscos econòmics, ja els hem identificat i analitzat en [l'apartat d'objectius i riscos 1.4.3](#), i també en [l'apartat 2.3](#), que parlem de la gestió d'aquests. En resum, si que és cert que respecte l'àmbit econòmic patim certes amenaces, sobretot pel que fa a la insatisfacció del nostre producte per part de l'empresa o dels mateixos treballadors que, en aquest cas, si podria provocar pèrdues econòmiques a les empreses.

2.5.4. Dimensió social

Respecte al desenvolupament del projecte també s'ha de tenir molt present quines són les conseqüències socials. En l'àmbit més personal, penso que aquest projecte em podia aportar i m'ha acabat aportant grans coses. En primer lloc, en l'experiència adquirida, ja que he pogut obtenir certs coneixements en informàtica en les tecnologies utilitzades per a desenvolupar aquest software. En aquest projecte hem parlat de la inexperiència que tenia en un inici, doncs en aquest projecte he pogut aprendre i treballar en un entorn de treball nou, aprendre un llenguatge de programació que no havia utilitzat mai, com és JavaScript i utilitzar tecnologies noves per mi com MS SQL Server, AWS o NodeJS. En general, he après tecnologies relacionades amb el desenvolupament web i he adquirit coneixements que em poden aportar molt de cara al futur. Addicionalment, seguint en la línia del nivell personal, un dels objectius principals d'aquest treball, com he comentat, és ajudar a la meva mare i a altres persones que m'envolten i, per tant, aquest projecte ens pot aportar molt.

En l'àmbit social durant la vida útil del software, també és rellevant plantejar-se com pot afectar el sistema que volem crear a la societat en general i, sobretot, a les persones que aquest afecti directament. Les persones a les quals més afectaria aquest sistema són la persona o persones encarregades de la gestió del personal i també aquest personal, que podrien ser tots els treballadors de qualsevol empresa usuària del software. Com hem comentat anteriorment, en aquest sentit correm el risc de si el sistema deixarà satisfets a aquestes persones o no, ja que això afectarà directament a la seva qualitat de vida. Tot i això, no és cap drama i l'impacte d'aquest risc no serà tan elevat com sembla, ja que el nostre sistema busca ser una solució complementaria, una funcionalitat extra, així que si aquests usuaris estiguessin insatisfets podrien utilitzar un altre mètode que preferissin. En el cas contrari, on el sistema que hem

implementat funciona perfectament i els hi resulta d'utilitat, aquest portaria grans beneficis socials, com poden ser els següents:

- Millorar la qualitat de vida dels treballadors, ja que els treballadors comptarien amb un sistema on poder mostrar les seves preferències, per exemple, si vol les tardes lliures o un dia en concret lliure, i augmentant la seva satisfacció en l'empresa.
- Millorar la qualitat de vida de la persona encarregada de RRHH, perquè s'estalviaria molt temps si l'horari el pot crear de manera automàtica a través del nostre producte, el qual podria dedicar a altres feines i estar més tranquil·la.
- Fomentar un ambient de treball positiu, al permetre una millor organització de les feines, i més igualitari, ja que davant de l'algorisme totes les persones són tractades de manera equivalent i justa.

2.6. Integració de coneixements

Per tal d'afrontar aquest projecte, he hagut d'adquirir alguns coneixements que no tenia en un inici, com per exemple la utilització del llenguatge de programació JavaScript o la utilització de MS SQL Server. Tot i això, durant el tot procés del treball, he hagut d'aplicar un conjunt de coneixements apresos al llarg dels estudis en la FIB. Les disciplines més utilitzades en aquest projecte són sobretot de la part més específica de la carrera que és durant l'especialitat de Computació i són les següents:

- **Llenguatges de programació**

Haver realitzat l'assignatura de llenguatges de programació (LP), a més a més d'altres assignatures on treballàvem altres llenguatges, m'ha estat de gran ajuda per a poder aprendre JavaScript, un llenguatge orientat a objectes que comparteix moltes similituds amb alguns altres. Així doncs, gràcies als coneixements previs de diferents llenguatges, m'ha resultat molt més senzill el seu aprenentatge.

- **Algorismes i Cerca Heurística**

En les assignatures d'Algorísmia (A) i Intel·ligència Artificial (IA), vam treballar molt el fet de buscar una solució algorísmica a certs problemes en concret i vam utilitzar molt la cerca heurística en forma d'arbres per a trobar la solució òptima o la que més ens interesses segons certs criteris. Aquests conceptes, com ja hem vist en l'apartat introductori de la memòria, han estat essencials per a poder solucionar el tipus de problema que planegem nosaltres.

- **Constraint Programming**

Durant l'assignatura de Lògica a la Informàtica (LI), vam treballar aquest tipus de problemes on tenim un seguit de restriccions i una possible manera de solucionar-lo és usant *constraint*

programming, convertint tot el problema en fórmules lògiques. Per obtenir més detalls sobre aquesta metodologia, llegir [l'apartat 1.5.2](#).

2.7. Identificació de lleis i regulacions

Inclús en un projecte d'enginyeria petit com és el nostre cas, és important efectuar un estudi per damunt de les possibles lleis i regulacions que podrien ser rellevants i afectar el projecte. Tot i que no entrarem molt en detall en les diferents legislacions, ja que aquestes tenen molts punts i casuístiques diferents, tot seguit mostrem algunes lleis que ens poden afectar:

➤ **Llei de protecció de dades**

En primer lloc, tenim la llei orgànica de protecció de dades personals i garantia dels drets digitals. En el nostre cas, treballem amb la base de dades d'una empresa com és Hit Systems, així que tenim accés a certa informació digital de les dades personals d'alguns treballadors. Aquesta llei fou aprovada en 2018 i tracta de molts detalls respecte a les dades personals dels ciutadans d'Espanya i com es poden tractar en tots els possibles àmbits. [6] Nosaltres en tot moment complim aquesta llei perquè Hit Systems només ens proporciona l'accés i els permisos d'edició a una petita part de la seva enorme base de dades, concretament a les dades de Cal Forner. Amb la qual cosa, jo tinc el consentiment per part d'ambdues empreses per poder accedir i utilitzar-les per al desenvolupament de l'algorisme i les proves necessàries.

➤ **Llei de la propietat intel·lectual**

També s'ha de tenir en compte la llei de la propietat intel·lectual [38]. Aquesta llei consisteix en un seguit d'articles que protegeix diferents drets com per exemple els drets d'autor, és a dir, quan un autor produeix una obra, per exemple un llibre, una pel·lícula, un software, etc., la llei dicta uns límits i estableix una protecció perquè altres persones no puguin copiar-ho o explorar-ho. En el nostre cas, el software és desenvolupat al 100% per nosaltres, això sí, utilitzant eines de software lliure com per exemple les llibreries npm de NodeJS. Addicionalment, tots els articles científics i informació recollida en aquest projecte és cita correctament en l'apartat de referències al final de la memòria.

➤ **Regulacions i drets laborals**

En l'apartat de riscos ja hem detallat l'amenaça que tenim en cas de no complir les regulacions i els drets laborals. Aquests han d'estar presents en tot moment, ja que el nostre projecte està directament relacionat amb l'horari que realitzaran els treballadors,

el qual generarem nosaltres. Per tant, haurem de respectar aspectes com la jornada laboral, el descans setmanal, les vacances i altres drets fonamentals del treballador, que detallarem més endavant durant l'explicació de l'algorisme.

2.8. Conclusions planificació

Per concloure la planificació, cal dir que la majoria d'aquesta es va dur a terme als inicis d'aquest projecte. La planificació s'ha seguit com es marcava en un principi, sense canvis notoris en el pressupost, la gestió o les tasques a desenvolupar, que han sigut sempre les mateixes. Malgrat això, si hem patit alguns canvis en el context del disseny i la implementació del mateix projecte.

A mesura que ha anat avançant el temps, com és lògic, t'acabes trobant en diferents situacions que et fan prendre determinades decisions per tal de seguir endavant amb el projecte per obtenir els millors resultats i modificant algunes idees que tenies al principi. En termes de planificació temporal, a causa de la falta d'hores per a realitzar el projecte que volia, ens vam veure obligats a endarrerir la presentació del treball, utilitzant el torn extraordinari per a presentar-ho. Pel que fa al projecte, en un inici teníem la idea de crear tota la web i el codi necessari per aconseguir aquest software, des de zero. És a dir, com ja hem comentat en l'apartat de la descripció general de la metodologia [1.7.1](#), en un inici es volia crear el *frontend* i el *backend* de la web, utilitzant HTML, CSS i NodeJS, junt amb altres tecnologies necessàries. Però a mesura que avançava l'estudi i el projecte, vam veure que realitzar-ho tot podia ser inviable per falta de temps i de coneixement, així que finalment, parlant i rebent l'ajuda del Jordi, vam decidir implementar la funcionalitat a la seva web. En conseqüència, tot i haver malgastat temps en altres coses que finalment no seran d'utilitat i començar més tard, ens podíem centrar més en la part de l'algorisme que és la que realment interessava, no en la part de la web, i així doncs, tot i canviar d'entorn, ens va facilitar les coses a llarg termini.

En definitiva, en aquesta planificació actual ja s'han tingut en compte el nou disseny i les noves tecnologies en el pressupost i gestió, per tant, l'única desviació important ha estat en les hores dedicades per part dels recursos humans, tot lo altre ha seguit el mateix camí. Però hem de tenir en compte que ja vam aplicar un 10% del pressupost a la contingència perquè érem conscients d'aquest risc i, en conseqüència, no ha fet falta afegir més pressupost del que ja teníem previst.

3. Desenvolupament

3.1. Introducció

En aquest punt, ja sabem perfectament de què tracta aquest treball i quin és el seu context. En els capítols anteriors, hem mostrat una introducció detallada del projecte, hem formulat el problema, el seu abast i la metodologia utilitzada, a més a més de tot lo referent a la planificació i molts altres detalls.

En aquest apartat mostrarem la part més pràctica d'aquest projecte i tot lo referent al desenvolupament del generador d'horaris. Descriurem els diferents moments per als quals hem passat, des de la fase de disseny, mostrant l'arquitectura i entorn principal, passant per la llarga etapa d'implementació del codi que explicarem minuciosament i acabant amb la realització de les proves necessàries i l'exposició dels resultats.

3.2. Disseny

3.2.1. Definició del problema

Durant [l'apartat 1.3.2](#), hem vist una descripció general del popular problema *Nurse Scheduling Problem*. També hem comentat que aquest problema té moltes variants segons el context, ja que per exemple, en el *Nurse Scheduling Problem* normalment es tracta amb torns de treball estables i, com veurem tot seguit, no és el nostre cas. Així doncs, en un principi definirem el problema d'una manera més específica, identificant i detallant tots els conceptes que envolten el nostre problema en concret:

➤ **Botiga**

En primer lloc, nosaltres tindrem el cas d'una sola botiga, tot i que com hem mencionat, aquest problema podria tenir moltes variants i especificacions diferents i podria ser que una altra empresa tingués varies botigues amb els mateixos treballadors. Concretament, aquesta botiga es tracta d'una cafeteria/pastisseria/forn de pa i tindrà un conjunt de dependents que hi treballen. Perquè les dependents puguin treballar de manera organitzada i eficient, s'ha de definir un horari per a aquesta botiga.

➤ **Dependentia**

La dependentia serà aquella persona encarregada d'atendre els clients de la botiga. El nombre de dependents variarà molt segons el nivell de l'empresa que gestiona la botiga, així que haurem de tenir en compte casos amb poques dependents i casos amb bastants d'elles. També tindran uns torns assignats per a treballar en la botiga i, probablement, mostraran certes preferències amb relació als torns de treball. Una dependentia s'identificarà a través d'un codi o del mateix nom i tindrà un contracte amb l'empresa que marcarà les hores que pot treballar setmanalment. A més a més, cada dependentia tindrà un calendari laboral concret, on pot tenir els dies festius i les vacances corresponents.

➤ **Torn**

Un torn es refereix a un període de temps amb el qual es divideix un horari, és a dir, un horari es dividirà en diferents torns de treball perquè les dependents puguin treballar de manera senzilla i ordenada. Hi ha empreses on els torns de treball són constants, per exemple tenen un torn de matí, un torn de tarda i un torn de nit i aquests sempre tenen les mateixes hores d'inici i fi. Nosaltres tractem un cas diferent i una mica més complex, ja que cada torn de treball pot tenir diferents característiques. Això és degut al fet que la botiga vol realitzar l'horari més eficient possible per tal de cobrir els dies i les hores segons els requisits de feina. En conseqüència, cada torn tindrà una hora d'inici i de fi i una data en concret. Aquests torns formaran part de l'horari i, tot i que en diferents dissenys del problema es podria considerar assignar diferents treballadors a un mateix torn, en el nostre cas, cadascun podrà tenir assignat únicament un treballador.

➤ **Horari**

La botiga tindrà un horari de treball, el qual representa la distribució de torns i dependents i és l'objectiu que volem generar a partir de totes les altres dades. És a dir, a partir de les dependents, els torns, les restriccions i les preferències volem aconseguir aquesta possible distribució en format d'horari. Així doncs, un horari serà la planificació temporal de la botiga durant un període de temps, concretament nosaltres ens enfocarem en el període diari i setmanal:

- Horari diari:

És important tractar cada dia per separat per tal que, per exemple, una dependentia no faci més torns en un sol dia o per a complir la restricció de dies treballats. A més a més, per tal de contemplar les preferències de les dependents, dividirem els dies en el període de matí i el de tarda.

- Horari setmanal:

L'horari setmanal es pot tractar com el conjunt d'horaris diaris que en concret seran 7, representant els dies de dilluns a diumenge. En altres casos es podria tractar també amb horaris mensuals, però en el nostre cas, cada setmana pot tenir un horari completament diferent de les anteriors i, per tant, cada setmana tindrà l'opció de generar un horari completament nou.

- **Restriccions**

Quan parlem de restriccions ens referim a les limitacions que poden existir per establir l'horari de treball de les diferents dependents. Aquestes restriccions poden ser fortes, si és estrictament necessari que es compleixin per a que la solució sigui vàlida, o suau, en el cas que no sigui estrictament necessària tot i que s'intentarà prioritzar. Una restricció pot venir donada per diversos motius, normalment serà per temes de preferències d'una dependent, com per exemple "la dependent 1 no vol treballar els dimarts", i sobretot una restricció es pot deure als drets i regulacions laborals, per exemple, "qualsevol treballador no pot superar les 40 hores setmanals".

- **Preferències**

Per últim, tenim les preferències personals que poden tenir les dependents respecte al seu horari de treball. Aquestes poden ser úniques i dependran exclusivament dels aspectes personals de cada persona. Sobretot ens centrarem en les preferències horàries, com per exemple si una dependent prefereix treballar un dia en concret a la tarda o al matí. Així doncs, aquestes preferències normalment generaran algunes restriccions.

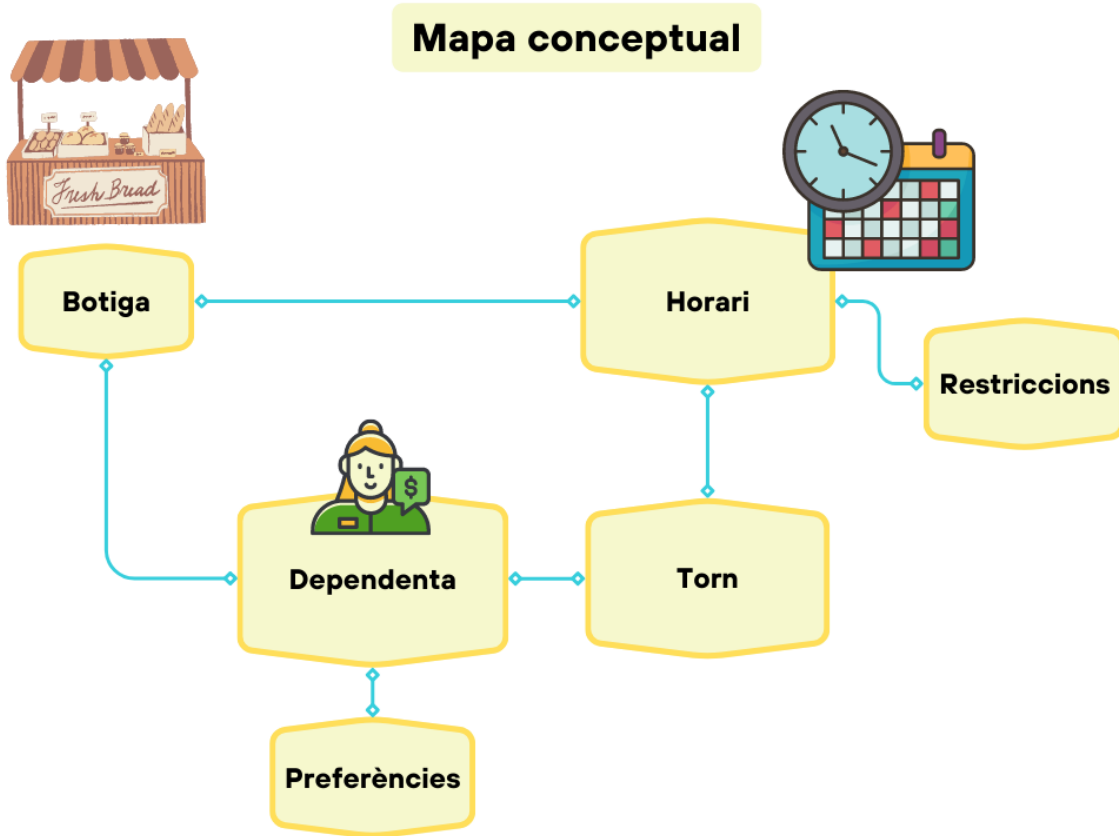


Figure 14. Mapa conceptual del problema

Diagrama de classes:

En l'anterior figura, hem vist un mapa conceptual per entendre de manera simple i esquemàtica els diferents conceptes que hem explicat en aquest apartat. A més a més, també hem dissenyat el diagrama de classes, amb l'objectiu de veure de forma més detallada els atributs, les funcions, les relacions i les multiplicitats de cada classe durant la implementació de codi:

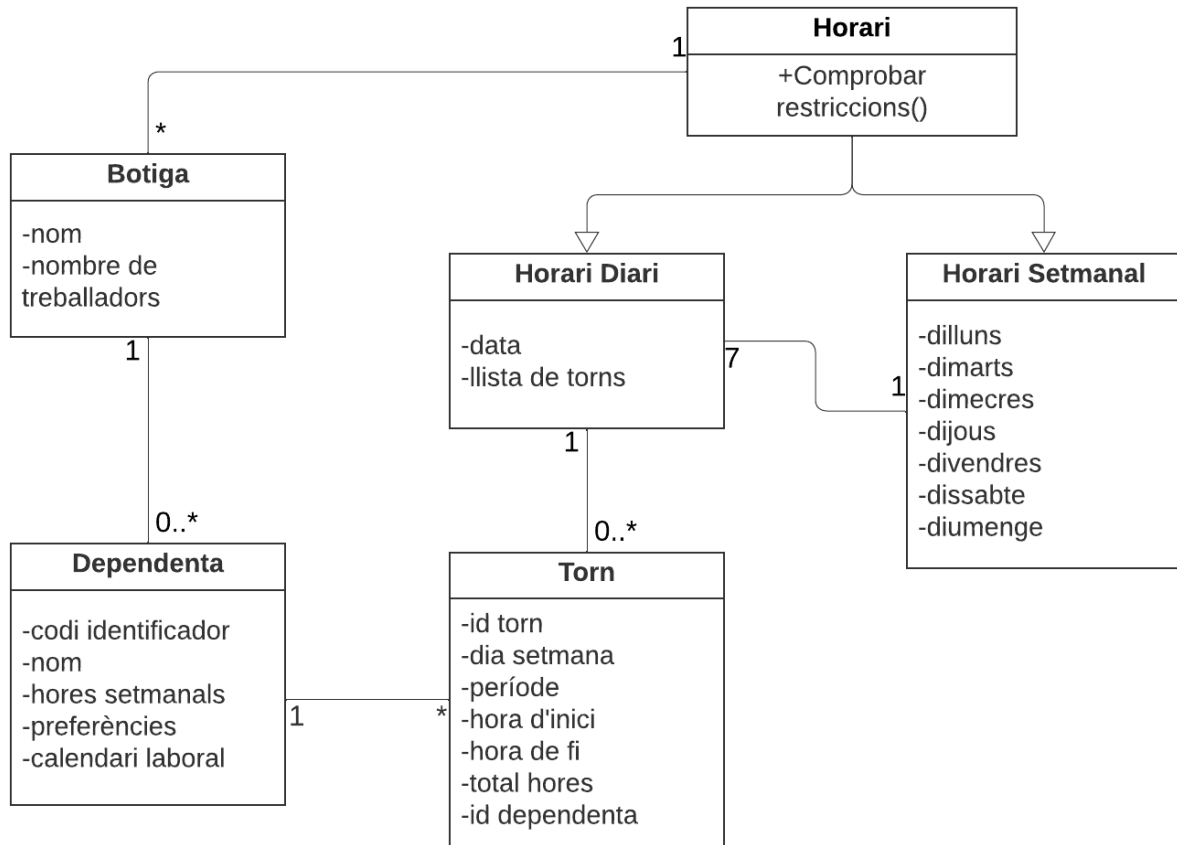


Figure 15. Diagrama de classes

Ja hem explicat el significat de cada classe i de quina manera es relacionen. Tot i això, en el diagrama de classes es pot observar com hem convertit les preferències i el calendari laboral de cada dependenta en atributs d'aquesta, pensant ja en la manera més eficient d'implementació del codi. A més a més, hem representat les restriccions com un mètode de la classe Horari, el qual comprovarà que es compleixin aquestes. Com ja hem comentat, cal diferenciar entre els dies i les setmanes i, en conseqüència, cada horari setmanal estarà compost per 7 horaris diaris, cadascun representant un dia en concret de la setmana (dilluns, dimarts...) i cada horari diari s'exposarà com un conjunt de torns. Cada torn tindrà els seus propis atributs, com el seu propi id i alguns d'altres com mostra en el diagrama, i es relacionarà amb una única dependenta a través de la seva clau primària, que serà un codi identificador.

Aquest disseny ens proporciona una gran ajuda a l'hora de crear el programa, tot i que aquest diagrama no representa tots els detalls del codi i, com veurem més endavant, hi haurà certes diferències i ampliacions.

3.2.2. Arquitectura general de la solució proposada

A continuació, explicarem de manera detallada el disseny de l'arquitectura general de la solució proposada, mostrant com s'estructura i com funcionen els diferents components del sistema. Per tal de realitzar aquest disseny, ens hem enfocat sobretot en els requisits funcionals, els quals són essencials per assolir l'objectiu principal de generar un horari automàtic i personalitzat. Ja els hem vist en [l'apartat 1.4.2](#), on es pot veure la descripció de cadascun, tot i que per recordar-los, els tornem a citar a continuació:

- Entrada dels treballadors.
- Entrada dels torns de treball.
- Sortida final del horari.
- Gestió de la informació.
- Restriccions.
- Visualització del horari.

Per a comprendre bé l'arquitectura general del software, ens centrarem en 4 components principals:

- L'**usuari** que utilitzarà el nostre sistema amb l'objectiu d'aconseguir una solució horària. Aquest usuari normalment serà la persona encarregada dels recursos humans en una empresa, la qual ha de dur a terme els horaris de la plantilla.
- La **pàgina web** on estarà allotjada la nostra funcionalitat. En el nostre cas utilitzarem la web de Hit Systems, la qual està desenvolupada amb les tecnologies ASP i PHP.
- El **programa** que s'implementa en aquest projecte utilitzant el llenguatge de JavaScript en l'entorn d'execució de NodeJS i fent ús d'altres eines, les quals especifiquem en [l'apartat d'eines i tecnologies 1.7.2](#).
- La **base de dades** relacional que conté tota la informació necessària de l'empresa distribuïda en taules, la qual s'utilitza a través del sistema MS SQL Server.

El procés de generació d'horaris constarà de diferents parts, creant un sistema de tal manera que podrem connectar els diferents components esmentats i assolir els requisits funcionals del projecte. Com ja havíem comentat amb anterioritat, alguns processos d'aquest sistema ja existien a la pàgina web de Hit Systems, on ja es poden dur a terme funcionalitats com emmagatzemar dades dels treballadors o crear els horaris (de manera manual). Nosaltres en aquest projecte hem realitzat la implementació del codi per tal de, connectant les diferents utilitats, aconseguir l'horari de manera automàtica. Tot i això, perquè quedi detallat i ben clar,

explicarem tot el procés de generació d'horari des de l'inici i explicant cada pas del sistema dissenyat.

Entrada de dades:

Per començar, una part fonamental per a l'assignació d'horaris és l'entrada de dades, l'usuari ha d'entrar certa informació al sistema perquè aquest pugui actuar i complir els objectius. Perquè el software que hem dissenyat funcioni correctament és estrictament necessari que l'usuari ens proporcioni dues fonts d'informació principals, les dependents i els torns:

En primer lloc, el sistema ha de recopilar informació respecte a les dependents que treballen a la botiga. L'usuari haurà d'entrar unes dades de manera senzilla respecte al nom de la dependenta, quines hores setmanals marca el seu contracte i quines preferències horàries té. Aquesta informació només fa falta que s'introdueixi al principi de la utilització d'aquesta web i en el cas que hi hagi alguna modificació, com per exemple contractar una dependenta nova o modificar les hores contractuals d'alguna treballadora. La resta del temps aquesta informació estarà guardada en la base de dades i no serà necessari tornar a introduir-la. Addicionalment, tot i que aquestes ja no són estrictament necessàries per a l'execució del programa, també es poden donar dades respecte als dies festius i les vacances de cada dependenta.

En segon lloc, l'usuari també haurà d'indicar al programa un seguit de torns que vulgui emplenar per a l'horari final. Així doncs, s'haurà d'indicar els diferents torns de treball que es volen assignar a les dependents a partir d'una hora d'inici i fi, per tal que el sistema sàpiga quines són les hores que es volen cobrir i amb quanta gent.

En un principi, volíem realitzar un disseny on l'usuari no hagués de crear aquests torns i generar-los també de manera automàtica, ja que aquest procés, en el cas que es vulgui crear uns horaris diferents cada setmana, l'usuari haurà de gastar bastant temps en generar aquests torns. Tot i això, el motiu pel qual s'ha optat per aquest disseny és per satisfer les necessitats de l'empresa de manera més eficient. És a dir, tot i que l'usuari hagi de crear aquests torns, aquesta és l'única preocupació que ha de tenir, ja que després el nostre programa analitzarà les diferents opcions i s'encarregarà que les restriccions es compleixin. Addicionalment, vam pensar que era necessària aquesta informació per tal que el sistema sàpiga l'horari i la càrrega de treball de cada dia i hora en concret amb l'objectiu d'assignar les dependents necessàries en cada hora concreta. En el cas de les cafeteries/forns de pa la càrrega de treball en cada hora i dia és molt variant, per exemple en el cas de les hores punta, hores on la botiga tingui encàrrecs, en dies especials, etc. la botiga necessitarà un nombre més elevat de dependents a l'habitual. També en el cas per exemple que un dia es vulgui treballar en un horari en concret, per exemple, començant més d'hora o només treballant de tarda, d'aquesta manera podem tenir en compte totes les possibles casuístiques i generar un horari òptim i eficient per a cada opció.

En definitiva, totes aquestes dades són introduïdes per part de l'usuari a la pàgina web de manera senzilla a través d'una interfície d'usuari que mostrarem més endavant. L'entrada de

dades no requereix cap ordre específic ni cap limitació de quantitat, per exemple, podríem primer entrar la informació dels torns i després de les dependents. El sistema es connectarà amb la base de dades del servidor, on s'emmagatzemaran i ens permetrà gestionar fàcilment aquesta informació de la manera que considerem a partir de la introducció, modificació o eliminació d'aquesta.

Execució del programa:

Una vegada introduïda tota la informació respecte a les dependents i els torns de treball a emplenar s'haurà de cridar al nostre algorisme perquè aquest realitzi l'assignació de torns a les diferents dependents, complint amb els criteris i les restriccions que es considerin. Per tal causa, l'usuari només haurà de pressionar un botó que mostrarà la mateixa web. Aquest esdeveniment executarà la funció principal del nostre programa escrit amb JavaScript, la qual només requereix el paràmetre del dia inicial de la setmana de la qual es vol generar l'horari.

Un cop dins del programa, aquest es connectarà amb la base de dades del servidor, amb la finalitat d'aconseguir tant les dades d'entrada de l'usuari, com totes les dades complementàries que considerem necessàries per a les heurístiques desenvolupades. Aleshores, es produirà l'assignació de dependents als diferents torns generant així la solució horària. El desenvolupament d'aquest codi es detallarà més endavant en l'apartat d'implementació.

En darrer terme, aquesta funció modificarà la base de dades perquè consti el nou horari generat automàticament i actualitzarà la pàgina web per a mostrar-lo. En la següent figura podem observar un esquema simplificat del problema, on a partir d'una llista de dependents i un conjunt de torns donats per l'usuari, les dependents s'han d'assignar als diferents torns complint una sèrie de restriccions.

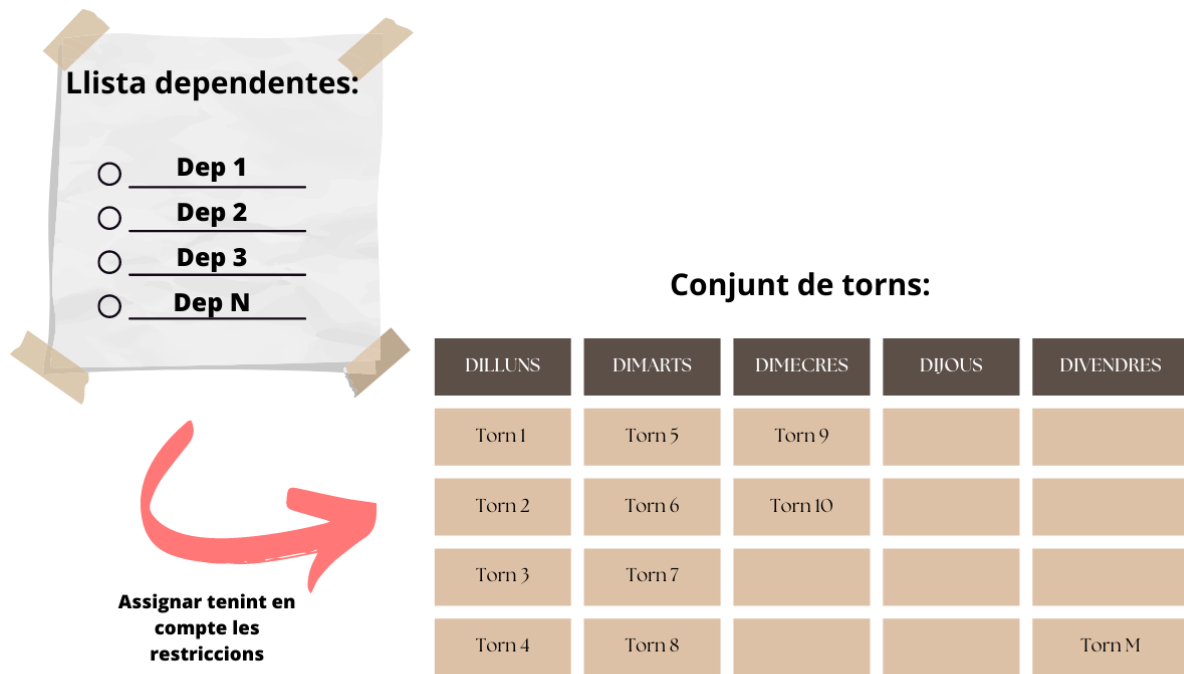


Figure 16. Esquema simplificat del problema

Visualització de l'horari:

En tot moment, la pàgina web permet a l'usuari la visualització dels horaris de cada setmana. Aquesta funcionalitat serà la que permetrà de manera clara i eficient l'entrada de dades dels diferents torns de treball per part de l'usuari. A més a més, un cop s'acabi d'executar el programa podrà observar l'assignació realitzada automàticament i, en cas que ho trobi necessari, podrà realitzar canvis de forma manual.

En la figura 17, mostrem el diagrama de seqüència de la solució proposada, on s'observa clarament l'arquitectura i les connexions de tot el procés que acabem d'explicar.

DIAGRAMA DE SEQÜÈNCIA

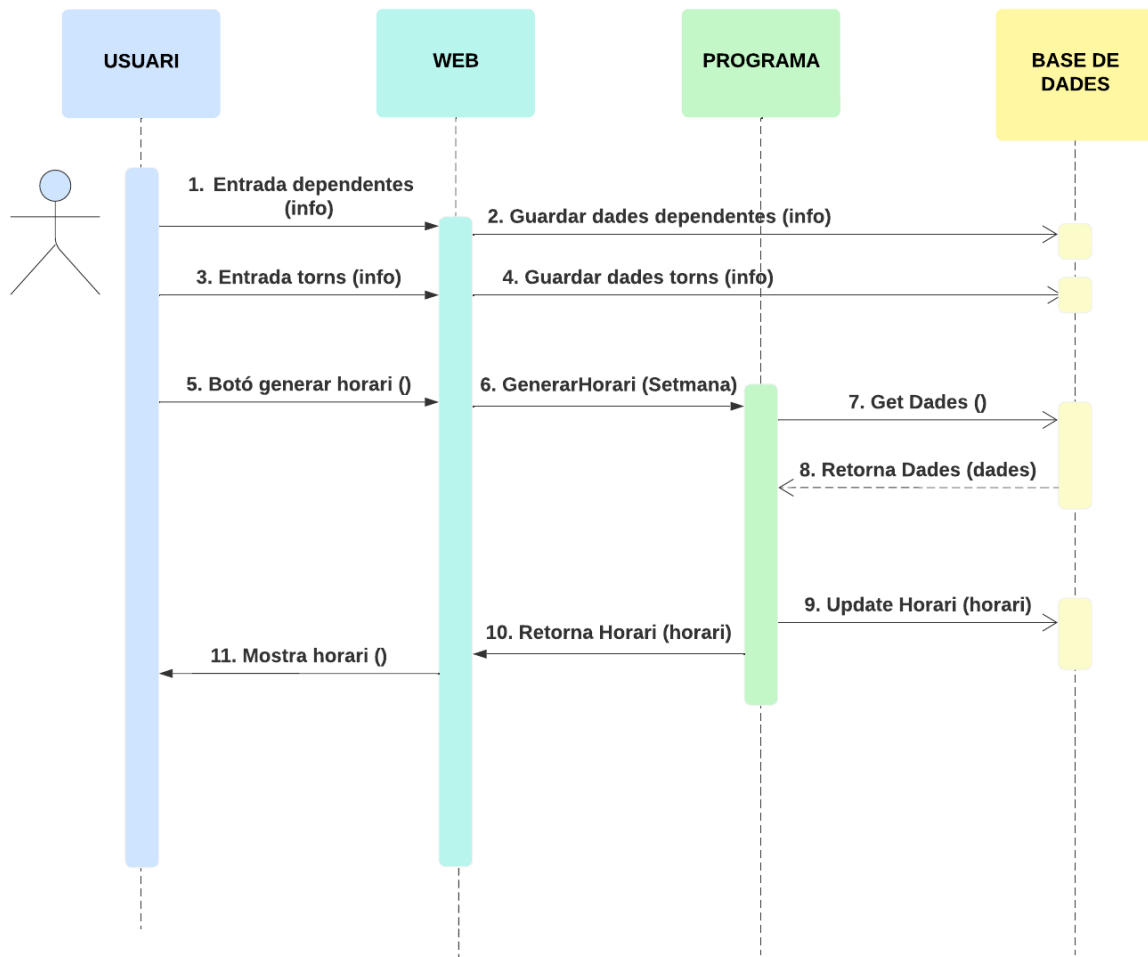


Figure 17. Diagrama de seqüència

Per a que no quedi cap dubte mostrem una petita llegenda de les accions nombrades en el diagrama de seqüència previ:

1. L'usuari entra a la web la informació de cada dependent a part.
2. La informació de les dependents es passa des de la web cap a la base de dades a mesura que l'usuari la introdueix o modifica. Aquestes accions es poden repetir en bucle.
3. L'usuari entra a la web la informació dels torns de treball.
4. Es guarda tota la informació respectiva als torns a la base de dades.
5. L'usuari pressiona un botó de la pàgina web amb l'objectiu que li generi l'horari.
6. La web per mitjà d'un 'event' al prémer el botó, fa una crida a la funció del nostre programa.
7. El programa accedeix a la base de dades per tal de rebre la informació necessària per assignar els horaris.
8. La base de dades retorna al programa les dades requerides. Les accions 7 i 8 es poden repetir en bucle i demanant diferents tipus de dades.
9. Al final del programa, s'actualitza l'horari general a la base de dades.
10. També es passa l'horari final a la pàgina web.
11. Per acabar el procés, la pàgina web mostra l'horari final a l'usuari.

3.2.3. Interfície i visió de l'usuari

El nostre disseny ha de tenir una interfície intuïtiva i fàcil d'utilitzar per tal que l'usuari pugui generar l'horari de forma clara, independentment del seu nivell d'experiència amb la pàgina web. En aquest apartat, explicarem breument quins passos ha de seguir l'usuari i mostrarem quina és la seva visió i experiència respecte a la UI (interfície d'usuari) quan s'usa aquest sistema per primera vegada. Com ja hem comentat, la pàgina web ha estat implementada per l'empresa Hit Systems i té moltes altres funcionalitats a part de la que proposem nosaltres, així que només mostrarem els detalls necessaris per a comprendre el nostre projecte.

En primer lloc, l'usuari accedirà a un portal on es pot veure i modificar la informació dels treballadors d'una empresa, a través d'un buscador.



Figure 18. UI. Portal de cerca treballadors

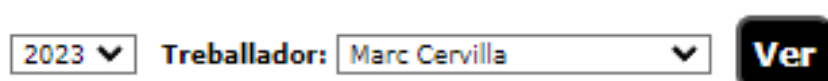
Quan entrem a la informació d'un treballador veurem molts camps, la majoria personals. Els camps que utilitzem nosaltres en aquest projecte són les hores totals que ha de treballar una dependent en una setmana i l'hora d'entrada de cada dia. Aquest camp l'utilitzem per saber les preferències de cada dependent i els dies que vol treballar de matí o tarda. Ha d'estar en format HH:mm però tan sols es mirarà si l'hora correspon al matí o a la tarda.

Hores base total:	<input type="text" value="20"/>														
Hora d'entrada:	<table><tr><td>L</td><td>M</td><td>X</td><td>J</td><td>V</td><td>S</td><td>D</td></tr><tr><td><input type="text" value="08:00"/></td><td><input type="text" value="08:00"/></td><td><input type="text" value="08:00"/></td><td><input type="text" value="14:00"/></td><td><input type="text" value="14:00"/></td><td><input type="text"/></td><td><input type="text"/></td></tr></table>	L	M	X	J	V	S	D	<input type="text" value="08:00"/>	<input type="text" value="08:00"/>	<input type="text" value="08:00"/>	<input type="text" value="14:00"/>	<input type="text" value="14:00"/>	<input type="text"/>	<input type="text"/>
L	M	X	J	V	S	D									
<input type="text" value="08:00"/>	<input type="text" value="08:00"/>	<input type="text" value="08:00"/>	<input type="text" value="14:00"/>	<input type="text" value="14:00"/>	<input type="text"/>	<input type="text"/>									

Figure 19. UI. Dades d'un treballador

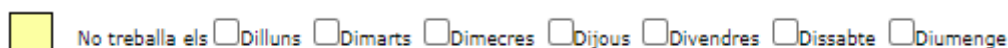
L'únic camp obligatori és el de 'hores base total', si 'hora d'entrada' està buit el programa interpreta que l'usuari no té preferència i el podrà assignar de matí o tarda indistintament.

Addicionalment, en un altre portal, es podrà modificar el calendari laboral de cada dependent. A través d'uns desplegable, podrem seleccionar l'any i el treballador i ens apareixerà el seu calendari anual on podrem marcar els dies festius, de vacances, de baixa, etc. Per exemple, en les següents figures podem observar com el treballador Marc, li hem donat un dia festiu el dia 31 i els dissabtes no treballa. Si es vol fer que no treballi tots els dissabtes d'aquell any, tan sols hem de marcar una casella.



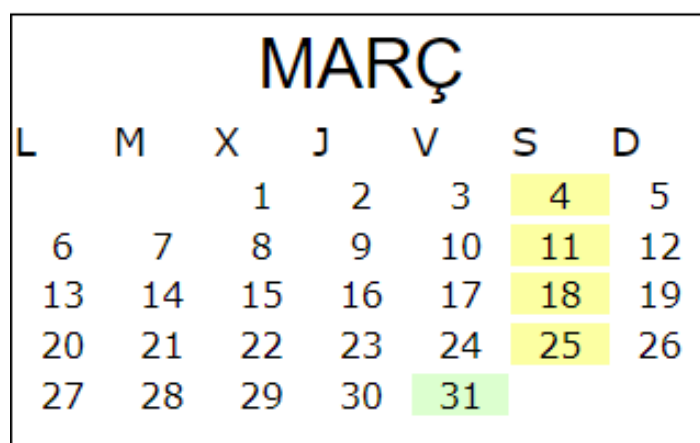
2023 ▼ Treballador: Marc Cervilla ▼ Ver

Figure 20. UI. Desplegables calendari laboral



No treballa els Dilluns Dimarts Dimecres Dijous Divendres Dissabte Diumenge

Figure 21. UI. Funcionalitat per marcar un dia de la setmana com a festiu



MARÇ						
L	M	X	J	V	S	D
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

Figure 22. UI. Exemple del mes de març d'un calendari laboral

Un cop introduïda tota la informació de les dependents, passem a entrar els torns que es volen cobrir per aquella botiga. Disposem d'un portal on es pot visualitzar molt fàcilment l'horari d'una setmana buscant la seva data, llavors ens apareixeran els dies de dilluns a diumenge

separats per matí i tarda. Òbviament, els botons són falsos, però seria la idea en cas que el projecte s'acabés implantant, també afegint la funcionalitat de copiar la setmana anterior.

The image shows a web interface for managing schedules. At the top, there is a date input field set to '05/06/2023', a dropdown menu for 'Botiga' set to 'BOT Granollers', a 'Ver' button, and two blue buttons: 'Horari automàtic' and 'Copiar setmana anterior'. Below this, there are three columns representing dates: '08/06/2023', '09/06/2023', and '10/06/2023'. Each column contains a 'Nou Torn' button and the text 'SIN DEFINIR'.

Figure 23. UI. Portal per modificar i visualitzar horaris.

Així doncs, l'usuari haurà d'introduir un seguit de torns segons les hores que vulgui cobrir. Per exemple, el dia 10 de juny al matí vol cobrir 3 torns, però sobretot necessita 3 dependents entre les 08:00 i les 12:00, doncs obtindrem la següent figura:

10/06/2023

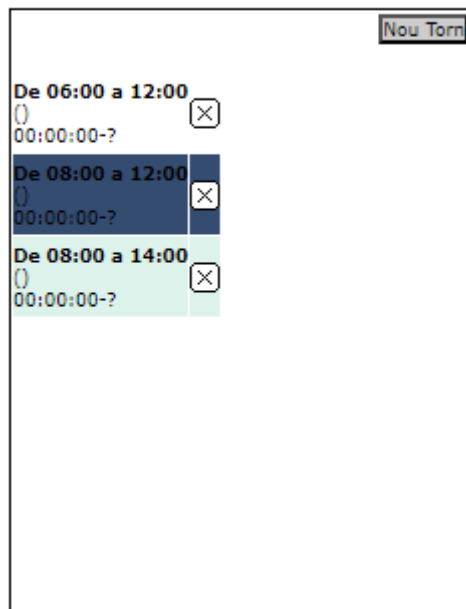


Figure 24. UI. Exemple dels torns planificats d'un dia concret

Un cop introduïda també tota la informació respecte als torns, l'usuari només haurà de pressionar el botó d'horari automàtic, el qual executarà el nostre programa i, finalment, l'usuari podrà visualitzar tota l'assignació horària d'aquella setmana.

10/06/2023

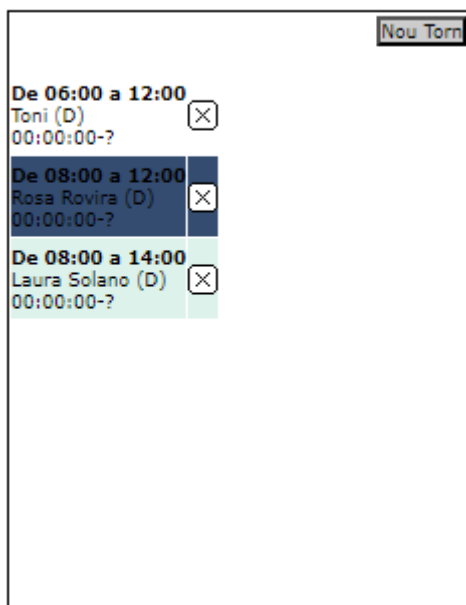


Figure 25. UI. Exemple dels torns una vegada han estat assignats

3.2.4. Base de dades

Ja sabem quina és l'arquitectura del projecte i quina és la funció que exerceix la base de dades. Malgrat això, abans de començar amb la implementació de codi és essencial veure de quina manera s'organitzen les diferents taules d'una base de dades relacional com *MS SQL Server*. D'aquesta manera sabrem de quines dades disposem i com s'emmagatzemen.

La base de dades de la que disposem pertany a Hit Systems i té una gran quantitat de dades, així doncs, nosaltres només tractarem amb una petita part i, en aquesta memòria, només detallarem les taules i atributs necessaris per a la comprensió del projecte. A continuació les mostrem, a partir del nom de la taula, les columnes, el tipus i la descripció de cadascuna i un exemple, tenint en compte que moltes columnes no apareixeran i només explicarem les essencials.

➤ **Taula dependentes:**

Aquesta taula recull informació de les dependentes.

<i>Nom taula</i>	Columnes	Tipus	Descripció	Exemple
<i>Dependentes</i>	Codi	Int	Codi numèric per identificar una dependenta	1
	Nom	String	Nom de la dependenta	'Marta'

Taula 8. Taula dependentes

- **Taula Dependentes Estès:**

Aquesta taula recull informació addicional de les dependentes a través d'una espècie de diccionari <clau,valor>, és a dir, tindrem unes claus específiques com per exemple 'HoresTreballades' i els diferents valors per a cada dependenta segons les seves hores treballades.

<i>Nom taula</i>	Columnes	Tipus	Descripció	Exemple
<i>DependentesExtes</i>	ID	Int	Codi numèric per identificar una dependenta	1
	Nom	String	Clau per identificar un atribut	'Hores Treballades'
	Valor	String	Valor per a l'atribut	'40'

Taula 9. Taula Dependentes Estès

- **Taula Tornos:**

Aquesta taula recull informació sobre els possibles tornos però de manera independent a l'horari o a les dependents.

<i>Nom taula</i>	Columnes	Tipus	Descripció	Exemple
<i>Turnos</i>	IdTurno	String	Codi alfanumèric per identificar un torn	'{A38F..}'
	horaInicio	String	Hora d'inici del torn	'8:00'
	horaFin	String	Hora de fi del torn	'14:00'

Taula 10. Taula Tornos

- **Taula Planificació:**

Aquesta taula recull informació sobre la planificació horària d'una setmana en concret, és a dir, per cada setmana existeix una taula diferent amb l'assignació horària d'aquesta. Una fila de la planificació és molt similar a un torn. En el nom apareix també <Data>, que ens referim a la data d'inici d'aquella setmana (dilluns).

<i>Nom taula</i>	Columnes	Tipus	Descripció	Exemple
<i>Planificació<Data></i>	idPlan	String	Codi alfanumèric per identificar una planificació	'FG37B..'
	fecha	Date	Data (any-mes-dia)	'2017-06-10'
	periode	String	Període del dia (matí, tarda, nit..)	'T'
	idTurno	String	Codi alfanumèric per identificar un torn	'{A38F..}'
	idEmpleado	Int	Codi numèric per identificar una dependentia	3

Taula 11. Taula Planificació

- **Taula Calendari Laboral:**

Aquesta taula recull informació sobre el calendari laboral de tots els treballadors de l'empresa, marcant els dies que són festius o els dies en que una dependentia no treballarà, per exemple, per una baixa, per vacances, etc. Amb <any> ens referim a l'any d'aquell calendari laboral en concret, ja que aquest apareix en el propi nom de la taula i cada any es crea una taula nova.

<i>Nom taula</i>	Columnes	Tipus	Descripció	Exemple
<i>CalendariLaboral<any></i>	Id	String	Codi alfanumèric per identificar un torn	'4548-A7..'
	fecha	Date	Data (any-mes-dia)	'2017-06-10'
	idEmpleado	Int	Codi numèric per identificar una dependentia	3

Taula 12. Taula Calendari Laboral

Com s'observa en les diferents taules, cadascuna té un codi identificador que s'utilitza com a clau primària. Així doncs, tant els torns, com les dependents tenen un codi identificador que no es repeteix i aquesta és la manera amb la qual s'estableixen relacions entre les taules.

3.2.5. Disseny addicional d'un sistema de puntuacions

Fins ara, hem vist tota l'explicació i disseny relacionada amb el nostre projecte i, en aquest apartat introduïrem per primera vegada una proposta de disseny addicional totalment complementaria a la nostra solució proposada fins ara.

La idea és construir un sistema de puntuacions de les dependents per obtenir les seves qualitats i característiques de manera interna utilitzant altres dades complementàries de l'empresa, com per exemple les dades de vendes, la qual cosa ens pot aportar varies coses que dividirem en els següents punts:

- En primer lloc, i respecte als requisits de personalització i utilitat, aquest sistema ens proporcionarà una manera completament nova de realitzar uns horaris molt personalitzats per a l'empresa, ja que els resultats dependran fortament de cada empresa i de les seves dependents, i pot ser de gran utilitat.
- Permetrà mantenir un control de cada dependenta en específic, obtenint la seva puntuació, factor que pot ser diferencial per a la botiga, permetent dissenyar un horari òptim i augmentar tant l'eficiència com el benefici del negoci.
- Aquest sistema també afegirà un valor addicional a la nostra solució proposada respecte a les altres opcions de mercat existents, vistes en [l'apartat 1.5.3](#), les quals no emmagatzemen dades complementàries. En canvi, nosaltres disposem de gran quantitat de dades addicionals que poden ser utilitzades en benefici de l'empresa o de les pròpies dependents.
- A més a més, aquesta proposta serà totalment complementària al problema dels horaris vist fins ara, és a dir, realitzarem dos programes diferents i, finalment, l'empresa podrà decidir quines solucions utilitzar i quines no, podent generar l'horari sense l'efecte d'aquestes puntuacions.
- Aquesta idea és molt escalable, és a dir, amb el temps es pot anar millorant i augmentant la utilitat, per exemple creant un projecte on les dependents amb millors qualificacions obtenen certs beneficis o altres idees que se'ns puguin ocórrer.

El nostre disseny consisteix a crear una taula mensual de puntuacions on cada mes es calculi les qualificacions de les dependents, tenint en compte també els valors acumulats de mesos

anteriors, i aquestes s'emmagatzemin a la base de dades. Introduïm les noves taules de la base de dades que utilitzarem per a aquest disseny:

- **Taula entrades i sortides:**

Aquesta taula recull informació sobre les dades d'entrada i sortida dels treballadors. Així doncs, nosaltres la utilitzarem per saber la puntualitat de les dependents a partir del seu horari real, comparant-lo amb l'horari que s'hauria de fer.

<i>Nom taula</i>	Columnes	Tipus	Descripció	Exemple
<i>DadesFichador</i>	tmst	Date	Data que defineix des de l'any fins a les hores i segons	'2017-06-10T08:03:23'
	accio	String	Entrada (1) o sortida (2)	'1'
	usuari	Int	Codi numèric per identificar una dependenta	3

Taula 13. Taula entrades i sortides

- **Taula vendes:**

Aquesta taula recull informació sobre les dades de vendes de la botiga en un mes concret. Per tant, nosaltres la podem utilitzar per a calcular les vendes d'una dependenta a partir de quants tiquets ha venut o quins imports tenen.

<i>Nom taula</i>	Columnes	Tipus	Descripció	Exemple
<i>Venut<any-mes></i>	dependenta	Int	Codi numèric per identificar una dependenta	3
	Num_tick	Int	Numero de tiquet	34
	import	Float	Import del tiquet	14.25

Taula 14. Taula vendes

En definitiva, el nostre programa cada mes haurà de crear una taula anomenada PuntuacioMensual<any-mes> on guardarem a la base de dades les puntuacions de cada dependent i el seu acumulat dels darrers mesos. Tot i que aquest sistema pot estar molt subjecte a canvis, hem decidit avaluar les següents qualificacions:

- **Puntualitat:**
Es calcularà el temps que habitual amb el que entra la persona, és a dir, es fa una mitjana dels minuts que aquesta arriba tard o d'hora. En principi, només es té en compte l'entrada a la botiga, no la sortida.
- **Tiquet:**
És el tiquet mitjà, concretament la mitjana dels imports dels diferents tiquets que ha venut la dependent.
- **Valoració:**
La valoració es podria calcular de mil maneres diferents, nosaltres proposem que sigui una nota del 1 al 5 que mesuri les dues valoracions anteriors, la puntualitat i el tiquet, amb el següent mètode:
 - 3 punts per la puntualitat:
 - Si arriba 5 o més minuts tard tindria 1 punt. (puntualitat > 5)
 - Si arriba entre -5 i 5 minuts tard tindria 2 punts. (-5 < puntualitat < 5)
 - Si arriba -5 o menys minuts tard tindria 3 punts. (puntualitat < -5)
 - 2 punts per al tiquet:
 - +1 punt si el tiquet mitjà de la dependent és més alt que el tiquet mig de la botiga.
 - +1 punt si el tiquet mitjà es troba en el top 3 més alt.

Així doncs, la nota sempre estarà entre [1,5]. Per exemple, en el cas d'una dependent que sigui puntual i arribi 10 minuts abans cada dia i el tiquet mitjà sigui dels millors de la botiga tindrà una puntuació màxima de 5.
- **Acumulat:**
És el valor acumulat de les valoracions i es calcula com la mitjana entre la valoració del mes actual i les valoracions dels últims mesos, amb l'objectiu d'obtenir una valoració de la dependent durant un període de temps més prolongat.

La taula resultant queda de la següent manera:

<i>Nom taula</i>	Columnes	Tipus	Descripció	Exemple
<i>PuntuacioMensual<any-mes></i>	idEmpleado	Int	Codi numèric per identificar una dependenta	3
	puntualitat	Float	Puntualitat	-10
	tiquet	Float	Tiquet mitjà	18.9
	valoracio	Float	Valoració	5
	acumulat	Float	Valoració acumulada dels darrers mesos	4.65

Taula 15. Taula puntuacions mensuals

3.3. Implementació

3.3.1. Procés d'implementació pas a pas

Un cop acabada la fase de disseny del projecte, de seguida vam iniciar la implementació del codi. Per al correcte desenvolupament d'aquest codi vam utilitzar una metodologia molt específica, on, per tal d'anar avançant amb bon peu, ens anàvem marcant petits objectius / subjectius realistes.

En aquest apartat, mostrarem tot aquest procés d'implementació pas a pas per entendre en detall en quin punt ens trobàvem en tot moment, tot i que per no mostrar tot el codi intermedi, detallarem el resultat final dels algorismes en la pròxima secció. És molt important tenir en compte la diferència de nivell pel que fa al coneixement en un inici i en un final del procés, atès que en un principi tenia molt pocs coneixements en la matèria i estava treballant amb un llenguatge de programació i entorn completament nou. Així doncs, sobretot el principi, es tractava d'un llarg exercici d'aprenentatge i millora respecte a aquest treball.

A continuació, detallarem els objectius que ens vam proposar i els passos que vam seguir per completar-los. Aquest és el punt més tècnic de la memòria on s'utilitzaran conceptes relacionats

amb la programació i amb JavaScript, els quals, o bé, ja s'han definit al llarg de la memòria o bé s'intentaran definir i redactar de forma entenedora per a un públic genèric.

Objectiu 1. Codificar un canvi d'assignació de torn

En primera instància, vam començar amb un programa completament buit. En aquest primer moment, com ja hem comentat, estàvem utilitzant i investigant el seu funcionament per primer cop de les tecnologies AWS Cloud9, JavaScript i NodeJs. Així doncs, el nostre primer objectiu va ser realitzar un cas molt bàsic però essencial per al projecte: intentar implementar un canvi d'assignació d'un torn en el cas d'un horari ja creat amb la web manualment.

Aquesta era la manera per començar a assentar les bases i veure de quina manera es podia accedir a la base de dades (BD), com s'organitzava aquesta i poder llegir o modificar les seves dades, pas clau per al desenvolupament d'aquest projecte.

En primer lloc, vam instal·lar des de npm un paquet anomenat *mssql*, una llibreria necessària per a poder treballar des de NodeJs amb la base de dades de 'MS SQL Server'. [30]

En concret, vam utilitzar les funcions *connect()* i *query()*, per tal de connectar-nos al servidor i poder fer peticions amb el llenguatge SQL.

Amb peticions SQL com per exemple el *SELECT* o l'*UPDATE*, ja podíem realitzar proves i començar a treballar amb la BD, així que vam assolir l'objectiu i vam poder modificar la informació d'un torn en concret. També ens va servir per observar certs comportaments respecte a la BD i la pàgina web, com per exemple que no es tractava amb el nom de les dependents, sinó que es tractava sempre amb el seu identificador, que passava si s'afegien o s'eliminaven tornos des de la pàgina web, etc.

Objectiu 2. Taula de puntuacions

Semblarà estrany, però en aquest moment no vam començar a implementar el programa que ens assignés els horaris, sinó que vam iniciar el desenvolupament del sistema de puntuacions explicat en [l'apartat 3.2.5](#). Això és degut al fet que aquest programa es podia fer de manera externa des d'un fitxer nou i treballava molt amb la base de dades, així que implementar primer aquest programa, tot i que no era un codi senzill i va requerir bastant temps, ens va servir molt en quant a l'aprenentatge de JavaScript i, sobretot, en relació a l'ús de la informació de la BD.

Durant aquest pas, ens vam trobar amb un munt d'obstacles relacionats amb els aspectes tècnics de l'entorn de programació i tots els errors que comporta:

- **Asincronia + ús de moltes peticions:**

La barrera més gran que ens va tocar resoldre va ser la de l'asincronia. NodeJS ens ofereix un entorn d'execució de JavaScript asíncron, és a dir, quan es fa una petició, per exemple a la base de dades, es crea una 'promesa' (una classe integrada en JavaScript) que permet seguir amb l'execució de codi tot i no haver rebut la resposta a la petició.

Aquest fet aporta grans beneficis en altres àmbits, per exemple una web dinàmica, però en el nostre cas, on volem treballar amb diferents peticions a la vegada doncs ens perjudica, ja que mentre s'estava fent una petició no ens deixava fer una segona consulta a la BD i donava error.

La manera de solucionar aquest tema va ser utilitzar les nomenclatures '*async*' en la definició de les funcions on s'utilitzen les promeses i '*await*' quan es crida a aquestes funcions. D'aquesta manera, totes les funcions on es realitza una petició (promesa) es convertien també en una promesa i gràcies al '*await*' l'execució del programa es parava fins a obtenir una resposta. En definitiva, ho convertíem en un programa que s'executés de manera seqüencial per tal de poder rebre les diferents peticions.

- ***SQL Queries:***

També ens vam trobar bastants obstacles més petits en relació a com tractar amb el format de les diferents taules o com accedir i recórrer en bucle tota la informació que ens donaven. Per exemple, vam utilitzar la funció *format()* per treballar amb tots els formats que pot adquirir una classe *Date*, perquè a vegades convé treballar només amb la data any-mes-dia i a vegades s'ha de treballar amb les hores i minuts. Vam aprendre també a l'ús de noves sentències SQL, per crear una taula, per agrupar un llistat d'atributs, per accedir a tots els noms de les taules, etc.

Finalment, al cap d'unes setmanes i malgrat els obstacles, vam poder assolir l'objectiu i crear aquest sistema de notes que podem aprofitar més endavant.

Objectiu 3. Algorisme d'assignació aleatòria

Sabent ja el funcionament bàsic de com llegir i modificar algunes dades, vam decidir marcar-nos el petit objectiu de realitzar, donats els torns d'un sol dia, una assignació de dependents de manera aleatòria. Aquesta va ser l'heurística número 1, que més endavant vam anar millorant i convertint en noves heurístiques. Gràcies a l'objectiu anterior de la implementació del programa de les puntuacions, aquests següents objectius van ser molt més assequibles i no em vaig trobar tants obstacles.

Objectiu 4. Millorar assignació aleatòria

L'objectiu 4 es basava a millorar relativament l'heurística 1, afegint diferents exploracions, també aleatòries, i que les persones que assigni tinguin una mica de sentit, ja que en l'anterior

pas, s'assignaven persones que estaven a la taula, però no treballaven com a dependents (altres càrrecs o casos de prova).

Aquest pas ens va ajudar molt per a les properes fites, ja que va ser just aquí quan vam iniciar el repositori de GitHub i el vam sincronitzar amb l'entorn d'AWS Cloud9. En els diferents commits² ens vam anar anotant els canvis i millores realitzades en el codi i en aquesta primera pujada al repositori les millores van ser:

- Crear les classes de torn i dependenta i tractar-los en el codi a través de vectors amb els objectes d'aquestes classes, per exemple un vector de dependents.
- També vam organitzar el codi per tal de poder realitzar diferents heurístiques segons convingui, permetent encarar millor el problema des de l'inici.
- Millorar l'heurística 1, en la qual ja vam comprovar la primera restricció, que una dependenta no es repetís en diferents torns. L'algorisme també feia més exploracions (el nombre que li donessis) i, entre les solucions aleatòries que sortien, triava l'opció on la suma de les puntuacions de les dependents fos més alta.
- L'altre millora va ser que ara ja només agafava les dependents de les taules a les quals li assignéssim unes hores de treball, per evitar el problema que hem comentat.

Objectiu 5. Afegir períodes

Com era d'esperar, els horaris aleatoris no sortien molt bé, ficant dependents que treballen al matí a la tarda i viceversa. L'objectiu, doncs, va ser realitzar una nova heurística, la 2, la qual no mirarà cap puntuació de les dependents, afegirà aquestes de manera seqüencial a l'horari i assignarà la primera que compleixi les dues restriccions que tenim fins al moment: que no es repeteixi en el mateix dia i que compleixi el seu període de treball. Per saber el període d'una dependenta, si va de matí o tarda, s'havia d'entrar la dada per a cada dependenta a la pàgina web.

En algunes parts del procés ens vam trobar que havíem de modificar gran part del codi perquè poguéssim continuar progressant i millorant. En aquest cas, ens vam trobar amb un problema en la representació de les dades, ja que quan volíem accedir a una dependenta concreta, per trobar-la havíem de recórrer tot el vector d'objectes. Vam estar analitzant les diferents estructures de dades que ens oferia JavaScript [3] i vam arribar a la conclusió que la millor manera era utilitzar diccionaris (la classe map), amb la qual podíem buscar una dependenta pel seu identificador sense haver de recórrer les altres, cosa que millora notablement l'eficiència.

Les millores d'aquest objectiu van ser:

² commit: un commit realitzat amb Git tracta de guardar un estat d'un programa en un moment determinat, és a dir, et guarda una versió d'un codi per tal que puguis tornar a accedir-hi en cas que es facin canvis.

- Afegir restricció dels períodes, canviant les funcions i atributs corresponents.
- Crear la nova heurística2.
- Canviar tot el programa per a que treballi amb diccionaris i no amb vectors.

En aquest punt ja teníem un esquema base que s'utilitzarà fins al resultat final i funciona de la següent manera:

1. Obtenir les dependents
2. Obtenir la planificació de torns
3. Executar heurística d'assignació de torns
4. Mostrar resultats amb l'horari final

Objectiu 6. Horari setmanal

Fins al moment, teníem que l'horari representava únicament un dia en concret. L'objectiu doncs, era canviar l'estructura del programa perquè tingués en compte els torns de tota la setmana, seguint fent ús de l'heurística 2.

Les millores van ser:

- Canviar l'heurística2 perquè recorri tots els torns de la setmana.
- Crear noves classes que representin l'horari diari i l'horari setmanal, aquest compost per 7 horaris diaris.
- Modificar la funció que obtenia els torns perquè aconseguís tota la setmana, passant per paràmetre una data.

Objectiu 7. Afegir hores setmanals

El següent pas havia de ser que les dependents complissin les hores setmanals que els hi pertocava, així que vam decidir crear una nova heurística 3, la qual tingués en compte la restricció que una dependenta fes les hores que li tocaven.

Respecte a aquest tema hi havia un greu problema, el qual ens acompanya durant tot el procés: com que treballem amb uns torns d'entrada per tal que la persona de recursos humans pugui dir quants treballadors vol a quines hores, no podem obligar-la que calculi les hores dels torns, tasca bastant complexa. En conseqüència, és massa complicat que es compleixin les hores setmanals de manera exacta i mai obtindríem una solució que assolís aquesta restricció. La solució que aplicarem, doncs, treballarà sempre de manera aproximada respecte a les hores setmanals de les dependents, sabent que després pots acabar d'arrodonir l'horari final com convingui.

De moment, en aquesta heurística vam pensar que la restricció fos que les hores que treballa una dependenta siguin sempre menor o igual a les hores contractuals, de manera que es podrien optimitzar recursos i mai faran més hores de les que poden. Amb aquest objectiu, afegim alguns atributs privats de cada dependenta essencials per al nostre algorisme, com les hores acumulades durant l'assignació, la llista de torns assignats a cada dependenta, o els dies que aquesta té ocupats.

També apliquem millores importants en funcionalitats com per exemple la visualització dels resultats o en l'assignació, on fins ara esborràvem una dependenta quan s'assignava i, a partir d'ara, tindrem uns atributs per a poder assignar i desassignar-la.

Objectiu 8. Backtracking

Per primera vegada en el procés introduïm el terme *backtracking*, un algorisme que tracta sobre explorar tot l'espai de solucions fins a trobar una que satisfaci les restriccions. Aquest ens serà molt útil, ja que podrem analitzar tots els possibles horaris, però porta amb si mateix un gran obstacle que ja hem comentat i és que es converteix en un problema *NP-Hard*, on haurem d'explorar molts nodes (solucions) i pot tardar molt temps a resoldre's.

Vam decidir només utilitzar restriccions fortes i complir sempre amb les preferències de les dependents quant al període al qual treballar. Vam observar que nosaltres les restriccions les podíem dividir en finals i intermèdies, ja que la majoria de les restriccions les comprovàvem abans d'assignar una dependenta, per exemple, no assignar la dependenta si ja veiem que superarà les hores totals que ha de fer. Per altra banda, un cop finalitzat l'horari podíem avaluar alguna restricció com que les hores acumulades de cada dependenta s'aproximessin al seu total d'hores (hores contractuals).

Vam estar dubtant entre si realitzar el backtracking de forma diària o setmanal, ja que la mida de l'arbre d'exploració creixia exponencialment quan usàvem tots els torns de la setmana i el programa explotava. Malgrat això, vam decidir explorar tota la setmana en conjunt, ja que si per exemple el divendres et trobes que ja no hi ha solució, és necessari tornar a modificar els dies anteriors per explorar totes les possibilitats.

En definitiva, aquest pas ens va ocupar molt temps i mals de cap, però finalment, vam poder fer una nova heurística 4 amb un algorisme de backtracking, funcionament del qual detallarem en el capítol de resultat final, però en resum va assignant les dependents als diferents torns i torna enrere quan veu que una solució no és vàlida. També teníem la manera ficar les restriccions que volguéssim, a partir d'unes funcions booleanes que retornaven true o false en funció de si es complien.

Objectiu 9. Optimització amb backtracking guiat

En aquest punt, ja teníem un algorisme de backtracking implementat i, en general, un programa bastant ben estructurat que ja ens permetia obtenir solucions al problema en casos senzills. El problema venia quan es complicava una mica i augmentàvem l'entrada, és a dir, casos amb més dependents i torns, en els quals s'havien d'explorar un nombre molt elevat de nodes i l'execució no s'aturava.

Com ja hem comentat en [l'apartat d'introducció 1.5.2](#), hi ha diferents mètodes d'optimització que es podrien utilitzar en aquest cas, com són els mètodes heurístics, els quals ja portem estona mencionant-los perquè realment els estem utilitzant. Per exemple, nosaltres ja estem utilitzant algunes tècniques com les següents:

- **Pruning:**
El 'pruning' és una tècnica utilitzada per descartar solucions parcials una vegada ja es sap que no és una solució vàlida. Nosaltres ja l'estem utilitzant en el moment en que mirem si una dependenta es pot assignar o no i, en cas negatiu, passem a la següent dependenta, deixant d'explorar totes aquelles ramificacions de l'arbre d'exploració.
- **Early Stopping:**
També estem utilitzant l'aturada anticipada, una tècnica molt senzilla que es basa en aturar l'algorisme de backtracking tan bon punt trobem una solució vàlida.

L'objectiu 9 era modificar l'heurística per a obtenir un backtracking guiat, és a dir, intentar conduir l'algorisme cap a les branques de l'arbre que considerem més prometedores, optimitzant un o més paràmetres. Per aconseguir això, vam crear una heurística 5, que és igual que l'anterior, però a cada exploració reordena les dependents a partir de les hores acumulades, a més a més, utilitzem també les notes per ordenar-les en cas d'empat. D'aquesta manera s'assoleix guiar l'algorisme perquè agafi primer les dependents amb menys hores assignades, fet que millora molt l'eficiència.

Objectiu 10. Afegir millores

En aquest pas, ens trobem ja en un punt on tenim bastant avançat el projecte; tot i això encara hi ha un gran ventall de possibles millores. Ens vam proposar seguir investigant de quina manera podíem aconseguir uns millors resultats respecte a l'horari o l'eficiència i es van implementar millores com:

- Modificar tot lo referent als dies ocupats de cada dependenta, per a que no treballi els 7 dies.
- Afegir control d'errors a totes les funcions per millorar la depuració de codi.

- Afegir la comprovació d'unes condicions inicials abans de començar l'execució del backtracking, sobretot per casos on les hores totals dels torns siguin més elevades que les hores que poden aconseguir treballar les dependents.

Objectiu 11. Noves heurístiques

A part d'anar afegint noves millores, també era moment de provar noves propostes d'heurístiques canviant algun detall, així que vam afegir l'heurística 6. Aquesta tindrà un nou backtracking guiat amb una funció d'ordenació diferent, la qual ordena segons un nou paràmetre que són les hores que deu la dependent a l'empresa o viceversa.

Aquesta opció intenta solucionar el problema de botigues on hi hagi poques dependents i moltes hores a omplir, botigues on normalment s'utilitza una bossa d'hores per intentar aprofitar al màxim els pocs recursos. Cal esmentar que el seu ús estaria subjecte a crear una nova taula a la base de dades que calcules les hores que es deuen entre l'empresa i el treballador.

Malauradament, i durant tota l'etapa d'implementació, no tots els intents van resultar positius i també va haver-hi moltes propostes de codi fallides que s'havien d'acabar esborrant. Alguns dels intents fallits van ser els següents:

- Vam intentar crear una heurística 7, ja que ens vam plantejar que si en comptes de recórrer el bucle de les dependents, primer recorriem els torns podríem plantejar un arbre totalment diferent amb menys nodes, pel fet que sempre teníem més torns que dependents. Finalment, amb algunes proves, vam veure que aquest arbre no era possible i que acabàvem obtenint el mateix arbre però explorant els nodes d'una manera menys òptima.
- Addicionalment, vam intentar fer que no només ens calcules una solució, si no varies. És a dir, que per exemple l'algorisme ens oferís 3 solucions per triar, encara que finalment es va cancel·lar la proposta perquè teníem problemes respecte a la complexitat temporal i aquesta característica ho empitjorava i el programa no acabava mai.
- També vam provar de separar els matins i les tardes, ja que en moltes empreses podia passar que les dependents sempre anessin de matins o de tardes i no es mesclassin. Tot i això, no era possible perquè el matí el calculava molt ràpid, però si la tarda fallava no hi havia forma de tornar al matí i no trobava solució, inclús en casos senzills on ajuntant matí i tarda adquiríem la solució ràpidament.
- Una altra possible idea era que l'usuari que crea els torns, tingués com a restricció que les hores totals sumant tots els torns no diferissin molt de les hores totals sumant totes les dependents, però vam veure que aquest fet li treia valor i utilitat al programa.

Objectiu 12. Constraint programming

L'altre mètode d'optimització totalment diferent de les heurístiques és la programació de restriccions o *constraint programming*, la qual hem explicat també en [l'apartat 1.5.2](#).

Aconseguir aquest objectiu també ens va ocupar una gran quantitat de temps, ja que s'havia d'implementar un codi completament nou que no tenia a veure amb la programació feta fins ara. Malgrat això era una cosa necessària que s'havia d'intentar, ja que podia millorar molt els temes d'eficiència.

Així doncs, vam poder implementar un *SAT Solver*, per al qual vam necessitar la llibreria de node *logic-solver*. L'explicació del algorisme es detallarà en el següent apartat.

Objectiu 13. Altres millores

Per últim, vam continuar aplicant noves millores necessàries al programa:

- Afegir funcions i atributs respecte als dies festius i les vacances de cada dependenta, no permetent que s'assignin en un dia que no poden treballar.
- Modificar que tot funcioni a partir d'una data que rebrà el programa per paràmetre, la qual representarà el primer dia de la setmana de la qual es vol generar un horari.
- Solucionar errors en general del sistema, per exemple, respecte les dades, vam haver d'arreglar que quan tractàvem amb una setmana on es canviava el valor del mes o l'any no es tenien en compte tots els dies.
- Afegir un màxim de temps d'execució, per als casos complexos on l'algorisme no troba solució i no s'atura mai .

3.3.2. Resultat final

Una vegada hem vist tot el procés de desenvolupament del projecte, en darrera instància, mostrarem el resultat final de la implementació.

Obviant altres fitxers de proves i de l'entorn de AWS Cloud9, Git i NodeJs, el codi font implementat per nosaltres es constitueix dels següents 5 fitxers:

- valoracioMensual.js
- main.js
- Dependenta.js
- Turno.js
- HorariSet.js

A continuació, detallarem cadascun d'aquests fitxers. Explicar cada petit detall del codi seria aclaparador i carregós i tampoc provocaria gran diferència a la comprensió del projecte, així doncs, simplement detallarem els diferents atributs i objectes i mostrarem cada funció del codi, a partir de l'entrada, la sortida i la seva descripció.

VALORACIÓ MENSUAL:

El fitxer `valoracioMensual.js` s'encarrega d'executar el sistema de puntuacions que hem comentat anteriorment i és totalment independent dels demés fitxers.

Aquest programa només es pot executar una vegada cada mes amb l'objectiu de generar una taula a la base de dades amb les puntuacions de cada dependent respecte els mesos anteriors, per més detalls, cal llegir [l'apartat 3.2.5](#). En primer lloc, mostrarem la funció principal, la qual utilitza diferents funcions auxiliars que també explicarem. Cada funció s'executa dins d'un *try and catch*, un mecanisme que ens ofereix JavaScript amb l'objectiu de controlar els errors i, en cas que es doni un error, saber quina funció ha fallat.

❖ Funció `valoracioMensual()`:

Entrada: Cap

Sortida: Cap

Descripció:

És la funció principal del programa, la qual s'hauria de cridar un cop al mes i s'encarrega de crear la taula amb les puntuacions de cada dependent. El mes i l'any els agafem en un valor global en format `Data`, agafant la data exacta en la qual s'executa el programa i que podran utilitzar totes les altres funcions. El procediment és senzill i es pot veure el seu funcionament de forma clara en la figura 26.

En primer lloc, ens connectem amb la base de dades i creem la taula de les puntuacions d'aquell mes a la BD. A continuació, obtenim les primeres dades necessàries, una llista dels treballadors i el tiquet mitjà de la botiga. A partir d'aquí, recorrem la llista de treballadors i, per cadascun d'ells, obtenim les seves puntuacions (puntualitat, tiquet, valoració i acumulat) i afegim una fila a la taula. Un cop acabat el bucle, sumem el punt o estrella extra als 3 millors treballadors.

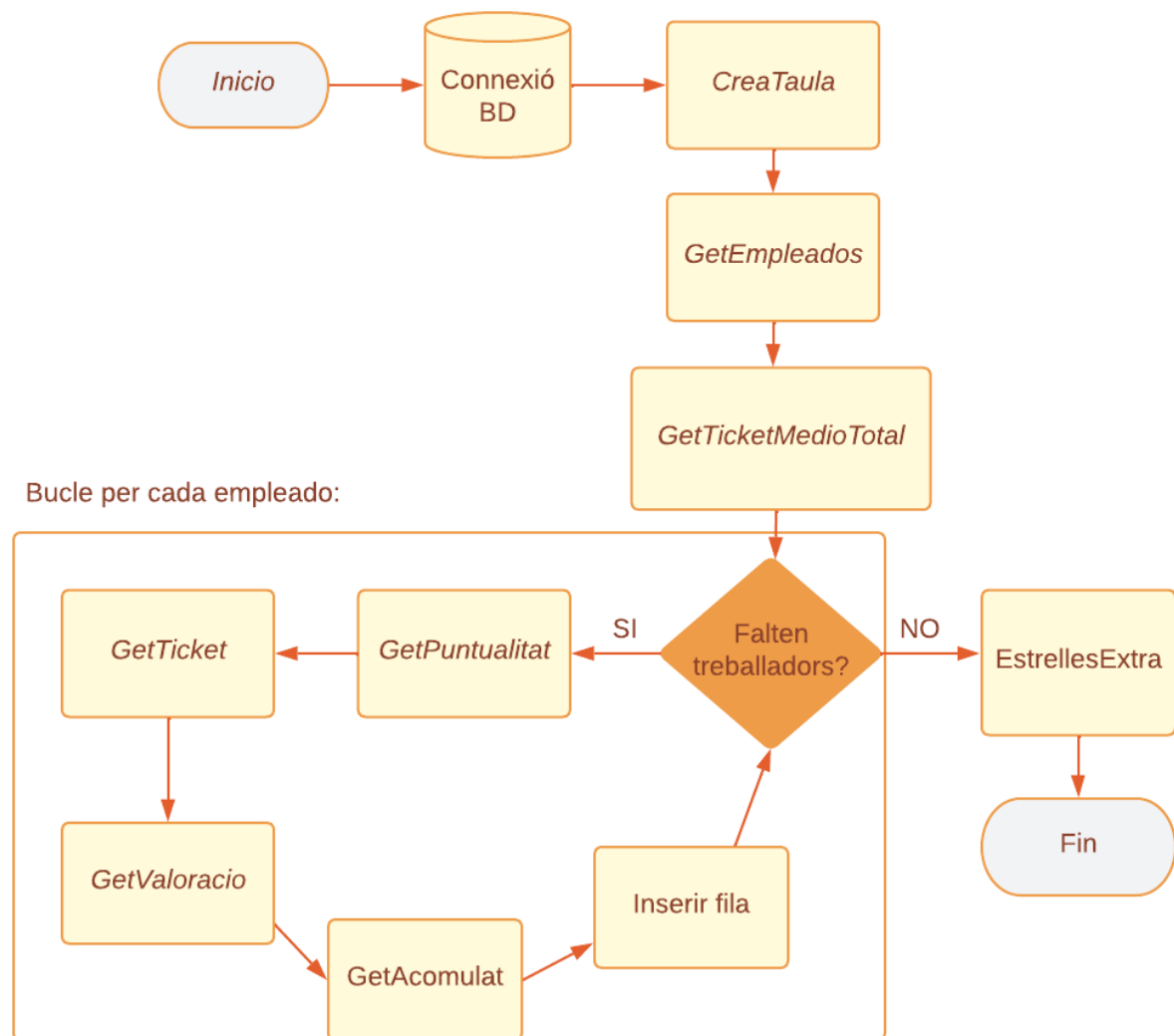


Figure 26. Diagrama de flux de la funció valoracioMensual().

❖ **Funció creaTaula():**

Entrada: Cap

Sortida: Cap

Descripció:

Executa una única petició a la base de dades amb el llenguatge SQL amb l'objectiu de, en cas que no existeixi, crear la taula mensual de puntuacions. En la següent figura podem observar com es genera aquesta petició amb SQL dins de JavaScript.

```

const sql = `IF OBJECT_ID(N'dbo.cdpPuntuacioMensual_${ANY}_${MES}', N'U') IS NULL
BEGIN
CREATE TABLE dbo.cdpPuntuacioMensual_${ANY}_${MES}(
  "idEmpleado" INT NOT NULL,
  "puntualitat" FLOAT,
  "ticket" FLOAT,
  "valoracio" FLOAT,
  "acomulat" FLOAT
);
END;`;

```

Figure 27. Petició SQL per a crear la taula de puntuacions.

❖ Funció getEmpleados():

Entrada: Cap

Sortida: Vector amb els codis dels treballadors

Descripció:

Envia una petició a la BD per obtenir els codis de totes les dependents i et retorna un llistat (vector) amb aquests, per exemple: [8,9,10,14,15,16,17]. Aquests codis els necessitarem a les següents funcions per a trobar les dependents en les altres taules de la BD a partir dels seus identificadors.

❖ Funció getTurnos():

Entrada: Cap

Sortida: Diccionari amb la informació dels torns

Descripció:

Accedeix a la taula de Turnos de la BD i crea i retorna un diccionari <clau, valor> on la clau és l'identificador del torn i el valor és una dupla amb la hora d'inici i fi d'aquest. Per exemple, una entrada del diccionari seria: < 'A38F..' , ['8:00', '14:00'] >

❖ **Funció getTicketMedioTotal():**

Entrada: Cap

Sortida: Float amb el valor mitjà dels tiquets de la botiga en aquell mes

Descripció:

Fa una consulta a la taula de vendes de la BD agrupant la columna 'Num_tick', ja que un mateix tiquet pot tenir productes diferents, i sumant els imports. Llavors només cal dividir entre en nombre de tiquets agrupats i obtenim la mitjana.

❖ **Funció getPuntualitatEmpleado():**

Entrada: Codi identificador del treballador

Sortida: Valor float que representa la puntualitat del treballador

Descripció:

L'objectiu d'aquesta funció és aconseguir la puntualitat, que com hem comentat, la representem com un valor amb els minuts de retard que arriba el treballador que li passes per paràmetre, aquest retard també pot ser negatiu, significat que arriba d'hora.

Sembla fàcil però aquesta funció és bastant complexa i utilitza moltes crides a la BD, ja que hem d'agafar els horaris teòrics de cada setmana i comparar-los amb els reals. Per resumir-ho de manera clara, la funció segueix els següents passos:

- Obtenim les taules de planificació setmanals del mes en concret usant la funció getTaulesMes().
- A partir de la funció getTurnos() aconseguim les hores inici i fi de cada torn.
- Fem un primer bucle recorrent les taules setmanals d'aquell mes, per cada setmana:
 - o Accedim a la taula Planificacion de la BD, consultant només els torns teòrics d'aquell treballador passat per paràmetre.
 - o Fem una matriu on cada fila és un d'aquests torns teòric.
 - o Fem un segon bucle recorrent aquesta matriu i, per cada torn teòric:
 - Accedim a la taula DadesFichador de la BD
 - Comparem les hores del torn teòric amb el real i anem afegint els retards a un vector de tantes posicions com torns.
- Finalment, fem la mitjana del vector i la retornem.

❖ **Funció getTicketEmpleado():**

Entrada: Codi identificador del treballador

Sortida: Tiquet mitjà del treballador

Descripció:

L'objectiu d'aquesta funció és obtenir el tiquet mitjà d'un treballador durant aquell mes. Funciona igual que getTicketMedio total però afegint a la crida a la BD una condició per a que els tiquets siguin només de la dependenta corresponent.

❖ **Funció getValoracio():**

Entrada: Valors amb puntualitat, ticket i ticketMedioTotal

Sortida: Valoració

Descripció:

Et calcula la valoració d'un treballador a partir de la seva puntualitat, el tiquet mitjà i la mitjana de tiquet de la botiga, utilitzant el mètode explicat en [l'apartat 3.2.5](#).

❖ **Funció getAcumulatEmpleado():**

Entrada: Id del treballador i la valoració del mes actual

Sortida: Valoració acumulada

Descripció:

La funció accedeix a les taules de puntuacions de la BD dels mesos anteriors, concretament a les del treballador que li passes per paràmetre, i fa una mitjana entre la valoració del mes actual i les dels darrers mesos.

❖ **Funció inserirFilaPuntuacio():**

Entrada: IdEmpleado, puntualitat, ticket, valoració i acumulat

Sortida: Cap

Descripció:

A partir d'un identificador de treballador passat per paràmetre, a més a més de les seves respectives puntuacions, s'insereix una nova fila en la taula PuntuacioMensual a través d'una sentència SQL.

❖ Funció estrellesExtra():

Entrada: Cap

Sortida: Cap

Descripció:

S'executa una vegada ja s'han creat totes les valoracions dels diferents treballadors i simplement mira quins són els 3 millors empleats respecte al tiquet mitjà i els hi afegeix una estrella o punt.

❖ Funció mitjaVector():

Entrada: Vector de int/float

Sortida: Mitja del vector

Descripció:

Simplement li passes un vector i et retorna la seva mitja.

❖ Funció compararHores():

Entrada: String horaTeorica i string horaReal

Sortida: Nombre de minuts de retard

Descripció:

A aquesta funció li passes en format string les hores teòrica i real i et retorna els minuts de comparació, és a dir, els minuts que la persona ha arribat tard o d'hora. També he tingut en compte que si aquesta comparació és més gran que mitja hora, no es tindrà en compte, ja que

seria un cas extrem on la persona ha patit un imprevist, està malalta, etc. S'utilitza durant la funció que calculem la puntualitat, si per exemple, li passes una entrada de '08:00' i '07:48', el resultat retorna -12.

❖ **Funció getTaulesDelMes():**

Entrada: Cap

Sortida: Vector de noms de les taules de planificació del mes actual

Descripció:

Aquesta funció s'utilitza dins de la funció getPuntualitatEmpleado() i la fem servir per obtenir els noms de les diferents taules de la BD on es mostren els horaris teòrics de cada setmana. Per aconseguir-les s'accedeix a l'esquema d'informació de la pròpia BD.

GENERADOR D'HORARIS:

El programa generador d'horaris principal està implementat en el fitxer main.js. Com ja hem comentat, aquest codi només fa servir dues llibreries npm, que són 'mssql' i 'logic-solver'. El nostre programa també requereix tres classes d'objectes, les quals constitueixen els altres fitxers. En primer lloc, mostrarem aquestes classes junt amb els seus atributs i, tot seguit, veurem les diferents funcions implementades i la seva descripció. A més a més, aquestes funcions les hem dividit quatre grups:

- la funció base principal
- les funcions heurístiques que realitzen l'assignació de torns
- les funcions que accedeixen a la base de dades per llegir o modificar-la
- les funcions auxiliars que utilitzen les anteriors

CLASSES:

❖ **Classe Dependenta:**

Aquesta classe representa a una dependenta de la botiga i està formada per diferents atributs que mostrem en la següent taula:

<i>Atribut</i>	<i>Descripció</i>	<i>Exemple</i>
codi	Identificador de la dependenta	3
nom	Nom de la dependenta	'Marta'
horesSet	Hores setmanals que marca el contracte	35
tipusHorari	Vector de strings amb 7 posicions que representa la preferència horària de cada dia de la setmana	[M, M, M, T, T, A, A]
horesAcomulat	Hores que porta acumulades l'algorisme en un cert punt de l'execució	16
llistaTurnos	Vector dels torns assignats a la dependenta	[Torn1, Torn3, Torn 8]
diesOcupats	Vector de booleans amb 7 posicions que representa els dies de la setmana que estan assignats	[true, false, false, true, false, false, true]
numDiesTreballats	Nombre de dies que la dependenta te assignats	3
nota	Nota de la dependenta entre 1 i 5	4
horesDeute	Hores que deu l'empresa a la dependenta o viceversa	-8

Taula 16. Classe Dependentat

❖ **Classe Turno:**

Aquesta classe representa un torn de treball i consta dels següents atributs:

<i>Atribut</i>	<i>Descripció</i>	<i>Exemple</i>
idPlan	Identificador del torn	'{A38F..}'
fecha	Any, mes i dia que es fa el torn	2019-11-03
diaSetmana	Valor del 0 al 6 que representa el dia de la setmana	2 (dimecres)
periode	Període de matí o de tarda	'M'
hi	Hora inicial del torn	'08:00'
hf	Hora final del torn	'14:00'
totalHores	Total d'hores que hi ha entre hi i hf	6
idEmpleado	Identificador de la dependenta que s'assigna al torn	3

Taula 17. Classe Turno

❖ **Classe HorariSet:**

Aquesta classe representa l'horari planificat d'una setmana en concret i únicament consta d'un atribut anomenat **dies**. Aquest és un objecte de JavaScript que té els seus propis atributs representats del 0 al 6, els quals representen els dies de la setmana, el 0 és dilluns i el 6 és diumenge. Cadascun d'aquests atributs és en si mateix un vector format per torns, així doncs, aquesta classe 'HorariSet' es podria considerar com una gran matriu que representa tots els torns d'una setmana dividits en dies.

FUNCIO BASE:

❖ **Funció main():**

Entrada: Data

Sortida: Cap

Descripció:

Aquesta és la funció base del programa i és la funció principal que es cridarà un cop l'usuari pressionari el botó de generar l'horari. La funció rep solament un paràmetre d'entrada en format Data, el qual representa el primer dia d'una setmana, i s'encarrega de generar l'horari d'aquella setmana.

La funció s'estructura d'una manera molt senzilla. Abans de tot, ens connectem a la base de dades i convertim l'any, el mes i el dia en variables globals, per a que totes les demés funcions puguin accedir-hi. A partir d'aquí estructurarem la funció en 4 passos fonamentals: obtenir les dependents, obtenir els torns, executar la funció heurística que nosaltres considerem, la qual realitza l'assignació dels torns i ens genera un horari i, finalment, visualitzem el resultat de l'horari. També afegim dues mesures de temps just abans i després de cridar la funció heurística, per tal de poder veure l'eficiència computacional. A la següent figura mostrem el mapa conceptual amb l'estructura base.

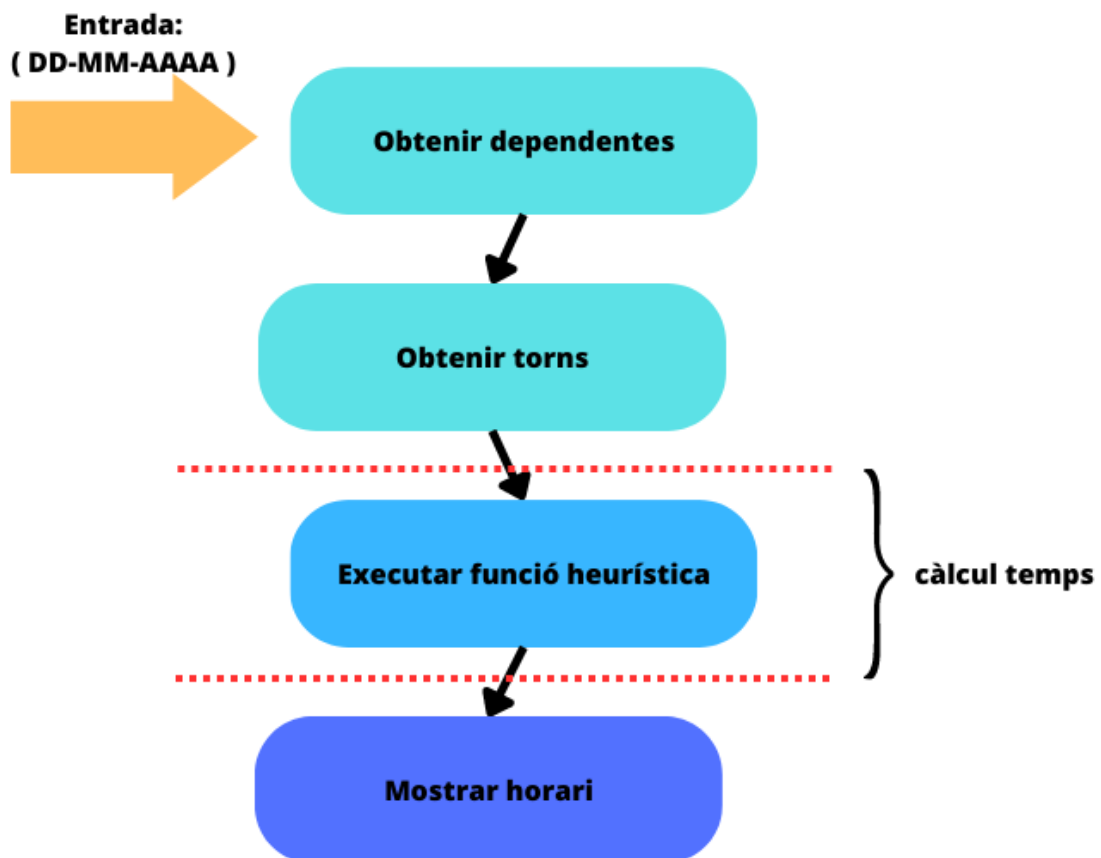


Figure 28. Mapa conceptual de la funció main()

FUNCIONS BASE DE DADES:

❖ Funció getDependents():

Entrada: Cap

Sortida: Diccionari de dependents

Descripció:

Aquesta funció és el primer pas bàsic de la funció main() i s'utilitza per obtenir les dades de les dependents introduïdes per l'usuari. En primer lloc, fa una lectura de la taula Dependents per llegir els identificadors i el nom i, a partir d'aquí, per cada dependenta, llegeix la informació addicional com les hores totals o les preferències de la taula DependentsExtes i es crea un objecte de la classe Dependenta, el qual s'insereix al diccionari. Així doncs, en el diccionari

que es retorna, a partir d'un id de la dependenta pots obtenir l'objecte que la representa. Finalment, cridem a la funció `afegeixFestius()` que expliquem tot seguit.

❖ **Funció `afegeixFestius()`:**

Entrada: Diccionari de dependents

Sortida: Cap

Descripció:

A partir del diccionari de dependents que li passes, aquesta funció s'encarrega de llegir la taula `CalendariLaboral` i, a través de l'atribut `diesOcupats`, afegim els dies festius de cada dependent, ja sigui per vacances, baixa laboral, etc., per a que l'algorisme els tingui en compte i no els pugui assignar.

❖ **Funció `getHorariSet()`:**

Entrada: Cap

Sortida: objecte de la classe `HorariSet`

Descripció:

Aquesta funció és el segon pas bàsic, en el qual s'obté tota la informació respecte els torns. És molt senzilla ja que només treballa englobant la setmana, així doncs, crea un objecte `HorariSet` que retornarà, i per cada dia de la setmana crida a la següent funció `getPlansLlista` per a que ompli els torns d'aquell dia.

❖ **Funció `getPlansLlista()`:**

Entrada: Dia de la setmana [0-6]

Sortida: Llista amb els torns planificats del respectiu dia de la setmana

Descripció:

Com ja hem dit, aquesta funció es crida per cada dia i s'encarrega de retornar una llista dels torns planificats per aquell dia. Li passes per paràmetre un valor que representa el dia de la setmana del 0 al 6, el qual es sumarà a la variable global que representa el primer dia de la setmana. Tot i això, hem hagut de tenir molt en compte les transicions entre mesos o anys. En

definitiva, es crea aquesta llista d'objectes de la classe Turno amb tota la informació com les hores, el període, etc., menys el idEmpleado, el qual haurem d'assignar més endavant.

❖ **Funció assignarEmpleado():**

Entrada: Identificador de la dependenta i del torn planificat

Sortida: Cap

Descripció:

Aquesta funció s'encarrega de modificar la base de dades assignant una dependenta a un torn planificat, ambdós passats per paràmetre amb l'identificador corresponent. S'hauria de cridar per cada torn planificat una vegada s'ha generat l'horari.

❖ **Funció getTurnos():** idèntica a la que hem explicat en la valoració Mensual

FUNCIONS AUXILIARS:

Les funcions auxiliars són funcions secundàries usades per altres funcions i penso que algunes no aporten res a la comprensió d'aquest apartat, així que només veurem algunes de les funcions més importants.

❖ **Funció assignarDependentia():**

Entrada: Objectes representant una dependenta i un torn

Sortida: Cap

Descripció:

Aquesta funció rep per paràmetre una dependenta i un torn i s'encarrega d'assignar la dependenta al torn, modificant els atributs corresponents, com les hores acumulades, els dies ocupats o el nombre de dies treballats.

Per altra banda, existeix la mateixa funció però contrària desasignarDependentia(), la qual restableix els atributs modificats. Aquestes funcions s'utilitzen sobretot en el backtracking, per tal de provar totes les possibilitats.

❖ **Funció compleixCondicionsInicials():**

Entrada: Diccionari de dependents i llista dels torns

Sortida: Cap

Descripció:

Aquesta funció s'utilitza per saber si un problema compleix certes condicions inicials, ja que si per exemple no es compleix que les hores dels treballadors puguin cobrir totes les hores dels diferents torns planificats, no fa falta seguir executant l'heurística, perquè ja sabem que no hi haurà solució.

❖ **Funció assignacioValida():**

Entrada: Objectes representant una dependenta i un torn

Sortida: Bool que ens diu si una assignació és vàlida o no

Descripció:

Aquesta funció té una gran importància, ja que té l'objectiu de comprovar les restriccions del problema. La funció rep per paràmetre una dependenta i un torn, comprova les restriccions que nosaltres trobem convenients, com per exemple mirar que la dependenta no treballi aquell dia, que no es passi de les hores totals del contracte, que compleixi les preferències, etc. Retornarem True en cas que les restriccions es compleixin i False en cas contrari.

❖ **Funció horariCorrecte():**

Entrada: Llista d'assignacions (estat)

Sortida: Bool que ens diu si un horari és correcte o no

Descripció:

Les restriccions que hem vist en la funció anterior són intermèdies, però també podem tenir restriccions sobre el resultat final, com per exemple que les hores acumulades de cada dependenta siguin semblants a les totals que han de fet. Així doncs, aquesta funció rep un possible estat final del backtracking i retorna cert o fals en funció de si es correcte o no.

FUNCIONS HEURÍSTIQUES:

Les funcions heurístiques són el pas més important del programa, ja que són les que s'encarreguen de realitzar l'assignació horària. Com ja hem comentat, es van anar provant diferents mètodes i millores, així que algunes heurístiques són similars i altres són molt diferents, però en general totes segueixen la mateixa estructura i funcionalitat:

❖ Funció heurísticaX():

Entrada: Diccionari de dependents i objecte de la classe HorariSet

Sortida: Cap

Descripció:

Tot i que cada heurística utilitza mètodes diferents, totes tenen en comú que reben per paràmetre un diccionari amb totes les dependents i la seva informació i un objecte de la classe HorariSet amb tota la informació respecte als torns planificats. A partir d'aquí es produeix l'assignació horària usant diferents tècniques, com el backtracking o el constraint programming.

3.3.3. Backtracking i constraint programming

En aquest apartat, degut a la seva importància en aquest projecte, hem separat el resultat final de les implementacions de les tècniques principals usades, les quals detallem a continuació.

BACKTRACKING:

En algunes funcions heurístiques hem implementat un backtracking amb l'objectiu de recórrer totes les possibles assignacions, comprovant les restriccions i fins a trobar una solució final que les compleixi.

Per veure-ho de forma clara, podem pensar en el backtracking com en l'exploració d'un arbre de nodes, on cada node és un estat de la solució parcial i els nodes finals o nodes fulla serà una possible solució al problema. Nosaltres ho hem plantejat de la següent manera:

Cada estat l'hem representat de la forma més senzilla possible i és bàsicament un vector amb la mateixa mida que el nombre de torns. Així doncs, a cada posició del vector ficarem els codis de les dependents assignades. Un node tindrà varis fills representant les dependents i cada nivell de l'arbre tindrà certs torns assignats, fins a arribar als nodes fulla, on tots els torns ja estaran assignats. En la següent figura podem observar com quedaria:

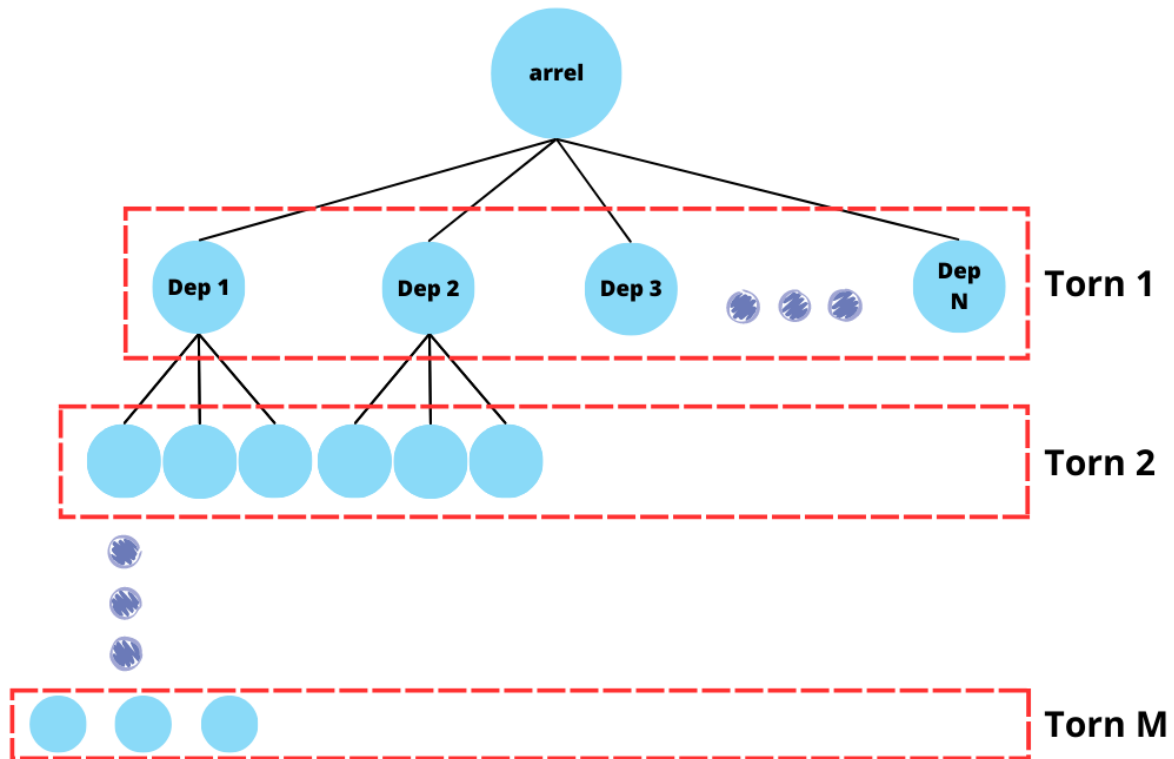


Figure 29. Arbre d'exploracions del backtracking

Tornem a centrar l'atenció en la complexitat computacional, ja que per explorar tot el domini d'estats de l'arbre, aquest va creixent de manera exponencial (problema NP-Hard). En concret, la mida d'un arbre es calcula: [57]

$$\text{num nodes} = \frac{k^{h+1} - 1}{k - 1}$$

On k és el nombre de fills d'un node i h és l'altura de l'arbre.

Podem observar com creix molt ràpidament, per exemple en un problema amb 4 dependents i 15 torns de treball, l'arbre tindrà un conjunt de més de mil milions de nodes, en concret 1.431.655.765 nodes. I a mesura que s'afegeixen dependents o torns de treball genera uns valors molt elevats. Per aquest motiu i com ja hem comentat, s'ha de fer ús de certs mètodes

d'optimització. Per exemple, un detall que veurem a continuació, és que si un torn ja està assignat no l'explorem, cosa que ens estalvia moltíssims nodes.

A la pràctica és una mica més complex, ja que s'ha hagut d'implementar una funció recursiva. Així doncs, la nostra heurística, en primer lloc, crea un estat buit, amb tot valors *null*, i un comptador de torns assignats i llavors crida a la funció recursiva `backtrack()`, que per entendre-la fàcilment, aquest és el seu pseudocodi:

```
backtrack(estat):
  // Cas base
  si tots els torns estan assignats:
    si l'horari és correcte:
      retorna estat
    sinó:
      retorna null

  // Recorrem els torns
  per cada torn:
    si el torn no està assignat:
      // Recorrem les dependents
      per cada dependenta:
        si la dependenta pot ser assignada al torn:
          assigna dependenta al torn
          incrementa el comptador de torns assignats
          // Recursivament crida backtrack per continuar l'exploració
          solucio = backtrack(estat)
          si solucio és diferent de null:
            retorna solucio
          sinó:
            desassigna la dependenta del torn
            decrementa el comptador de trons assignats
        retorna null
  retorna null
```

Respecte al backtracking guiat, simplement ha fet falta afegir una funció d'ordenació de les dependents, que es cridi en cada node. Així doncs, abans de començar a recórrer els bucles del backtracking, ordenem a les dependents segons certs criteris.

Adicionalment, hem afegit petits detalls relacionats amb l'eficiència i que usarem en l'apartat de proves, com el càlcul dels nodes explorats i del temps d'execució de l'heurística. Per tal que el programa no quedi executant-se indefinidament, hem afegit un màxim de temps que es pot variar segons convingui, de moment nosaltres deixarem que es pugui executar durant 3 minuts.

CONSTRAINT PROGRAMMING:

L'altra tècnica usada per solucionar el problema és la programació de restriccions. La teoria ja l'hem explicada en [l'apartat 1.5.2](#), així que passarem directament a explicar com s'ha implementat.

Aquesta part d'implementació ha estat força complexa i ha requerit bastant temps, ja que hem hagut d'utilitzar una llibreria de NodeJS anomenada *logic-solver*, de la qual hi ha molt poca informació i només disposàvem de la seva pròpia pàgina de documentació. Hem estructurat el codi en uns passos bàsics, com podem observar en la següent figura:

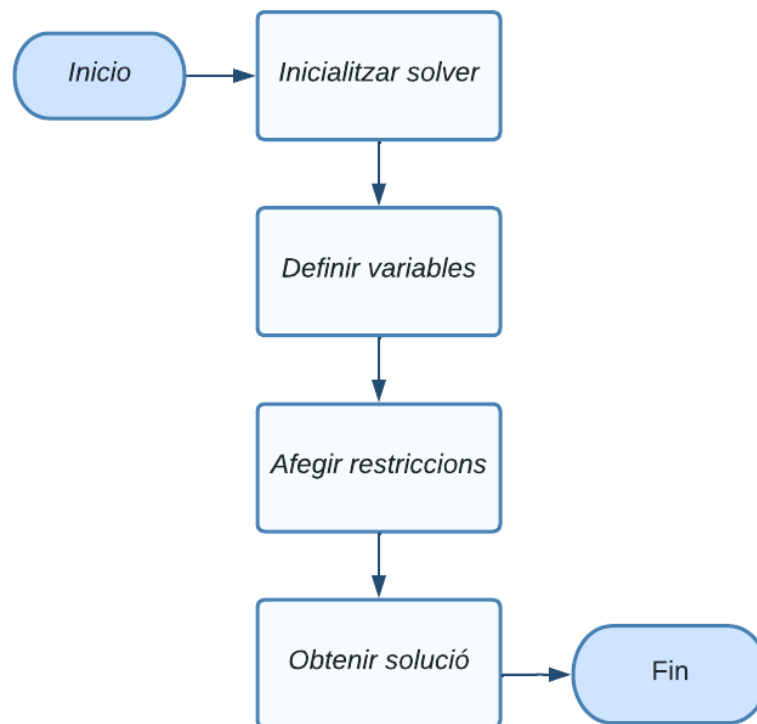


Figure 30. Diagrama de flux de la programació de restriccions.

❖ Inicialitzar solver:

En primera instància, creem una nova variable que serà un objecte de la classe *Solver*, la qual ens ofereix la llibreria *logic-solver*. Aquesta llibreria conté un MiniSat, un *SAT solver* de codi obert que ens ofereix diversos atributs i funcions per treballar amb el problema de satisfacció de restriccions.

❖ Definir variables:

Com ja hem vist en les anteriors funcions, nosaltres utilitzem variables que representen objectes amb les dades de les dependents, els torns, etc. Però en aquest tipus de metodologia, a més a més, hem de definir unes noves variables en relació amb el problema. Aquestes són més complicades de representar, ja que només poden ser valors bool, cert o fals, i han de representar tot el domini del problema. En el cas de les variables numèriques, la llibreria ens aporta l'opció d'utilitzar bits, valors 0 o 1 per a representar els nombres en binari.

En el cas del problema dels horaris amb tantes possibilitats, hem hagut de crear un gran nombre d'aquestes variables booleanes i les hem definit de la següent manera: cada variable representarà un string amb l'identificador del torn junt amb l'identificador de la dependenta i haurem de cobrir totes les possibles combinacions. Per exemple, la variable 'E38S_5' podrà ser certa o falsa i, en cas que sigui certa, significarà que la dependenta 5 s'ha assignat al torn planificat 'E38S'.

En definitiva, s'han de crear moltes variables d'aquest tipus i, en la següent etapa, s'hauran de crear restriccions a partir d'aquestes així que hem creat el següent sistema de variables per a que ens sigui lo més còmode possible:

- **variables:**

Les variables principals es representaran dins d'una matriu, on cada fila és un torn i cada columna és una dependenta. Per ficar els exemples utilitzarem un cas on hi ha 3 dependents i 4 torns, en aquest cas tindríem la següent matriu:

T1_D1	T1_D2	T1_D3
T2_D1	T2_D2	T2_D3
T3_D1	T3_D2	T3_D3
T4_D1	T4_D2	T4_D3

- **variablesT:**

Veurem més endavant com també necessitem la matriu variables transposada, per poder treballar tant amb dependents com amb torns. Així doncs, ara les files seran les dependents i l'exemple ens quedaria així:

T1_D1	T2_D1	T3_D1	T4_D1
T1_D2	T2_D2	T3_D2	T4_D2
T1_D3	T2_D3	T3_D3	T4_D3

- **variablesPerDia i variablesTPerDia:**

Aquestes dues variables s’han creat per poder treballar amb dies per separat i són una llista de les matrius anteriors separades per dies. Tot i que treballem amb matrius de 3 dimensions, a la pràctica és més senzill, per exemple, en el cas de variablesTPerDia, serà una llista de 7 posicions i cada posició tindrà les matrius anteriors, però separant els torns per dies:

variablesT dia 0	variablesT dia 1	variablesT dia 2	variablesT dia 3	variablesT dia 4	variablesT dia 5	variablesT dia 6
---------------------	---------------------	---------------------	---------------------	---------------------	---------------------	---------------------

- **horesDependentes:**

És una llista de mida nombre de dependents i cada posició té les hores totals que pot fer una dependenta. Cal tenir en compte que cada valor vindrà representat per valors binaris, per exemple:

1000 (8)	1010 (10)	110 (6)
----------	-----------	---------

- **weights:**

És una llista de mida nombre de torns on cada posició representa les hores que ocupa el torn. Aquest vector s’utilitzarà dins d’una funció de la llibreria com a pesos i no cal que estiguin en binari, tot i que la funció no pot treballar amb decimals, així que aquestes hores sempre estaran arrodonides.

5	5	6	4
---	---	---	---

❖ **Afegir restriccions:**

En aquest tipus de problemes, les restriccions són la peça fonamental i s’han de representar com un conjunt de formules lògiques, les quals utilitzaran les operacions lògiques bàsiques com NOT, AND i OR, etc. Les restriccions que hem aplicat en la nostre implementació són:

- **Restricció 1:**

“Un torn només pot tenir una dependenta assignada.”

S’utilitza la clàusula *exactlyOne()*, la qual ens la proporciona la llibreria i significa que d’una llista de variables, només una pot ser certa, i se li passen les files de ‘variables’.

Aquestes clàusules s'hauran de cridar en bucle i n'hi haurà moltes, així que donem només un primer exemple, on el torn1 només pot assignar-se a una de les tres dependents.

La crida seria:

$$\text{exactlyOne} ([T1_D1, T1_D2, T1_D3])$$

i les fórmules lògiques serien les següents:

$$\begin{aligned} &(T1_D1 \text{ OR } T1_D2 \text{ OR } T1_D3) \\ &\text{AND NOT}(T1_D1 \text{ AND } T1_D2) \\ &\text{AND NOT}(T1_D1 \text{ AND } T1_D3) \\ &\text{AND NOT}(T1_D2 \text{ AND } T1_D3) \end{aligned}$$

- **Restricció 2:**

“Una dependenta com a molt té assignat un torn al dia.”

En aquest cas, recorrem la matriu tridimensional de variablesTPerDia i utilitzem la clàusula atMostOne().

Per exemple, si el dia té el torn1 i el torn2 tindrem:

$$\text{atMostOne} ([T1_D1, T2_D1])$$

```
//una dependenta com a molt te un torn al dia
for (let d = 0; d < 7; ++d) {
  for (let i = 0; i < variablesTPerDia[d].length; i++) {
    solver.require( Logic.atMostOne( variablesTPerDia[d][i] ) );
  }
}
```

Figure 31. Restricció 2 implementada en JavaScript i utilitzant 'logic-solver'

- **Restricció 3:**

“Una dependenta no pot treballar en un torn on no coincideixin els períodes.”

En aquest cas, utilitzem la variable 'variablesPerDia' per tal de revisar totes les variables i comprovar si els períodes de matí o tarda de la dependenta i el torn coincideixen, en cas que no coincideixin ho convertim directament en fals.

Per exemple, si veiem que el torn 1 és de tarda i la dependenta 1 vol anar de matí, afegirem la clàusula:

$$\text{NOT} (T1_D1)$$

- **Restricció 4:**

“Una dependenta no pot treballar els 7 dies de la setmana.”

Per cada dependenta, hem de mirar que no treballi tots els dies de la setmana. Per realitzar aquesta fórmula, utilitzem totes les operacions bàsiques AND, OR i NOT.

Per exemple, la dependenta 1 no pot treballar els 7 dies utilitzaríem:

$$NOT (\{treballaDia1\}AND \{treballaDia2\} AND ... \{treballaDia7\})$$

on treballaDia1 seria:

$$OR ([T1_D1, T2_D1])$$

- **Restricció 5:**

“Les dependents no poden superar les hores totals setmanals.”

Aquesta restricció és la més complexa, utilitza la variable ‘variablesT’ i la funció que ens proporciona la llibreria per treballar numèricament ‘weightedSum’. L’algorisme realitza una suma ponderada de les variables de tots els torns per una dependenta (T1_D1, T2_D1...) multiplicat per les hores d’aquell torn en concret. Així doncs, si una de les variables és certa sumarà les hores d’aquell torn, sinó no. Finalment, fa la comparació entre el resultat de la suma ponderada i les hores totals de la dependenta.

Per a que s’entengui millor, la fórmula final genèrica quedaria d’aquesta manera i s’hauria de crear per cada dependenta:

$$T1_D1*horesT1 + T2_D1*horesT2 + \leq horesD1$$

❖ **Obtenir solució:**

Per acabar tot aquest procés de programació de restriccions, finalment cridem a la funció *solve()*, per tal que ens solucioni el problema que hem definit en cas que sigui possible. A partir d’aquí podríem mostrar per pantalla les variables que són certes, mostrar diferents solucions, etc. En el nostre cas, assignem les variables booleanes que han acabat sent certes, per exemple si ‘T2_D3’ es certa, assignem la dependenta 3 al torn 2.

3.3.4. Taula resum funcions heurístiques

Finalment, mostrem una taula resum de les funcions heurístiques implementades, on veurem com cadascuna va evolucionant respecte les anteriors i quines són les diferències principals.

<i>Heurística</i>	Descripció
1	Realitza l'assignació d'un únic dia, realitzant diferents exploracions aleatòries i agafant la que els treballadors sumen millor nota.
2	Realitza l'assignació setmanal de manera seqüencial, mirant que compleixi la restricció del període i que no s'assigni dos cops al mateix dia.
3	Segueix realitzant l'assignació de forma seqüencial, sense tirar enrere en cap moment, però mirant que una assignació compleixi totes les restriccions.
4	Primer backtracking bàsic, simplement explora totes les solucions en l'ordre inicial del problema tenint en compte totes les restriccions.
5	Backtracking guiat 1. Ordena dinàmicament les dependents segons les hores acumulades en un 80% i segons la seva puntuació en un 20%.
6	Backtracking guiat 2. Aquest cop ordena les dependents segons les horesDeute en un 60%, les hores acumulades en un 30% i la nota en un 10%.
<i>logicSolver</i>	Constraint programming mitjançant un SAT Solver.

Taula 18. Taula resum de les funcions heurístiques

3.4. Proves

3.4.1. Definició proves

Per finalitzar l'apartat de desenvolupament, durem a terme unes petites proves per comprovar el funcionament del codi implementat. És important mencionar que aquestes proves es realitzen en l'entorn tecnològic de AWS Cloud9, és a dir, el codi s'executa en una instància EC2 dins dels servidors d'Amazon.

Sobretot, com que estem tractant amb una classe de problema NP-hard, l'objectiu d'aquestes proves és analitzar l'eficiència computacional. Per fer això, utilitzarem les següents mesures:

- Temps d'execució en milisegons
- Nombre de nodes explorats

A partir d'aquestes mesures compararem quines són les heurístiques més eficients, concretament utilitzarem les següents:

- **Heurística 4**
És el backtracking bàsic i volem comparar la seva eficiència respecte a un backtracking guiat.
- **Heurística 5**
És el primer backtracking guiat que hem implementat. No utilitzarem notes, ja que cada cas podria ser molt diferent, però volem comprovar el comportament del algorisme utilitzant la ordenació dinàmica de les hores acumulades.
També havíem pensat a utilitzar l'heurística 6 amb una funció d'ordenació diferent, però finalment no el provarem, ja que es un algorisme que se centra en que l'usuari no s'hagi de preocupar tant per les hores i afegeix unes hores de deute, per tant, no es centra en l'eficiència que es el que estem analitzant ara.
- **LogicSolver**
Per últim, mirarem com actua en termes d'eficiència un SAT Solver en comparació amb les altres heurístiques. Aquest cas no fem l'exploració d'un arbre així que no podem calcular els nodes.

Per tal d'obtenir una bona mostra de resultats, haurem de provar diferents entrades per cada heurística, seran les següents:

➤ **Entrada fàcil:**

- 4 dependents que entre totes poden realitzar 120 hores.
- 13 torns que entre tots sumen 104 hores.

➤ **Entrada mitjana:**

- 7 dependents que entre totes poden realitzar 155 hores.
- 23 torns que entre tots sumen 149 hores.

➤ **Entrada difícil:**

- 10 dependents que entre totes poden realitzar 230 hores.
- 37 torns que entre tots sumen 219 hores.

➤ **Entrada molt difícil:**

- 12 dependents que entre totes poden realitzar 270 hores.
- 47 torns que entre tots sumen 264 hores.

➤ **Entrada extrema:**

- 13 dependents que entre totes poden realitzar 375 hores.
- 63 torns que entre tots sumen 369 hores.

En [l'apartat 8](#) d'annexos de la memòria, mostrarem l'exemple de l'entrada extrema, així com la seva solució.

3.4.2. Hipòtesi

En aquest cas, estem duent a terme unes proves senzilles on únicament tenim dues variables respecte a l'eficiència, així que no haurem de realitzar cap hipòtesi de relació entre les variables. Pensem que la relació entre els nodes explorats i el temps d'execució hauria de ser pràcticament lineal, ja que les dues variables mesuren l'eficiència.

Respecte als diferents casos d'entrada, la nostra opinió és que està clar quina serà la diferència, ja que el cas fàcil serà el més assequible pel programa i el resoldrà ràpidament. Llavors, a mesura que augmenti la dificultat de l'entrada, com hem comentat en repetides ocasions, la complexitat computacional augmentarà exponencialment i és molt possible que les pitjors heurístiques no puguin arribar a solucionar l'entrada difícil.

Per últim, respecte a les heurístiques, farem la hipòtesi a través d'una llista ordenada on primer esmentarem les heurístiques que pensem que aniran millor i la justificació, tot i que després a resultats podria aparèixer algú totalment diferent.

➤ **logicSolver**

En primer lloc, penso que la programació de restriccions junt amb el SAT Solver serà la tècnica més eficient per resoldre aquest tipus de problemes. Per la seva pròpia naturalesa, el problema dels horaris comporta un seguit de restriccions i el constraint programming es va crear just per aquests problemes on hi ha un gran nombre de possibilitats, com per exemple un Sudoku. A més a més, el funcionament intern dels SAT Solvers està dissenyat per obtenir una gran eficiència en aquest cas on es volen explorar totes les solucions i és el mètode més efectiu respecte els problemes que treballen amb formules lògiques. En definitiva, tot i que en aquest cas no podrem obtenir en nombre de nodes explorats, pensem que serà la funció amb menor temps d'execució.

➤ **Heurística 5**

Pensem que la segona millor heurística en termes de rendiment serà la heurística 5. Aquesta utilitza el primer backtracking guiat, el qual es centra molt en les hores acumulades de cada dependenta. El problema té greus problemes quan arriba a un punt avançat de l'exploració i es troba que les dependents ja tenen les hores molt ocupades. Així doncs, gràcies a l'ordenació dinàmica d'aquestes hores acumulades, l'algorisme podrà mirar primer les dependents que tinguin més hores disponibles, fet que pot millorar en gran mesura el rendiment.

➤ **Heurística 4**

Finalment, l'heurística que no utilitza ninguna ordenació dinàmica hauria de ser la que tingui més problemes computacionals i no ens sorprendria que el programa no s'acabés d'executar mai.

3.4.3. Resultats

A continuació, mostrem els resultats obtinguts en aquesta prova:

<i>Heurística</i>	<i>Entrada</i>	<i>Nodes</i>	<i>Temps (ms)</i>
4	Fàcil	263	7,02
	Mitjana	6.859	24,83
	Difícil	5.557.615	589,69
	Molt difícil	No acaba	No acaba
	Extrem	No acaba	No acaba
5	Fàcil	21	5,29
	Mitjana	2.882	24,22
	Difícil	582.737	195,15
	Molt difícil	No acaba	No acaba
	Extrem	No acaba	No acaba
logicSolver	Fàcil	-	192,69
	Mitjana	-	437,33
	Difícil	-	654,59
	Molt difícil	-	2954,77
	Extrem	-	3141,62

Taula 19. Resultats de les proves

3.4.4. Conclusió

Un cop obtinguts tots els resultats de les proves, podem observar com el logicSolver és clar guanyador en termes d'eficiència. Així doncs, es compleix la hipòtesi que havíem plantejat, la qual semblava bastant òbvia, tot i que no esperàvem que hi hagués tanta diferència entre el constraint programming i les heurístiques implementades.

Podem veure com en els casos més senzills les heurístiques funcionen més ràpid que el logicSolver. Això és degut a la definició de les variables i de les restriccions dins del procés de la programació de restriccions. Per altra banda, en els casos més complexos, s'observa com les heurístiques tenen molts problemes i no poden trobar la solució en un temps raonable, mentre que el logicSolver, no tan sols troba la solució, sinó que la troba molt ràpidament.

També podem observar la diferència entre el backtracking bàsic i el backtracking guiat. Si, per exemple, mirem el nombre de nodes i el temps d'execució durant l'entrada difícil entre les heurístiques 4 i 5, ràpidament arribem a la conclusió que si es proporciona una ajuda al programa per tal que explori els nodes més prometedors, aquest trobarà la solució amb molts menys nodes. No es pot veure com creix, ja que a partir dels casos complexos l'algorisme ja no acaba, però veuríem com aquesta diferència de nodes entre les dues funcions creix exponencialment.

En conclusió, les heurístiques podrien anar bé per treballar casos d'assignació horària amb poques dades, en els quals troba la solució en molts pocs nodes. També caldria analitzar la qualitat de la solució, ja que nosaltres només estem mirant que es compleixin certes restriccions, així doncs penso les heurístiques tindrien l'avantatge en els casos que no només es volgués tenir en compte les restriccions, sinó que també volguéssim mirar que tan bona és una solució. Tot i això, queda clar que en situacions reals i en casos més complexos, la programació de restriccions és la forma més viable i eficient per a generar els horaris.

4. Conclusions

4.1. Objectius del projecte i resultats obtinguts

Per concloure aquest projecte, finalment extraurem algunes conclusions respecte als resultats obtinguts i tenint en compte els objectius marcats en un principi.

En primer lloc, penso que hem assolit l'objectiu principal que es tractava de crear un sistema automàtic, personalitzat i continuat que ens permetés generar els horaris d'una empresa. No només això, sinó que també hem pogut assolir l'objectiu secundari, analitzant diferents algorismes per tractar el problema i, finalment, hem trobat la solució tecnològica òptima, basada en la programació de restriccions.

Adicionalment, tot i que no era part de cap objectiu inicial, hem aconseguit crear un sistema de puntuacions que, com ja hem comentat, en cas que s'acabés implantant aportaria grans avantatges a l'empresa usuària.

En la següent taula, mostrem els requisits funcionals i no funcionals que havíem plantejat en l'inici del treball i els puntuarem amb una nota del 1 al 5 calculant el grau d'assoliment (1 - no assolit, 5 – assolit amb excel·lència):

	Requisit	Nota
Requisits funcionals	Entrada dels treballadors	5
	Entrada dels torns de treball	4
	Sortida final del horari	4
	Gestió de la informació	5
	Restriccions	4
	Visualització de l'horari	5
Requisits no funcionals	Eficiència	5
	Seguretat	5
	Utilitat	3
	Usabilitat	5
	Disponibilitat	5
	Escalabilitat	5

Taula 20. Grau d'assoliment dels requisits

Així doncs, penso que hem assolit més que correctament tots els requisits i objectius que s'havien plantejat inicialment. Hem aconseguit crear un software amb diferents funcionalitats i que, sobretot, ofereix grans característiques, com l'eficiència, la usabilitat o l'escalabilitat, entre altres. Malgrat tot això, i aquest és el motiu pel qual hi ha notes que no arriben al 5, també som conscients que existeixen certes limitacions en la solució proposada, com veurem en el següent apartat.

4.2. Avaluació de les limitacions i possibles millores

El sistema generador d'horaris proposat ofereix grans funcionalitats i característiques, però també és cert que té certes limitacions i carències quan pensem en el seu ús en casos d'empreses reals.

El problema principal que ens hem trobat va relacionat amb l'entrada dels torns de treball, ja que l'usuari ha de complir la tasca de donar la informació de cada torn per separat i de cada dia, una tasca manual que s'ha de realitzar cada setmana que es vol generar un horari i, tenint en compte que volíem aplicar la màxima automatització, aquest factor podria generar certa insatisfacció a l'usuari. Si ve és cert que aquesta tasca manual és d'alguna manera necessària, ja que necessitem obtenir informació de quines són les hores i els dies amb més demanda, caldria investigar si existeix una manera més còmoda per fer-la. Ja estem parlant d'un problema molt complex NP-Hard amb infinitat de possibilitats i sense aquesta entrada de dades manual ens hauria sigut impossible de resoldre.

Aquest problema afecta molt a la utilitat del nostre software, un requisit no funcional que havíem definit, ja que haver de crear els torns pot resultar poc útil per l'usuari. Tot i això, penso que assolim parcialment aquesta utilitat i li estalviem molta feina a aquest usuari, perquè és la màquina qui s'encarrega d'analitzar l'ampli ventall de possibilitats que pot tenir un conjunt de torns, una tasca que seria impossible de dur a terme per una persona.

En definitiva, veiem aquest projecte situat en una fase inicial en la qual encara queda certa etapa de desenvolupament abans d'arribar a la implantació en el mercat real. La bona notícia és que pensem que hem creat un sistema que de base és molt escalable, el qual es pot anar millorant i, a partir de noves funcionalitats i característiques, podria arribar a implantar-se i tenir una gran utilitat.

A continuació, mostrem quines possibles millores es podrien implementar de cara al futur:

❖ **Nou disseny d'entrada de torns.**

Respecte al principal problema que acabem de comentar, es podria dissenyar una nova entrada de dades on, en comptes de ficar tots els torns, l'usuari només entrés un valor del nombre de dependents que vol a cada hora. Però pensem que utilitzar una entrada més

senzilla també produirà pèrdua d'informació i caldria veure en nous sistemes com afecta la qualitat de la solució.

❖ **Intel·ligència artificial.**

En el cas més optimista, podríem aplicar una tecnologia que hem mencionat en aquesta memòria, però no hem sigut capaços d'utilitzar, com és la intel·ligència artificial. Així doncs, en aquest cas òptim, seria la pròpia IA la que, analitzant les dades de vendes, calcularia les hores puntes de cada dia i generaria automàticament també els torns. Tot i això, no se fins a quin punt és realista, ja que també s'hauria de tenir en compte dies especials, noves dependents, la gran variabilitat de feina en etapes diferents, etc.

❖ **Més solucions horàries.**

En tractar-se d'un problema molt complex, nosaltres hem decidit simplement generar la primera solució que trobi, però també és cert que podríem aplicar una millora on es generessin més horaris i que l'usuari pogués triar el seu preferit.

❖ **Puntuació en la solució.**

Relacionat amb el punt anterior, es podria dissenyar un sistema de puntuacions per tal d'avaluar un horari final, o també es podria fer en solucions parcials de cada node. D'aquesta manera obtindríem una nova heurística que, tot i perjudicar l'eficiència, podria obtenir solucions més òptimes dins del conjunt d'estats.

❖ **Noves dades de les dependents.**

Una altra millora que s'ha contemplat durant aquest projecte és l'entrada de més informació detallada respecte a les dependents. A partir de l'entrada de dades per part de l'usuari o de l'anàlisi de dades de la BD, podríem aconseguir nous atributs de les dependents per tal de generar un horari, com per exemple quines persones són líders, quines persones són aprenents, etc. Gràcies a aquest nou sistema podríem distingir les dependents en diferents categories i dissenyar una solució més bona.

Una altra idea, tot i que més delicada, seria afegir les relacions interpersonals entre els treballadors d'una botiga, per tal d'ajuntar les persones que es porten millor entre elles o el contrari. Totes aquestes noves dades implicarien l'ús de noves restriccions, així que també complicarien el problema.

❖ **Afegir o modificar restriccions.**

Per últim, es podrien afegir petites millores respecte al tema de les restriccions. Per exemple, afegir que un torn pogués tenir més d'una dependenta o afegir el doble torn de matí i tarda, permetent que una dependenta pugui aparèixer més d'una vegada al dia.

A més, respecte a la millora d'afegir noves dades, si afegíssim una categoria per cada dependenta, hauríem de crear noves restriccions com "Sempre hi ha d'haver una líder a la botiga" o "No pot haver-hi més de dues aprenents".

Finalment, un cop aplicades les millores que consideréssim, també caldria realitzar proves més completes i exhaustives que les que hem realitzat nosaltres, per tal de veure la viabilitat del projecte en una implantació real. Les úniques proves que hem dut a terme anaven destinades a comprovar l'eficiència de la solució proposada, però també es podrien analitzar moltes altres mesures, com per exemple la desviació que produeix cada algorisme respecte a les hores assignades a cada dependenta o la qualitat de la solució.

4.3. Conclusió personal i agraïments

Com a acabament, faré una breu conclusió personal. Començant per la part negativa, aquest projecte, com a qualsevol altre similar, ha provocat certs obstacles i moments de dificultat durant tot el procés. Com ja havíem identificat al principi, la qual cosa era bastant inevitable, la inexperiència i la gestió del temps ens han dificultat molt poder avançar en certs moments, causant alguns maldecaps en el bon sentit de la paraula.

No obstant això, un cop acabat el projecte, la conclusió final que obtinc és molt positiva. Posar en pràctica aquest treball m'ha aportat molta experiència, sobretot en aspectes tècnics com el llenguatge de programació JavaScript, en NodeJs, en l'ús de bases de dades relacionals com MS SQL Server, entre altres. També he obtingut coneixements molt útils respecte a l'àmbit dels problemes de generació horària, així com d'altres problemes relacionats amb les restriccions, i he pogut treballar amb AWS Cloud9 dins de l'entorn d'una empresa real.

Finalment, amb el sistema dissenyat i aplicant certes millores molt concretes, em permetrà ajudar a la meva mare, la qual s'encarrega de la realització d'horaris a la seva empresa i podrà ser usuària d'aquest i generar l'assignació de manera més còmoda.

A l'últim, agrair al Joaquim Deulofeu, director del TFG, als responsables de GEP i als lectors/avaluadors d'aquesta memòria, espero no haver avorrit amb tant text . Sobretot agrair a l'empresa Hit Systems, al seu encarregat, en Jordi Bosch, i als seus treballadors, que m'han aportat les eines i solucions necessàries, a part de guiar-me durant tot el procés. També agrair a la meva mare, Rosa Rovira i al Toni Vila, actors usuaris del sistema que m'han motivat a la realització d'aquest projecte, i als familiars i amics que m'han ajudat indirectament a la finalització d'aquest treball. Moltes gràcies.

5. Referències

- [1] AI Powered Auto Scheduling Software | Rotageek. (s. f.). <https://www.rotageek.com/rota-auto-scheduling>
- [2] AWS Pricing Calculator. (s. f.). <https://calculator.aws/#/>
- [3] Babu, R. (2023, 28 marzo). Array vs Set vs Map vs Object — Real-time use cases in Javascript (ES6/ES7). Medium. <https://codeburst.io/array-vs-set-vs-map-vs-object-real-time-use-cases-in-javascript-es6-47ee3295329b>
- [4] Béjar, J. (2022a). Búsqueda Heurística. https://www.cs.upc.edu/~bejar/ia/transpas/teoria/2-BH2-Busqueda_heuristica.pdf
- [5] Béjar, J. (2022b). Búsqueda local. https://www.cs.upc.edu/~bejar/ia/transpas/teoria/2-BH3-Busqueda_local.pdf
- [6] BOE-A-2018-16673 Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales. (s. f.). <https://www.boe.es/buscar/act.php?id=BOE-A-2018-16673>
- [7] Brucker, P., Burke, E., Curtois, T., Qu, R., & Van Den Berghe, G. (2010). A shift sequence based approach for nurse scheduling and a new benchmark dataset. *Journal of Heuristics*, 16(4), 559-573. <https://doi.org/10.1007/s10732-008-9099-6>
- [8] Burke, E., De Causmaecker, P., Van Den Berghe, G., & Van Landeghem, H. (2004). The State of the Art of Nurse Rostering. *Journal of Scheduling*, 7(6), 441-499. <https://doi.org/10.1023/b:josh.0000046076.75950.0b>
- [9] Canva. (s. f.). Free Design Tool: Presentations, Video, Social Media | Canva. <https://www.canva.com/>
- [10] Castellanos, E. (2021). Git vs GitHub – ¿Qué es el Control de Versiones y Cómo Funciona? freeCodeCamp.org. <https://www.freecodecamp.org/espanol/news/git-vs-github-what-is-version-control-and-how-does-it-work/>
- [11] colaboradores de Wikipedia. (2023). Microsoft SQL Server. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Microsoft_SQL_Server

- [12] Com escrib el meu TFG/TFM | bibliotècnica. (2021, 23 novembre).
<https://bibliotecnica.upc.edu/estudiants/6-passos-que-teu-tfg/tfm-sigui-exit/escric-meu-tfg/tfm#consells-redaccio>
- [13] Competències. (s. f.). Facultat d'Informàtica de Barcelona.
<https://www.fib.upc.edu/ca/estudis/graus/grau-en-enginyeria-informatica/pla-destudis/competencies>
- [14] ¿Cuánto Cobra un Programador? (Sueldo 2023) | Jobted.es. (s. f.).
<https://www.jobted.es/salario/programador>
- [15] DISTRIBUCIÓN DE LA JORNADA LABORAL PARA 2022. (2022). <https://asime.es/wp-content/uploads/2021/12/DISTRIBUCION-JORNADA-2022.pdf>
- [16] Eligenio. (2023). ¿Cuánto consume un ordenador de sobremesa? Eligenio.
<https://eligenio.com/es/blog/cuanto-consume-un-ordenador-de-sobremesa/>
- [17] Fernández, Y. (2020). Licencias de Windows 10: tipos y dónde comprar. Xataka.
<https://www.xataka.com/basics/licencia-windows-10-tipos-donde-comprar>
- [18] Gorbachenko, P. (2021, 18 noviembre). Functional vs Non-Functional Requirements [Updated 2021]. Enkonix. <https://enkonix.com/blog/functional-requirements-vs-non-functional/>
- [19] Hakim, L., Bakhtiar, T., & Jaharuddin. (2017). The nurse scheduling problem: a goal programming and nonlinear optimization approaches. IOP Conference Series: Materials Science and Engineering, 166, 012024. <https://doi.org/10.1088/1757-899x/166/1/012024>
- [20] IDE en la nube - AWS Cloud9 - AWS. (s. f.). Amazon Web Services, Inc.
<https://aws.amazon.com/es/cloud9/>
- [21] idealista.com. (s. f.). Evolución del precio de la vivienda en alquiler en Barcelona provincia. idealista. <https://www.idealista.com/sala-de-prensa/informes-precio-vivienda/alquiler/cataluna/barcelona-provincia/>
- [22] Introduction to Node.js. (s. f.). Introduction to Node.js. <https://nodejs.dev/en/learn/>
- [23] JavaScript Tutorial. (s. f.). <https://www.w3schools.com/js/default.asp>
- [24] Lucid visual collaboration suite: Log in. (s. f.). <https://lucid.app/>
- [25] Lutkevich, B. (2023). expert system. Enterprise AI.
<https://www.techtarget.com/searchenterpriseai/definition/expert-system>
- [26] Maenhout, B., & Vanhoucke, M. (2007). An electromagnetic meta-heuristic for the nurse scheduling problem. Journal of Heuristics, 13(4), 359-385. <https://doi.org/10.1007/s10732-007-9013-7>

- [27] Microsoft. (2022). Comparar todos los planes de Microsoft 365 (anteriormente Office 365): Microsoft Store. <https://www.microsoft.com/es-es/microsoft-365/buy/compare-all-microsoft-365-products>
- [28] Nihar-Raval. (2022). Python vs JavaScript: Battle of the Two Greatest Programming Languages. Radixweb. <https://radixweb.com/blog/python-vs-javascript>
- [29] npm: logic-solver. (s. f.). npm. <https://www.npmjs.com/package/logic-solver>
- [30] npm: mssql. (s. f.). npm. <https://www.npmjs.com/package/mssql>
- [31] Office Timeline Online - Interactive Timeline & Gantt Chart Maker. (s. f.). Office Timeline Online. <https://online.officetimeline.com/>
- [32] Performance of JavaScript .forEach, .map and .reduce vs for and for..of. (s. f.). https://leanylabs.com/blog/js-forEach-map-reduce-vs-for-for_of/
- [33] Precio de la tarifa de luz por horas HOY | Consulta ahora. (2023, 5 noviembre). <https://tarifaluzhora.es/>
- [34] Preu targeta T-jove metro bus Barcelona | Transports Metropolitans de Barcelona. (s. f.). <https://www.tmb.cat/ca/tarifes-metro-bus-barcelona/senzills-i-integrats/t-jove>
- [35] ¿Qué es AWS? (s. f.). Amazon Web Services, Inc. <https://aws.amazon.com/es/what-is-aws/>
- [36] ¿Qué es JavaScript? - Aprende desarrollo web | MDN. (2023, 8 mayo). https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/What_is_JavaScript
- [37] Qué es un diagrama de flujo. (s. f.). Lucidchart. <https://www.lucidchart.com/pages/es/que-es-un-diagrama-de-flujo>
- [38] Real Decreto Legislativo 1/1996, de 12 de abril. (s. f.). <https://www.boe.es/buscar/act.php?id=BOE-A-1996-8930>
- [39] Rodríguez, E. (2023). Introduction to Constraint Programming. <https://www.cs.upc.edu/~erodri/webpage/cps/theory/cp/intro/slides.pdf>
- [40] Simplilearn. (2023). JavaScript vs Python: Understand the Key Differences. Simplilearn.com. <https://www.simplilearn.com/tutorials/programming-tutorial/javascript-vs-python>
- [41] Skello Pricing, Alternatives & More. (s. f.). <https://www.capterra.com/p/179936/Skello/#prosCons>
- [42] Softcatalà – Informàtica i programari en català. (s. f.). <https://www.softcatala.org/>
- [43] Software de gestion de personal - Skello. (s. f.). <https://www.skello.es/>

- [44] Solos, I. P., Tassopoulos, I. X., & Beligiannis, G. N. (2013). A Generic Two-Phase Stochastic Variable Neighborhood Approach for Effectively Solving the Nurse Rostering Problem. *Algorithms*, 6(2), 278-308. <https://doi.org/10.3390/a6020278>
- [45] SQL Server 2022—Pricing | Microsoft. (s. f.). <https://www.microsoft.com/en-us/sql-server/sql-server-2022-pricing>
- [46] Stsepanets, A., & Stsepanets, A. (2023). Modelo cascada, qué es y cuándo conviene usarlo. Gantt Chart GanttPRO Blog. <https://blog.ganttpro.com/es/metodologia-de-cascada/>
- [47] The end of the scheduling headache with Smart Planner, Skello's flagship innovation. (s. f.). <https://www.foodhoteltech.com/en/smart-planner-skello/>
- [48] Valouxis, C., Gogos, C., Goulas, G., Alefragis, P., & Housos, E. (2012). A systematic two phase approach for the nurse rostering problem. *European Journal of Operational Research*, 219(2), 425-433. <https://doi.org/10.1016/j.ejor.2011.12.042>
- [49] Villegas, E. (2023). Descubre qué es y cómo hacer el estado del arte de tu proyecto. Tesis y Másters Colombia. <https://tesisymasters.com.co/estado-del-arte-en-investigacion/>
- [50] Wikipedia contributors. (2023a). Nurse scheduling problem. Wikipedia. https://en.wikipedia.org/wiki/Nurse_scheduling_problem
- [51] Wikipedia contributors. (2023b). Metaheuristic. Wikipedia. <https://en.wikipedia.org/wiki/Metaheuristic>
- [52] Wikipedia contributors. (2023c). Optimal job scheduling. Wikipedia. https://en.wikipedia.org/wiki/Optimal_job_scheduling
- [53] Wikipedia contributors. (2023d). Artificial intelligence. Wikipedia. https://en.wikipedia.org/wiki/Artificial_intelligence
- [54] Wikipedia contributors. (2023e). NP-hardness. Wikipedia. <https://en.wikipedia.org/wiki/NP-hardness>
- [55] Wikipedia contributors. (2023f). Mathematical optimization. Wikipedia. https://en.wikipedia.org/wiki/Mathematical_optimization
- [56] Wikipedia contributors. (2023g). Heuristic. Wikipedia. <https://en.wikipedia.org/wiki/Heuristic>
- [57] Wikiwand - Árbol k-ario. (s. f.). Wikiwand. https://www.wikiwand.com/es/%C3%81rbol_k-ario

6. Índex de figures

Figure 1. Diagrama d'Euler de la complexitat dels problemes en el cas $P \neq NP$	8
Figure 2. Gràfic d'una funció objectiu amb un màxim global en el punt (0,0,4).....	18
Figure 3. Gràfic on podem veure els diferents estats segons una funció objectiu.....	19
Figure 4. Exploració d'un arbre amb un DFS fins a trobar una possible solució (nodes grisos).....	20
Figure 5. Gràfic per visualitzar com actuen en l'espai de solucions els algorismes de 'Hill Climbing' i 'Simulated Annealing'	21
Figure 6. Model simple d'una xarxa neuronal utilitzada en l'aprenentatge automàtic de reconeixement d'animals.....	23
Figure 7. Interfície gràfica d'un horari en el software Skello.....	25
Figure 8. Característiques que té en compte Rotageek per realitzar horaris automàtics.	26
Figure 9. Diagrama de les fases de la metodologia cascada.....	30
Figure 10. Conjunt de serveis més importants que ofereix l'AWS	31
Figure 11. Captura de pantalla del entorn de treball on s'ha desenvolupat aquest projecte.....	32
Figure 12. Dependències entre les tasques.....	39
Figure 13. Abast del TFG.....	47
Figure 14. Mapa conceptual del problema	57
Figure 15. Diagrama de classes.....	58
Figure 16. Esquema simplificat del problema.....	62
Figure 17. Diagrama de seqüència.....	63
Figure 18. UI. Portal de cerca treballadors.....	64
Figure 19. UI. Dades d'un treballador	64
Figure 20. UI. Desplegables calendari laboral	65
Figure 21. UI. Funcionalitat per marcar un dia de la setmana com a festiu.....	65
Figure 22. UI. Exemple del mes de març d'un calendari laboral	65
Figure 23. UI. Portal per modificar i visualitzar horaris.....	66
Figure 24. UI. Exemple dels torns planificats d'un dia concret.....	67
Figure 25. UI. Exemple dels torns una vegada han estat assignats.....	67
Figure 26. Diagrama de flux de la funció valoracioMensual().	84
Figure 27. Petició SQL per a crear la taula de puntuacions.	85
Figure 28. Mapa conceptual de la funció main().....	92
Figure 29. Arbre d'exploracions del backtracking	97
Figure 30. Diagrama de flux de la programació de restriccions.....	99
Figure 31. Restricció 2 implementada en JavaScript i utilitzant 'logic-solver'.....	102

7. Índex de taules

Taula 1. Distribució del temps per tasca	40
Taula 2. Resum dels riscos amb la seva probabilitat d'ocurrència i impacte.	41
Taula 3. Pressupost hardware.....	44
Taula 4. Pressupost software.....	44
Taula 5. Pressupost despeses generals.....	45
Taula 6. Pressupost total	46
Taula 7. Matriu de sostenibilitat	47
Taula 8. Taula dependents	68
Taula 9. Taula Dependentes Estès	69
Taula 10. Taula Torns	69
Taula 11. Taula Planificació	70
Taula 12. Taula Calendari Laboral.....	70
Taula 13. Taula entrades i sortides	72
Taula 14. Taula vendes	72
Taula 15. Taula puntuacions mensuals.....	74
Taula 16. Classe Dependenta.....	90
Taula 17. Classe Turno	90
Taula 18. Taula resum de les funcions heurístiques	104
Taula 19. Resultats de les proves.....	108
Taula 20. Grau d'assoliment dels requisits.....	110

8. Annexos

Annex 1.

Com a annex 1, mostrarem l'entrada de dades i la solució de la prova en el cas extrem. Consistirà en les matrius que contindran la informació de cada dependent i cada torn:

Dependents:

M = Matí
T = Tarda
A = Ambdues

Codi	HoresSet	TipusHorari
1	40	['M', 'M', 'M', 'M', 'M', 'A', 'T']
2	30	['T', 'T', 'T', 'T', 'T', 'T', 'A']
3	35	['A', 'A', 'A', 'A', 'A', 'A', 'A']
4	25	['M', 'M', 'M', 'M', 'M', 'A', 'A']
5	35	['M', 'T', 'M', 'M', 'M', 'M', 'M']
6	40	['M', 'M', 'T', 'M', 'M', 'A', 'A']
7	25	['A', 'T', 'A', 'T', 'M', 'M', 'M']
8	20	['T', 'T', 'T', 'T', 'T', 'A', 'A']
9	25	['T', 'T', 'T', 'T', 'T', 'T', 'T']
10	20	['M', 'M', 'M', 'T', 'T', 'M', 'M']
11	35	['M', 'M', 'T', 'M', 'M', 'T', 'T']
12	30	['T', 'T', 'T', 'M', 'T', 'M', 'M']
13	15	['M', 'T', 'M', 'M', 'T', 'T', 'M']

Torns:

ID	DiaSetmana	Període	Hi	Hf	TotalHores
01	0	M	06	14	8
02	0	M	08	14	6
03	0	M	10	15	5
04	0	M	09	16	7
05	0	T	15	20	5
06	0	T	14	21	7
07	0	T	15	20	5
08	0	T	12	20	8
11	1	M	06	13	7
12	1	M	08	14	6
13	1	M	08	15	7
14	1	M	09	15	6
15	1	T	15	20	5
16	1	T	14	21	7
17	1	T	15	20	5
21	2	M	06	14	8
22	2	M	08	14	6
23	2	M	08	14	6

24	2	M	09	15	6
25	2	M	09	14	5
26	2	M	09	16	7
27	2	T	13	18	5
28	2	T	15	20	5
29	2	T	14	21	7
210	2	T	15	20	5
31	3	M	08	14	6
32	3	M	08	14	6
33	3	M	06	13	7
34	3	M	08	13	5
35	3	T	14	21	7
36	3	T	16	20	4
37	3	T	16	20	4
41	4	M	08	14	6
42	4	M	08	14	6
43	4	M	08	13	5
44	4	M	08	13	5
45	4	T	14	21	7
46	4	T	16	20	4
47	4	T	16	20	4
51	5	M	06	14	8
52	5	M	08	14	6
53	5	M	08	14	6
54	5	M	09	15	6
55	5	M	09	14	5
56	5	M	09	16	7
57	5	T	13	18	5
58	5	T	15	20	5
59	5	T	14	21	7
510	5	T	15	20	5
511	5	T	15	18	3
61	6	M	06	13	7
62	6	M	08	14	6
63	6	M	08	15	7
64	6	M	09	15	6
65	6	M	08	13	5
66	6	M	10	16	6
67	6	M	10	15	5
68	6	M	09	13	4
69	6	M	08	15	7
610	6	T	12	17	5
611	6	T	13	19	6
612	6	T	13	21	8
613	6	T	14	18	4

Solució:

ID TORN	CODI DEPENDENTA
01	6
02	7
03	13
04	5

05	12
06	8
07	2
08	3
11	1
12	4
13	11
14	10
15	12
16	2
17	3
21	10
22	3
23	1
24	7
25	5
26	4
27	9
28	2
29	11
210	6
31	5
32	6
33	1
34	11
35	3
36	9
37	2
41	5
42	1
43	6
44	11
45	12
46	9
47	8
51	7
52	6
53	4
54	1
55	12
56	5
57	13
58	8
59	9
510	11
511	3
61	12
62	3
63	6
64	4
65	13
66	10
67	7
68	5
69	2
610	9
611	11
612	1

613	8
-----	---

Annex 2.

Com a annex 2, veurem el diagrama de Gantt en la última pàgina horitzontal.

Diagrama de Gantt

