Aplicatiu de Creació Automàtica de Planificacions d'Entrenament Personalitzat

Roger Almató Baucells

Treball De Fi de Grau
Facultat d'Informàtica de Barcelona
Grau en Enginyeria Informàtica

Supervisat per:
Tutor: José María Rodríguez Valls (Trainerer)
Ponent: Enrique Romero Merino (UPC)

Projecte realitzat en cooperació amb l'empresa Trainerer

Resum

El projecte Aplicatiu de Creació Automàtica de Planificacions d'Entrenament és el Treball de Fi de Grau d'Enginyeria Informàtica de la Universitat Politècnica de Catalunya (BarcelonaTech) realitzat per l'alumne Roger Almató Baucells. Està realitzat en cooperació amb l'empresa Trainerer i supervisat pel CTO de Trainerer, José María Rodríguez Valls. El ponent del projecte és Enrique Romero Merino, professor de la Facultat d'Informàtica de Barcelona (FIB) i de la Facultat de Matemàtiques i Estadística (FME).

Quan una persona es vol iniciar en el món de l'esport, o fins i tot els atletes més avançats, volen realitzar o realitzen exercici físic amb un objectiu. Aquest podria ser posar-se en forma, perdre pes o preparar-se físicament per una marató. És en aquest moment, quan s'ha escollit l'objectiu esportiu, que apareix el següent problema:

- Com ho faig?
- Què és el que necessito?
- Quan i quines sessions d'entrenament haig de fer?

El projecte pretén aportar a aquest col·lectiu una resposta en forma de planificacions de temporades completes i planificacions setmanals de sessions d'entrenament. Les temporades s'ajusten al nombre de setmanes que disposa la persona per preparar-se i al seu objectiu esportiu. Per altra banda, les planificacions setmanals, s'ajusten a la disponibilitat d'horaris de la persona, al moment de la temporada en el qual es troba i al seu estat físic actual.

A al llarg de la memòria es descriuen les dificultats que sorgeixen al resoldre el problema de generar plans d'entrenament personalitzat al llarg d'una temporada. Primer, els derivats de la quantitat de factors que hi intervenen, com la disponibilitat de l'atleta, l'objectiu esportiu o les preferències per els diferents tipus d'entrenament. El treball, també explica el que representa a nivell algorítmic, i que actualment no s'ha desenvolupat un algorisme capaç de resoldre de manera eficient els problemes de distribució de tasques.

Pel que fa al mercat, el projecte pretén aportar un aplicatiu accessible per a tothom i també promoure la pràctica de l'esport de manera sana, progressiva i que eviti lesions. D'aquesta manera tant l'atleta que s'inicia com el més experimentat tenen un full de ruta a seguir per complir els seus objectius esportius amb èxit.

Resumen

El proyecto Aplicatiu de Creació Automàtica de Planificacions d'Entrenament es el Trabajo de Fin de Grado d'ingenieria Informática de el alumno Roger Almató Baucells a la Universitat Politècninca de Catalunya (BarcelonaTech). Está realizado en cooperación con la empresa Trainerer i supervisado por el CTO de Trainerer, José María Rodriguez Valls. El ponente del proyecto es Enrique Romero Merino, profesor de la Facultat d'Informàtica de Barcelona (FIB) y Facultat de Matemàtiques i Estadística (FME).

Cuando una persona se quiere iniciar en el mundo del deporte, o incluso atletas más avanzados, quieren realizar ejercicio físico con un objetivo. Este podría ser desde ponerse en forma, perder peso o prepararse físicamente por una maratón. Es en este punto, donde se ha escogido el objetivo deportivo, que aparece el siguiente problema:

- ¿Cómo lo hago?
- ¿Qué es lo que necesito?
- ¿Cuándo y que sesiones de entrenamiento debo realizar?

El proyecto pretende aportar a este colectivo una respuesta en forma de planificaciones de temporadas completas i planificaciones semanales de sesiones de entrenamiento. Las temporadas se ajustan al número de semanas que dispone la persona para preparar-se i a su objetivo deportivo. Así mismo, las planificaciones semanales, se ajustan a la disponibilidad de horarios de la persona, al momento de la temporada en la que se encuentre y a su estado físico.

A lo largo de esta memoria se describen las dificultades que surgen al resolver el problema de generar planes de entrenamiento personalizado a lo largo de una temporada. Primero, los derivados de la cantidad de factores que intervienen, como la disponibilidad del atleta, el objetivo deportivo o las preferencias por distintos tipos de entrenamiento. El trabajo, también explica lo que representa a nivel algorítmico, y que actualmente no se ha desarrollado un alegorismo capaz de resolver de manera eficiente los problemas de distribución de tareas.

A nivel de mercado, el proyecto pretende aportar un producto accesible para todo el mundo y promover la práctica del deporte de manera sana, evitando lesiones y progresiva. De este modo, el atleta que se inicia en el deporte o atletas mas experimentados, dispondrán de un camino a seguir para que sus objetivos deportivos sean un éxito.

Abstract

The project Aplicatiu de Creació Automàtica de Planificacions d'Entrenament is the Informatics Engineering final thesis of the student Roger Almató Baucells at the Universitat Politècnica de Catalunya (BarcelonaTech). This project is being developed together with Trainerer company and supervised for CTO at Trainerer, José María Rodriguex Valls. The professor supervisor is Enrique Romero Merion, professor at the Facultat d'Informàtica de Barcelona (FIB) and Facultat de Matemàtiques i Estadística (FME).

When a person wants to start exercising, or even more experienced athletes, practice exercise with an objective. This objective may range from getting fit, losing weight or preparing for a marathon. Is in this moment, when the problem appears:

- how I need to do it?
- What do I need?
- How should I start?

The project intends to provide an answer to those type of questions by preparing long term season to achieve the athletes objective and weekly personalized training plans. The long term seasons are adapted to the athletes availability till the objective, and the weekly personalized training plans to his weekly availability, the moment from the season in which the athlete is and also his physical state.

The report describes the difficulties that arise in solving the problem of generating personalized training plans over the course of a season. First, the derivatives of the number of factors involved, such as the availability of the athlete, the sporting goal or the preferences for the different types of training. The work also explains what it represents at the algorithmic level, and that an algorithm capable of efficiently solving task distribution problems has not been developed at present.

The project will bring to the market a product accessible to everyone in order to promote the sports practice in a healthy, progressively way and avoiding injuries. In that way, from the athlete who starts exercising to the more experienced one, will have a route to follow so as to achieve their personal goals.

Índex

Glossari

El projecte està directament relacionat amb l'esport i per tant, per tal de dissenyar un aplicatiu per la creació automàtica de plans d'entrenament personalitzats, es fa imprescindible conèixer i entendre molts conceptes de l'àmbit esportius. Aquest serà precisament l'objectiu d'aquest glossari. Es recomana com a pas previ a la lectura d'aquesta memòria, comprendre els següents conceptes.

Atleta

En el projecte un atleta és un usuari del sistema. Se'n coneixen els seus paràmetres físics.

Objectiu esportiu

Un objectiu esportiu, dins del projecte, es refereix a un estat de forma que l'atleta vol assolir a llarg termini. Això significa que un objectiu esportiu no serà mai inferior a vuit setmanes. Un exemple seria voler-se preparar per una marató. En aquest cas, l'objectiu esportiu seria aconseguir l'estat físic necessari per realitzar una marató.

Temporada esportiva

Tot el procés d'entrenament que segueix l'alteta, des del principi fins que s'arriba a la data de l'objectiu esportiu establert a llarg termini. Una temporada esportiva està subdividida en períodes d'entrenament.

Període d'entrenament

Conjunt de N setmanes seguides d'una temporada que es dediquen a treballar un o un conjunt d'aspectes físics. El nombre de setmanes sempre és un múltiple de 4, per exemple un període d'entrenament de 4 setmanes per millorar la força. Si fos de 8 setmanes, el sistema ho veu com 4 + 4, si fos 12, el sistema ho veu 4 + 4 + 4 ...

Planificació setmanal

Planificació de sessions esportives en una setmana. Remarcar que una setmana està dins d'un període d'entrenament.

Sessió d'entrenament

És la unitat funcional més petita. Consisteix en una sèrie d'exercicis físics que s'han de realitzar en un dia i una hora determinada.

Les sessions d'entrenament es classifiquen en sessions de qualitat i sessions complementàries.

Sessió de qualitat

Una sessió de qualitat es treballen els aspectes físics més rellevants del període d'entrenament en el qual es troba l'atleta. És la sessió o sessions més importants de la setmana.

Sessió complementaria

Una sessió complementària, tal com diu el nom, complementa el treball físic realitzat en les sessions de qualitat de la setmana.

Relació temporada esportiva, període d'entrenament, planificació setmanal i sessió d'entrenament

Com s'ha explicat anteriorment, el concepte més gran és el de temporada esportiva. Una temporada esportiva està constituïda per períodes d'entrenament. Finalment, un període d'entrenament està format per setmanes i aquestes setmanes s'ompliran progressivament amb planificacions setmanals personalitzades.

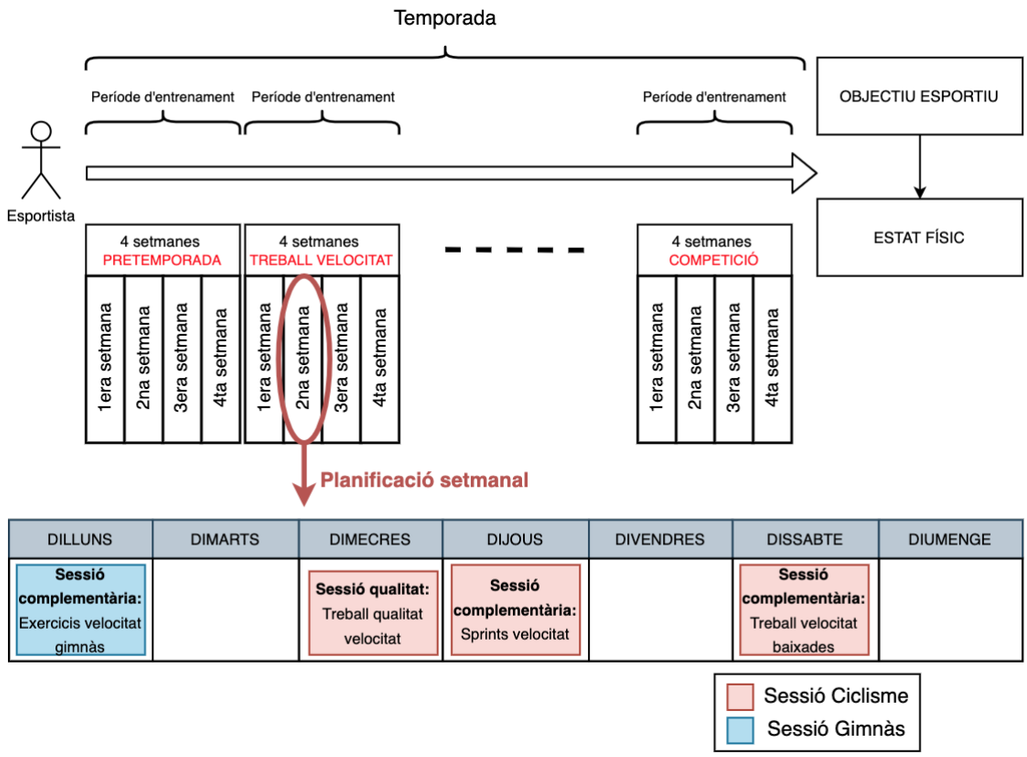


Figura 1: Esquema dels conceptes del glossari

Disponibilitat de l'atleta

Per últim, tenim la disponibilitat de l'atleta. Pel sistema el concepte de disponibilitat es refereix a la disponibilitat en minuts i esport per cada dia de la setmana.

El sistema contempla l'opció d'haver d'escollir un esport un dia o poder-ne fer més d'un. Per exemple, un atleta que els dilluns té 1 hora per anar al gimnàs, anar amb bicicleta o anar a córrer. En el segon cas, un atleta el dilluns pot anar amb bici 1 hora i pot realitzar 1 hora de gimnàs.

Cal remarcar que el fet que tingui disponibilitat no vol dir que el sistema li assigni una sessió d'entrenament.

Marxa Cicloturista

Una marxa cicloturista és una prova de l'esport del ciclisme no competitiva que sol tenir una distància entre 150 i 200 quilòmetres. Pot ser individual o en grup, i l'objectiu és arribar al final del recorregut.

Introducció

1.1 Contextualització

El projecte "Aplicatiu de Creació Automàtica de Planificacions d'Entrenament Personalitzat"és el Treball de Fi de Grau de l'especialitat de Computació dels estudis d'Enginyeria Informàtica [11] a la Universitat Politècnica de Catalunya. És un projecte que s'ha realitzat en cooperació amb l'empresa Trainerer [30], que ofereix a esportistes de tots els nivells la possibilitat d'assolir els seus objectius mitjançant plans d'entrenament personalitzats.

El projecte pretén aportar una extensió sobre la plataforma actual de l'empresa Trainerer per tal d'automatitzar la generació d'entrenaments personalitzats mitjançant un sistema intel·ligent. D'aquesta manera es podrà guiar als atletes al llarg de cada setmana, des de l'inici fins a arribar a assolir els seus objectius esportius. Actualment a l'empresa Trainerer, el disseny de temporades i creació de plans d'entrenament personalitzats està a càrrec d'experts del sector de l'esport.

1.1.1 Treball especialitat de computació

Com s'ha dit prèviament, és un Treball de Fi de Grau de l'especialitat de Computació i per aquest motiu la gran majoria del temps s'ha dedicat a la part d'implementació de temporades esportives i planificacions setmanals personalitzades. Tanmateix, l'estudiant ha desenvolupat la totalitat del projecte i per tant, com aplicatiu en si, s'han tocat aspectes d'altres especialitats com dissenyar una arquitectura del projecte, arquitectura de la base de dades, gestió de connexions asíncrones entre contenidors, disseny d'una interfície...

Com a projecte es fa necessari una explicació completa també d'aquestes altres parts comentades anteriorment, però remarcar que aproximadament un 80% del temps s'ha dedicat a la implementació.

1.1.2 Trainerer

Tal com s'ha comentat anteriorment, el projecte ha estat realitzat en cooperació amb l'empresa Trainerer. Trainerer, anteriorment ADNCiclista, ofereix a esportistes de tots els nivells la possibilitat d'assolir els seus objectius mitjançant plans d'entrenament personalitzats i el posterior seguiment i avaluació d'aquests.

L'objectiu de Trainerer és aconsellar i guiar a l'esportista, des dels inicis fins a nivells de competició, per tal de realitzar entrenaments complets, progressius i evitar lesions. Tot aquest procés es realitza a través de plans setmanals personalitzats

adaptats a les característiques físiques, objectius i disponibilitat dels atletes.

1.1.3 Entrenaments personalitzats

El projecte es basa en la creació d'entraments setmanals personalitzats, per aquest motiu és de gran importància deixar clar el concepte d'entrenament personalitzat.

Un entrenament personalitzat és un conjunt de sessions d'entrenament, adaptades al nivell i condició física de l'atleta, que l'hi permet arribar a un objectiu d'una manera progressiva, efectiva temporalment i sobretot, evitant lesions. D'aquesta manera, i tal com s'explicarà amb més detall en l'apartat d'objectius, aquest serà el principal objectiu del projecte.

1.1.4 Cost econòmic

Un altre problema per el qual les persones que s'inicien o fins i tot atletes de més alt nivell és que no disposen d'un pla d'entrenament personalitzat per potenciar els seus objectius. Tenir un entrenador personal que ens realitzi planificacions a nivell de temporada i que setmana rere setmana avaluï la nostra progressió per programar sessions d'entrenament té un cost molt elevat.

Aquest projecte, una mica més a llarg termini, intentarà que tothom pugui tenir accés a plans d'entrenaments específics i personalitzats a un cost molt reduït.

1.2 Motivació

Des de ben petit he tingut dues grans passions, les noves tecnologies i l'esport. Per una banda, abans de començar els estudis universitaris, tenia molt clar que volia estudiar el grau d'Enginyeria Informàtica, ja que com deia, sempre m'han interessat molt les noves tecnologies. El treball de recerca de primer de batxillerat ja el vaig enfocar al desenvolupament d'una aplicació mòbil per la gestió de tornejos i classificacions del món del tennis. Aquest treball em va confirmar el que ja sospitava, havia d'encaminar el meu futur cap en aquesta direcció.

Referent a l'esport, i també des de ben petit, he entrenat, jugat i competit a tennis. Des del meu punt de vista, el tennis com a esport individual, m'ha ajudat molt a superar-me i a aconseguir físicament i mentalment objectius i reptes. Tot i això, al començar a estudiar a la Universitat Politècnica de Catalunya em vaig veure obligat a deixar la competició del tennis i seguir entrenant ocasionalment.

Quan estava al quart curs del grau, vaig començar a buscar una empresa tecnològica on pogués posar en pràctica els coneixents adquirits durant la carrera i guanyar experiència dins del món laboral. En aquest moment va aparèixer l'empresa Trainerer amb el projecte ja en marxa. Després d'unes entrevistes i de conèixer més a prop la plataforma en si, vam formalitzar la idea per tal de realitzar-ho com a treball de fi de grau. La veritat és que desenvolupar un projecte que enllaça les meves dues grans passions i que a més em permet viure el dia a dia rodejat de persones tècniques en l'àmbit de la informatica i apassionats de l'esport, afegeix una motivació encara més gran.

En conclusió, tenir la possibilitat de realitzar el treball de fi de grau en una empresa tecnològica del sector esportiu, on seguir guanyant experiència i coneixements,

i poder aplicar tot l'après durant els quatre anys del grau d'Enginyeria Informàtica a la Universitat Politècnica de Catalunya, em va semblar una gran oportunitat.

1.3 Problema a resoldre

Després de les seccions anteriors on s'ha posat el projecte en context, ja es pot concretar sobre quin és realment el problema a resoldre.

Al món hi ha una gran quantitat de persones que en algun moment o altre han pres la decisió de realitzar esport. En la majoria dels casos la decisió de fer esport bé lligada d'un objectiu, com podria ser aprimar-se, posar-se en forma o fins i tot preparar-se per alguna carrera com per exemple una marató. Arribat aquest punt, molt freqüentment apareix alguna de les preguntes següents:

- Com ho faig?
- Què necessito?
- Com ho haig de fer per evitar lesions?
- Com ho haig de fer perquè sigui de manera progressiva i adaptada a les meves necessitats?

És en aquests aspectes on apareix el problema a resoldre. Si ho mirem des del punt de vista esportiu, són les preguntes que resoldria un entrenador. Sempre que l'entrenador estigués en tot moment pendent dels entrenaments i evolució de l'atleta, el que es coneix com a entrenador personal. És evident que totes les persones amb aquesta necessitat no tenen la possibilitat econòmica o la disponibilitat per tal d'assessorar-se amb un entrenador personal.

El projecte intenta solucionar aquest problema mitjançant la planificació de sessions d'entrenament personalitzades per tal de guiar a l'esportista cap a la seva meta.

Trainerer vol fomentar la pràctica d'esport en la societat, però sempre de manera saludable, progressiva, amb entrenaments correctament planificats per tal de complir l'objectiu dels atletes i evitar possibles lesions.

Actualment, Trainerer disposa d'una plataforma on els atletes es poden comunicar amb l'entrenador personal de Trainerer, on poden observar les sessions d'entrenament que l'entrenador els hi ha assignat i també una anàlisi de les dades (freqüència cardíaca, potència, TSS [31], durada de l'entrenament...) dels seus últims entrenaments. Té sentit que el projecte sigui una extensió d'aquesta plataforma que ofereixi la possibilitat d'automatitzar la tasca de generar entrenaments personalitzats per l'atleta mitjançant un sistema intel·ligent.

1.3.1 Dificultat d'un problema de distribució de tasques

El plantejament anterior, implica estructurar el problema com un problema de distribució de tasques. A nivell informàtic, aquest tipus són coneguts per la seva complexitat a l'hora de resolra'ls i verificar possibles solucions. Aquest ha estat el repte més gran per aquest projecte i a on s'han dedicat gran part dels recursos.

Estat de l'art

Per tal de conèixer una mica més el sector i comprendre del potencial d'aquest mercat, convé estudiar les diferents alternatives que un atleta té actualment a la seva disposició. En els següents apartats trobem algunes de les principals alternatives explicades amb els seus punts forts i les seves mancances. Per últim, es realitzarà una comparativa mitjançant una taula de totes les alternatives esmentades.

2.1 Entrenadors personalitzats

Tal com s'ha explicat prèviament en l'apartat 1.3, un entrenador personal seria una solució actual que funciona. El problema, és que no està a l'abast de la majoria de la població. Tenir un entrenador que estudi temporades completes (2 a 12 mesos normalment) per maximitzar el potencial de l'atleta i complir els seus objectius, té un cost molt elevat. A més a més, s'ha de tenir en compte que es necessita una avaluació constant en els diferents esports que practiqui l'atleta.

2.2 Strava

Per una banda hi hauria les aplicacions mòbils, com és el cas de Strava [47]. Strava és una aplicació mòbil per corredors i ciclistes, que en els seus inicis va començar com una xarxa social per esportistes on compartir les rutes i entrenaments.

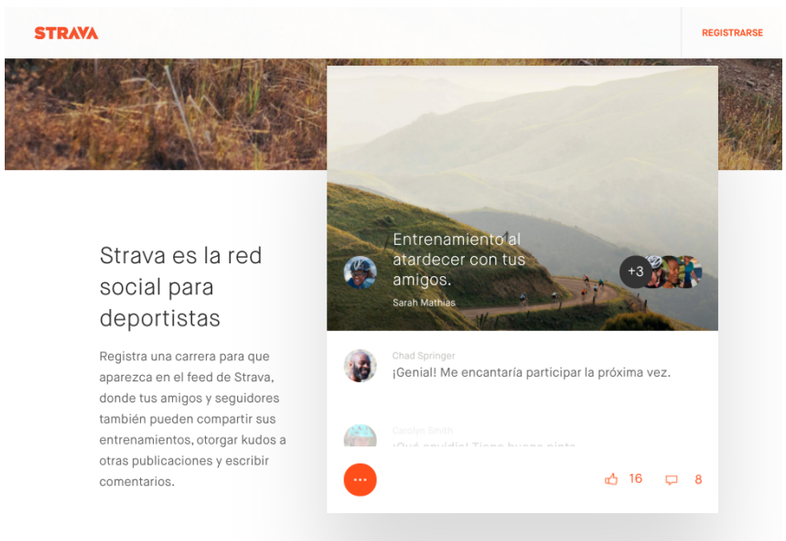


Figura 2.1: Strava com a xarxa social.

Actualment Strava ha fet un pas més i ofereix als atletes un cop han acabat l'entrenament, una valoració en forma de paràmetres físics per tal que l'atleta pugui valorar el seu estat.

STRAVA

Funciones	Gratis	Suscripción
Registro de actividades	✓	✓
Soporte de dispositivos	✓	✓
Red social	✓	✓
Planificación de rutas		✓
Competición en segmentos		✓
Panel de entrenamiento		✓
Análisis del ritmo cardiaco y potencia		✓
Mediciones avanzadas		✓
Definición de objetivos		✓
Registro de entrenamiento		✓
Comparación de esfuerzos		✓
Beacon		✓
Mapas de actividad personal		✓
Ventajas ofrecidas por los socios		✓
Ayuda Premium		✓

Figura 2.2: Funcionalitats de Strava gratuïtes i amb subscripció mensual.

Tal com es pot veure en la figura 2.2 l'anàlisi dels paràmetres físics no és gratuït, sinó que està disponible a través d'una subscripció mensual de 5 euros. Tot i això, el preu és raonable per la gran quantitat d'informació que l'esportista rep dels seus entrenaments.

Finalment, cal remarcar que Strava a part de realitzar el seguiment no té cap servei de programació de planificacions ni sessions d'entrenament. En el seu cas, compten amb l'usuari o d'un entrenador personal extern a ells per realitzar aquesta tasca. Strava doncs, no resoldria el problema identificat en el capítol anterior.

2.3 Runtastic

Una altra opció dins del mercat de les aplicacions mòbils per esportistes és Adidas Runtastic [42]. A diferència de Strava, Adidas Runtastic no està tan orientada a xarxa social sinó més a entrenar-se i complir objectius esportius.

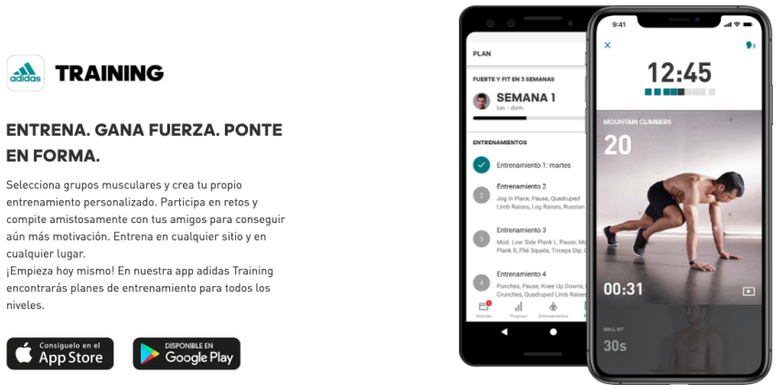


Figura 2.3: Descripció funcionalitats Runtastic

En la figura anterior(Figura [42]), Adidas Runtastic ofereix grups d'entrenaments als seus usuaris per tal que puguin aconseguir els seus objectius. A més a més, un cop acabat l'entrenament, l'usuari disposa dels diferents paràmetres físics enregistrats durant l'activitat. De la mateixa manera que Strava, per tenir la funcionalitat d'entrenaments, cal pagar una quota mensual de 4,99 euros.

Runtastic a diferència de Strava, enregistra menys paràmetres físics, només quedant-se amb els més bàsics que no requereixen un gran coneixement tècnic per poder entendre.

Per últim, remarcar que és veritat que runtàstic ofereix entrenaments predefinits, però el sistema no és capaç fer recomanacions de plans d'entrenament donada les necessitats físiques i objectius de l'usuari. També afegir que els entrenaments són més enfocats en l'àmbit de preparació física i no tant específics a un esport concret, com podria ser el ciclisme i o running. Per tant, tampoc seria una solució vàlida al problema plantejat en l'apartat anterior.

2.4 Comparativa final

Finalment es realitzarà una comparativa final mitjançant una taula, comparant les opcions de l'entrenador personal, Strava i Adidas Runtastic amb el que vol aconseguir aquest projecte dins l'empresa Trainerer.

	Entrenador personal	Strava	Adidas Runtastic	Projecte (Trainerer)
Oferta d'entrenaments	Si	No	Si	Si
Entrenaments personalitzats	Si	No	No	Si
Anàlisi i monitoratge	No	Si	Si	Si
Feedback de la temporada	Si	No	No	Si
Preu accessible	No	Si	Si	Si

És important remarcar que la part d'anàlisi de les dades dels entrenaments programats i el feedback de com l'atleta està rendint a mesura que avança la temporada, és extern a aquest projecte. Tot i això, s'ha decidit incloure en aquesta comparativa per poder realitzar una comparativa completa amb les opcions de marcat.

2.5 Justificació

El món de l'esport és un món complex, ja que per un sistema informàtic és difícil definir una persona amb paràmetres físics. Aquest és segurament el motiu pel qual, tot i haver-hi aplicacions i serveis al mercat per fer i monitorar l'exercici, encara cap d'ells inclou plans d'entrenament personalitzat per diferents esports.

És cert que per exemple Adidas Runtastic ofereix plans d'entrenament, però com s'ha comentat en la seva anàlisi, no ofereix als usuaris temporades d'entrenament a llarg termini (2 a 12 mesos) ni recomana programes d'entrenament.

Pel que fa al cas dels entrenadors personals, és cert que en la majoria de casos són una solució pel problema. Un entrenador personal és capaç de planificar-te una temporada i cada setmana dissenyar plans d'entrenament adaptats al nivell de l'atleta per tal d'aconseguir els seus nivells esportius. El principal obstacle és que un entrenador personal no està a l'abast de tothom, ja que per realitzar aquestes tasques el preu és relativament alt.

En conclusió, un cop estudiat totes les alternatives presents actualment al mercat, s'ha pogut observar la mancança de les aplicacions esportives en dissenyar temporades i plans d'entrenament setmanals que s'ajustin a les necessitats i objectius dels esportistes.

Definició de l'abast

En aquest capítol s'explicarà fins a on pot arribar el projecte, detallant el conjunt d'objectius amb els seus requisits funcionals i no funcionals corresponents. A continuació es detallran tots els agents que es veuen implicats en el desenvolupament d'aquest projecte, dels quals s'intentarà definir també quin paper hi juguen. Finalment, es detallarà els obstacles i els riscs que es contemplen abans del desenvolupament del projecte i quina metodologia i seguiment s'utilitzarà durant el desenvolupment d'aquest.

3.1 Objectius

Es dividiran els objectius en objectiu prinicipal i objectius secundaris, amb la intenció de deixar clar l'objectiu principal envers la resta.

3.1.1 Objectiu principal

Un cop contextualitzat i analitzat l'estat de l'art, l'objectiu del projecte ja queda més ben definit. Aquest objectiu principal doncs, passa per la creació d'un aplicatiu o sistema que sigui capaç de planificar temporades a llarg termini en funció de l'objectiu de l'atleta i completar-la amb planificacions setmanals adaptades al nivell i necessitats físiques de l'atleta.

Si un atleta es volgués preparar per una marató, el sistema hauria de dissenyar una temporada dividint les setmanes de les quals disposa en diferents períodes d'entrenament, on en cadascún dels períodes es treballin diferents factors per tal de preparar-se el millor possible per l'objectiu. A continuació, setmana a setmana, d'acord amb la disponibilitat que tingui l'atleta i les seves capacitats físiques, es realitzaria plans d'entrenament setmanal personalitzats.

3.1.2 Objectius secudaris

Ja definit l'objectiu del projecte, a continuació es passarà a identificar tot el conjunt d'objectius secudaris que intenten completar i enmarcar una mica més el desenvolupament d'aquest projecte.

3.1.2.1 Fàcil integració del projecte amb la plataforma de Trainerer

El projecte s'ha de poder integrar amb facilitat dins l'ecosistema actual en el que treballa Trainerer. És un objectiu secundari clar, donat que l'aplicatiu no té sentit si no es pot integrar amb la plataforma que l'empresa té actualment en funcionament. Això suposarà integrar aquest projecte amb el de Trainerer per tal de tenir accés a tota la informació que necessita aquest.

3.1.2.2 Integrable amb el projecte de David Mingueza

Tal com s'ha explicat en el capítol de contextualització, aquest projecte es realitza junt amb el projecte de l'estudiant David Mingueza. És necessari tenir present que en un futur, l'aplicatiu d'aquest projecte s'ha d'integrar amb l'ànalisi d'entrenaments que realitza el seu projecte.

3.1.2.3 Aplicar coneixements

El projecte és el treball de fi de grau del desenvolupador. Aquest fa que un dels objectius del projecte sigui poder aplicar els conneixements adquirits durant els anys d'estudi i adquirir-ne encara més.

3.2 Requisits

Els requisits del projecte estan definits amb la finalitat de complir l'objectiu principal i secundaris definits en l'apartat anterior. Els requisits s'han dividit en requisits funcionals i no funcionals. Els requisits funcionals són els requisits indispensables per tal de que el projecte es dugui a terme correctament i poder complir els objectius. Per l'altra banda, els objectius no funcionals són aquells que si bé no són indispensables pel correcte funcionament del projecte, s'haurien de tenir en compte pel millor funcionament i desenvolupament d'aquest.

3.2.1 Requisits funcionals

A continuació es datallaràn el conjunt de requisits funcionals del projecte:

3.2.1.1 Generar planificacions de temporades automàticament

Per complir l'objectiu principal del projecte és necessari que el sistema sigui capaç de planificar temporades a llarg termini, aíxi com els períodes d'entrenament pels quals està formada una temporada. D'aquesta manera, els atletes usuaris del servei, tindran definit el camí per assolir els seus objectius.

3.2.1.2 Generar planificacions setmanals automàticament

En la mateixa direcció que el requisit funcional anterior, un cop s'ha definit la temporada de l'atleta, el sistema ha de realitzar planificacions setmanals que s'adaptin a la disponibilitat de l'atleta automàticament.

3.2.1.3 Gestionar la disponibilitat dels usuaris

El sistema ha de poder llegir i gestionar la disponibilitat dels usuaris setmanalment. Es fa impossible realitzar planficacions setmanals de sessions esportives d'entrenament sense tenir la disponibilitat dels atletes.

3.2.1.4 Gestionar els objectius dels usuaris

El sistema ha de poder llegir i entendre els objectius esportius dels usuaris. Es fa imprescindible per tal de poder dissenyar temporades i planificar entrenaments que s'ajustin a l'objectiu d'aquests.

3.2.1.5 Accés a planificacions setmanals predefinides

El sistema també ha de tenir accés a les planificacios setmanals predefinides per un expert en l'entrenament de l'esport. Acutalment ja estan disponibles a la plataforma que utilitza actualment l'empresa Trainerer.

3.2.2 Requisits no funcionals

A continuació es datallaràn el conjunt de requisits funcionals del projecte:

3.2.2.1 El projecte tingui estructura de microservei

El sistema ha de tenir una estructura de microservei per tal que es pugui integrar amb facilitat a la plataforma ja existent de Trainerer. Aquesta estructura també presenta els següents avantatges:

- El sistema sigui escalable. És a dir, que el número d'atletes (usuaris de Trainerer) que necessiten planificacions no suposi un problema en cas d'aquest ser molt gran.
- El sistema sigui testajable. Un microservei es pot comprobar que funciona correctament tant abans de posar-lo en funcionament en la plataforma com durant el seu funcionament.
- El sistema sigui mòdulable. Un microservei és fàcilment integrable amb altres microserveis (altres extensions) o millorar el sistema existent.

3.2.2.2 El projecte sigui eficient

El projecte hauria de ser eficient tant en l'àmbit de memòria com de temps d'execució.

3.2.2.3 El projecte sigui fiable

El projecte hauria de ser fiable o bastant fiable, ja que els usuaris finals hauran de realitzar un esforç físic per la planificació generada pel sistema.

3.2.2.4 El projecte sigui segur

El projecta hauria de ser segur i no permetre sota cap concepte l'accés a persones no autoritzades. El sistema tracta amb dades físiques i fisiològiques dels esportives i per tant no poden sortir fora del mateix sistema.

3.2.2.5 El projecte sigui mínimament usable

El projecte ha de ser usable en l'àmbit de llegir i entendre la planificació generada pel sistema.

3.3 Abast

En el primer capítol s'ha posat en context el projecte, a continuació s'ha analitzat les diferents alternatives presents al mercat i per últim s'ha definit els objectius i els requistis que ha de tenir el projecte. Un cop arribats en aquest punt podem dividir el projecte en dues grans funcionalitats. Per una banda, el sistema de creació de temporades automàtiques i per l'altra, el sistema de planificació de sessions setmanals.

3.3.1 Creació automàtica de temporades

En primer lloc, el sistema ha de ser capaç de d'entendre l'objectiu esportiu que un atleta podria tenir. Cada objectiu dins del sistema ha de tenir també una data per la qual, sempre que sigui possible, l'atleta hauria de complir. Per altra banda es fa evident que també s'han de contemplar unes franges de temps mínims per aconseguir segons quin tipus d'objectius. Per exemple, el cas de preparar-se per una martató, per la gran majoria d'atletes no es pot realitzar en dos mesos sino que es necessita molt més temps.

Un cop implementat aquest component en el sistema, Trainerer serà capaç d'oferir ja un servei que es diferencia de totes les opcions que hi ha al marcat. A més a més, aquesta funcionalitat també aconseguiria un cert compromís i fidelitat als usuaris per tal de seguir utilitzant els plans setmanals d'entrenament fins que s'arribés al final de la temporada.

3.3.2 Creació de planificacions setmanals de sessions d'entrenament

Un cop l'atleta té establert una temporada per assolir el seu objectiu esportiu, toca començar a entrenar seguint els plans setmanals d'entrenament personalitzat. El sistema en aquest punt ha de poder classificar l'atleta en un nivell segons l'esport el qual practica. A més a més, s'haurien de tenir en compte les seves preferències alhora d'entrenar, com per exemple sessions curtes o sessions llargues, més dies de descans o menys ...

La disponibilitat de l'atleta també és una informació clau en aquesta part. El sistema ha de poder entendre la dispoinibilitat dels atletes per tal de seleccionar la millor planificació segons el moment de la temporada en la qual l'atleta es trobi.

Per últim també s'ha de tenir en compte algun mecanisme per valorar les planificacions setmanals que s'assignen als atletes. Per tal de poder assignar una planificació setmanal a un atleta, es fa imprescindible saber quan una planificació és millor que una altra.

Tots aquests factors es diferencien dels serveis actuals disponibles al mercat on la majoria es basen en deixar que l'usuari sel·leccioni el pla d'entrenament que més li agrada, deixant de banda els plans d'entrenaments que millor s'adapten a ell i al seu objectiu.

3.4 Agents implicats

A continuació s'anomenen totes les persones, organitzacions i empresa involucrades i/o afectades pel desenvolupament i posada en marxa del projecte.

3.4.1 Director del projecte

El director del projecte, José María Rodríguez i precisament pel seu rol en el projecte, és un dels principals interessats en el correcte desenvolupament i funcionament del projecte. Ocupa el càrrec de director de tecnologia (CTO) a Trainerer i donarà supervisió dia rere dia al desenvolupament del projecte.

3.4.2 Ponent del projecte

El ponent del projecte, Enrique Romero Merino [33], professor a la Facultat d'Informàtica de Barcelona i a la Facultat de Matemàtiques i Estadística. Es va involucrar des del principi, abans de tenir el vist-i-blau de la universitat. Durant tot el període del projecte supervisarà el correcte desenvolupament del projecte i intentarà aportar possibles millores i extensions.

3.4.3 Equip desenvolupador

Tots els membres de l'equip de desenvolupament, pel simple fet de ser un equip, també tenen una implicació directa amb el correcte desenvolupament del projecte. A més a més, com hem comentat abans a l'hora de definir el problema, el projecte és una extensió de l'actual plataforma i sistema de Trainerer, el qual ha estat desenvolupat pel mateix equip.

3.4.4 Trainerer

Trainerer com empresa és una de les principals parts interessades en el correcte desen- volupament del projecte. Es beneficia d'afegir valor el seu actual producte amb la nova funcionalitat que ha d'aportar el projecte, i també obrirà les portes a un nou model de funcionament en la part d'automatitzar la generació de planificacions d'entrenaments i no dependre d'entrenadors.

3.4.5 Usuaris

Esportistes actuals que utilitzen alguns dels serveis i mètodes d'entrenament de Traine- rer o possibles clients en un futur, es beneficiaran de tenir planificacions d'entrenament personalitzades per tal d'assolir més fàcilment els seus objectius d'una manera correcte i sana.

3.4.6 Entrenadors personals

Entrenadors personals els quals realitzen les funcions que el projecte té com a objectius. Tot i que com s'ha comentat anteriorment la majoria d'atletes no tenen l'accés

degut a l'elevat cost econòmic, a llarg termini es podrien veure afectats si els atletes els substituissin per aquest sistema.

3.5 Obstacles i Riscs

S'exposarà en forma de llista els possibles riscs i obstacles que poden sorgir en el desenvolupament o posada en marxa del projecte en la plataforma de Trainerer. Aquests fets podrien fer perillar la viabilitat del projecte.

3.5.1 Temps d'execució limitat

El temps d'execució a l'hora de generar les planificacions esportives personalitzades està limitat, donada la gran quantitat de planificacions que s'hauran de realitzar setmanalment. Aquest fet també implica que el sistema ha d'estar altament optimitzat, ja que en cas contrari podria significar un problema considerable.

3.5.2 Falta d'experiència en la creació de microserveis

El fet que algunes persones de l'equip de desenvolupament no siguin expertes en la gestió i creació de microserveis, és un fet que endarrereix el desenvolupament del sistema pel temps d'aprenentatge que requereix.

3.5.3 Problema amb les dades

El fet de que les dades no aportin la informació necessària que s'havia previst per gestionar el sistema de generació de planificacions.

3.5.4 Temps de desenvolupament limitat

És un projecte que s'ha de seguir desenvolupant i millorant un cop ja s'hagi entregat aquest document. Aquest fet fa que si hi ha algun inconvenient, el temps de desenvolupament es pugui posar en contra del projecte.

3.5.5 Problemes amb la integració a la plataforma de Trainerer

El projecte té la necessitat de poder integrar-se amb la plataforma actual de Trainerer, ja que necessita accés a la informació de per exemple als atletes o de les planificacions setmanals predefinides. Si per algun inconvenient no es pogués realitzar, el projecte es podria veure molt afectat.

3.6 Metodologia i rigor

Per tal de poder realitzar el desenvolupament del projecte de manera estable i continuada, es defineix la metodologia i rigor que el desenvolupador seguirà durant tot el transcurs del projecte. A continuació es detallen les eines utilitzades segons la funció que aporten en el desenvolupament.

3.6.1 Metodologia de desenvolupament: Gitlab

Gitlab [15] com a eina de manteniment principal del projecte. Mitjançant diferents *branches*, el GitLab permetrà mantenir en tot moment les diferents parts del desenvolupament del projecte:

- Master branch, on es mantindrà la versió definitiva del projecte. Tots els progressos que es vagin realitzant, un cop siguin comprovats i verificats s'inclouran en aquesta branca.
- Altres branch, per explorar diferents alternatives i/o treballar en diferents parts del projecte simultàniament.

Per últim el GitLab també permetrà tenir un històric de les versions del projecte, per si en cas de produirse algun imprevist, tenir la possiblitat de restaurar alguna versió.

3.6.2 Metodologia de desenvolupament: Trello

Trello es una plataforma online on mitjançant taulers compartits es poden organitzar les tasques i treballar de forma colaborativa amb més facilitat. S'utilitzarà prinicpalment per organitzar les tasques. Les tasques es categoritzen en quatre llistes:

- Tasques per fer.
- Tasques que s'estan realitzant.
- Tasques que falta testejar i verificar.
- Tasques tancades.

D'aquesta manera, l'equip de desenvolupament té coneixement en tot moment sobre quines tasques està treballant cadascú.

3.6.3 Metodologia de desenvolupament: Slack

Slack [44] és una plataforma de comunicació organitzada en canals i orientada al desenvolupament de projectes en equip. Durant el desenvolupament del projecte s'utilitzarà per mantenir la comunicació entre els membres de l'equip de desenvolupament de Trainerer.

També s'habilitarà un canal per resoldre incidències que puguin sorgir durant el desenvolupament del projecte.

3.6.4 Metodologia de desenvolupament: Python

El projecte serà desenvolupat en Python 3.7. Python recull en una guia [39] tot una sèrie de normes alhora de desenvolupar un projecte. Aquestes normes inclouen com organitzar el projecte, com han de ser el nom de les variables, funcions o classes, com realitzar els comentaris i documentació... Tot el projecte s'ha desenvolupat seguint els seus estàndars.

Planificació temporal del projecte

4.1 Duració del projecte

El projecte es desenvoluparà en l'empresa Trainerer entre el 13 de Gener i el 12 de juny de 2020, en un període de 5 mesos que equivalen a 106 dies laborables. La dedicació total en el desenvolupament del projecte en l'empresa és de 735 hores, que tenint en compte el nombre de dies laborables calculats, resulta en 7 hores diàries de treball. Per part de la universitat, es fa imprescindible realitzar l'assignatura de Gestió de Projecte (GEP) [1] que té una dedicació definida de 75h.

En total, tenint en compte els paràmetres anteriors, s'estima una dedicació de 810 hores (735h + 75h) entre el 13 de Gener i el 12 de juny de 2020. A continuació s'ha dividit el desenvolupament en diferents tasques.

4.2 Definició de les tasques

En aquesta secció es divideix el desenvolupament del projecte en tasques més petites dins de Sprints. De cada tasca s'especificarà una breu descripció, les seves dependències, la duració estimada, l'assignació de responsabilitats i els recursos necessaris. De cada Sprint s'especificarà la seva durada en dies i hores.

4.2.1 Gestió de projecte: 17 de Febrer - 16 de Març 2020

- **Descripció:** Documentació escrita. Inclou la contextualització, l'abast, la planificació temporal, el pressupost i l'informe de sostenabilitat.
- **Duració:** 75 hores
- **Dependències:** Cap.
- **Recursos:** Ordinador, Overleaf (Editor Latex) [36].
- **Assignació:** Cap de projecte.

4.2.2 Memòria: 17 de Març - 30 de Juny 2020

- **Descripció:** Documentació escrita del projecte i preparació de la prestentació.
- **Duració:** S'ha aproximat en unes 3-4 hores setmanals, amb excepció de les últimes 3 setmanes que hi haurà dedicació casi completa. S'estima un total de 100 hores.

- **Dependències:** Gestió de Projecte.
- **Recursos:** Ordinador, Overleaf (Editor Latex) [36].
- **Assignació:** Cap de projecte.

4.2.3 Sprint 1: 13 de Gener - 17 de Gener de 2020

- **Descripció:** Contextualitzar el projecte a nivell d'empresa, definir objectius del projecte dins l'empresa, definir el mètode de treball dins l'empresa i definir totes les parts i elements necessàris.
- **Duració:** Durada de 7 dies a 7 hores de dedicació per dia. Resulta en un total de 35 hores.
- **Dependències:** Cap
- **Recursos:** Ordinador, Overleaf (Editor Latex) [36].
- **Assignació:** Cap de projecte.
- **Tasques:**

Tasca	Descripció	Dependència	Duració (hores)
T-11	Reunions		10
T-12	Definir estructura i llenguatge de programació	T-11	5
T-13	Definir i parametrizar amb dades	T-11	10
T-14	Revisió Sprint	T-11,T-12,T-13	5

Figura 4.1: Taula Sprint 1

4.2.4 Sprint 2: 20 de Gener- 7 de Febrer de 2020

- **Descripció:** Treball previ al desenvolupament del projecte. Abans de començar a desenvolupar el projecte a nivell de programació cal apendre els frameworks, entorns i llenguatges de programació que s'utilitzaran
- **Duració:** 15 dies laborables a 7 hores treballades per dia. Total de 105 hores.
- **Dependències:** Sprint 1
- **Recursos:** Ordinador.
- **Assignació:** Programador
- **Tasques:**

Tasca	Descripció	Dependència	Duració (hores)
T-21	Reunions		15
T-22	Entendre l'entorn actual (amb tots els microserveis)	T-21	20
T-23	Entendre la connexió entre els diferents contenidors (docker)[50]	T-21,T-22	25
T-24	Entendre l'estructura de les bases de dades actuals i els valors de les taules corresponents.	T-21,T-22	15
T-25	Entendre el concepte de ORM (Object Oriented Mapping)[51] i estudiar l'accés d'aquests a les bases de dades.	T-21,T-22, T-24	25
T-46	Revisió Sprint.	T-21, T-22, T-23, T-24, T-25	5

Figura 4.2: Taula Sprint 2

4.2.5 Sprint 3: 10 de Febrer - 28 de Febrer de 2020

- **Descripció:** Creació de l'entorn del projecte per desenvolupar el microservei, creació o verificar la connexió a les bases de dades necessessàries i definir els patrons de disseny del projecte.
- **Duració:** 15 dies laborables a 7 hores treballades per dia. Total de 105 hores.
- **Dependències:** Sprint 2.
- **Recursos:** Ordinador.
- **Assignació:** Programador
- **Tasques:**

Tasca	Descripció	Dependència	Duració (hores)
T-31	Reunions		15
T-32	Creació d'un nou contenidor dins l'entorn actual.	T-31	25
T-33	Creació o verificació de la conneixió a les bases de dades i taules corresponents.	T-31	25
T-34	Definir patrons de disseny del projecte [52].	T-31, T-32, T-33	25
T-35	Revisió Sprint.	T-31, T-32, T-33, T-34	20

Figura 4.3: Taula Sprint 3

4.2.6 Sprint 4: 2 Març - 24 d'Abril de 2020

- **Descripció:** Implementació del sistema d'optimització amb restriccions. Definir funció objectiu, restriccions, sel·leccionar algorisme...
- **Duració:** 8 setmanes a 7 hores treballades per dia. Total de 280 hores.
- **Dependències:** Sprint 3.
- **Recursos:** Ordinador.

- **Assignació:** Programador.
- **Tasques:**

Tasca	Descripció	Dependència	Duració (hores)
T-41	Reunions		30
T-42	Creació dels objectes i models necessàris pels accesos a les bases de dades	T-41	40
T-43	Representació de la disponibilitat de l'atleta en forma de matriu	T-41, T-42	25
T-44	Creació i connexió a la taula dels resultats dels entrenaments anteriors dels atletes	T-41	40
T-45	Creació de la funció objectiu a omptimitzar	T-41	40
T-46	Creació del sistema d'optimització amb l'algorisme més eficient.	T-41, T-45	165
T-47	Revisió Sprint.	T-41,T-42,T-43,T-44,T-45,T-46	40

Figura 4.4: Taula Sprint 4

4.2.7 Sprint 5 27 d'Abril - 15 de Maig de 2020

- **Descripció:** Donada la complexitat i la granularitat de les parts del projecte, l'Sprint 5 és considerat un període de marge pels imprevistos. Remarcar que hi ha la llista completa de riscs en l'apartat d'obstacles i riscs de l'abast.
- **Duració:** 15 dies a 7 hores treballades per dia. Total de 105 hores.
- **Dependències:** Sprint 4

4.2.8 Sprint 6: 18 de Maig - 29 de Maig de 2020

- **Descripció:** Posada en marxa en el sistema i servidor.
- **Duració:** 10 dies a 7 hores treballades per dia. Total de 70 hores.
- **Dependències:** Sprint 5
- **Recursos:** Ordinador, servidor de l'empresa.
- **Assignació:** Programador i Enginyer de sistemes.
- **Tasques:**

Tasca	Descripció	Dependència	Duració (hores)
T-61	Reunions		15
T-62	Migració del servei al servidor.	T-61	20
T-63	Instal·lació de dependències necessàries	T-61, T-62	10
T-64	Comprovar les connexions del microservei amb totes les parts	T-61,T-62,T-63	20
T-65	Revisió Sprint.	T-61, T-62, T-63, T-64	5

Figura 4.5: Taula Sprint 6

4.2.9 Sprint 7: 1 de Juny - 12 de Juny de 2020

- **Descripció:** Realització de proves amb casos extrems. Anàlisi i estudi dels resultats.
- **Duració:** 10 dies a 7 hores treballades per dia. Total de 70 hores.
- **Dependències:** Sprint 6.
- **Recursos:** Ordinador.
- **Assignació:** Programador, Cap de projecte i Enginyer de sistemes.
- **Tasques:**

Tasca	Descripció	Dependència	Duració (hores)
T-71	Reunions		15
T-72	Realització de proves amb casos extrems	T-71	30
T-73	Anàlisi i estudi dels resultats	T-71, T-72	20
T-74	Revisió Sprint.	T-71, T-72, T-73	5

Figura 4.6: Taula Sprint 7

4.3 Diagrama de Gantt

A continuació es realitzaran els corresponents diagrames de Gantt per cadascun dels Sprints desglossats anteriorment. Remarcar també que tant la part de l'assignatura de Gestió de projecte, la part de redactar la memòria i l'Sprint 5 (temps de marge) no hi ha diagrama de Gantt donat que no s'han desglossat en tasques.

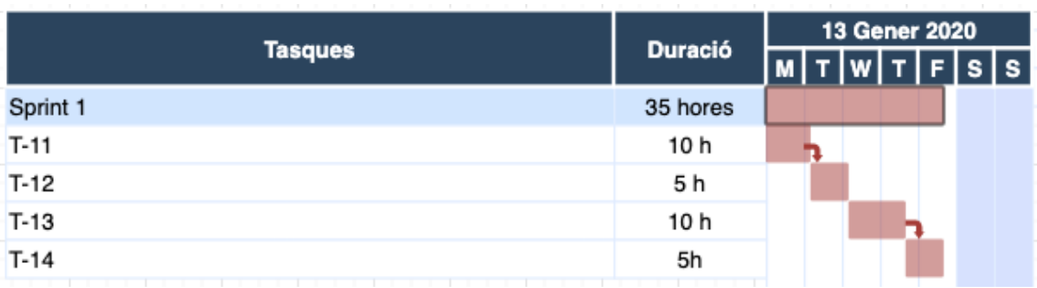


Figura 4.7: Diagrama de Gantt de Sprint 1

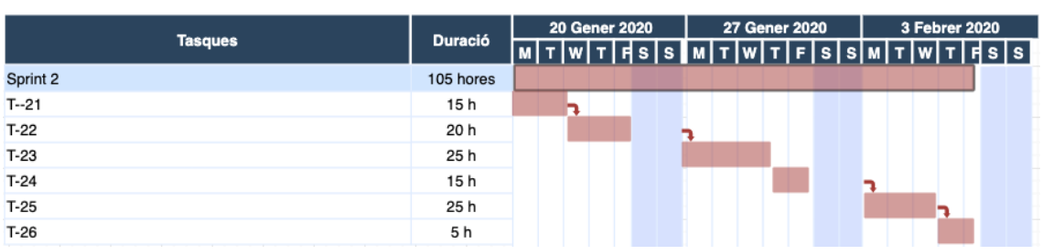


Figura 4.8: Diagrama de Gantt de Sprint 2

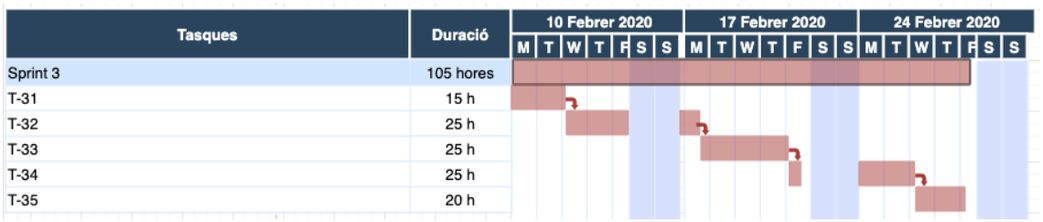


Figura 4.9: Diagrama de Gantt de Sprint 3

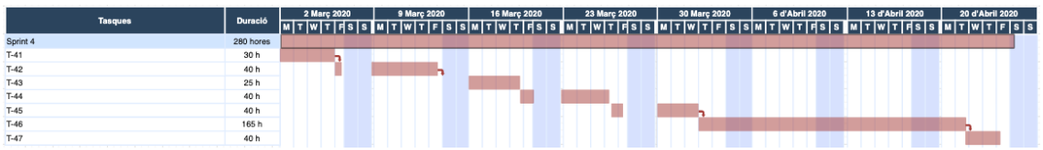


Figura 4.10: Diagrama de Gantt de Sprint 4

Figura 4.11: Diagrama de Gantt de Sprint 6

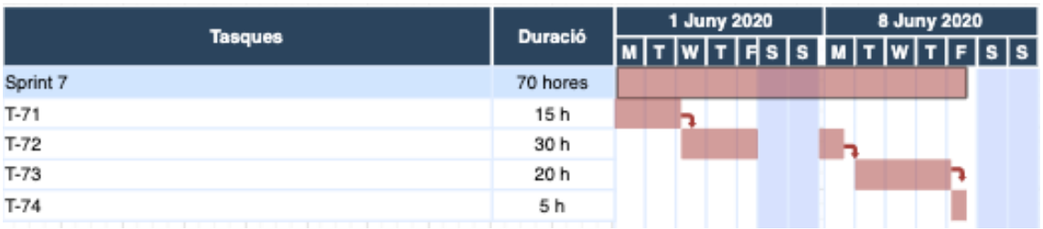


Figura 4.12: Diagrama de Gantt de Sprint 7

Pressupost

A continuació es descriu el procés i el conjunt d'elements que s'han tingut en compte per l'elaboració del pressupost del projecte. El càlcul d'aquest ve estimat pel nombre d'hores de cada tasca i la seva corresponent assignació. Finalment també es proposarà una metodologia per controlar possibles desviacions de les estimacions.

5.1 Costos

5.1.1 Costos del personal

Per determinar el cost del projecte, primer cal definir quin és el sou en relació amb l'hora de treball de cada especialista. Es fa important remarcar que l'autor d'aquest projecte serà l'encarregat de realitzar totes les tasques i com a conseqüència el preu/hora en tots els casos és mateix. Tanmateix, per fer una estimació realista dels costos del projecte, es defineixen el sou/hora per les especialitats de la Figura 5.1 .

Especialista	Sou anual	Preu/hora
Cap de projecte	39.210 [17]	21.5 €/h
hline Programador	29.734 [16]	16.3 €/h
hline Tester	32.129 [18]	17.6 €/h

Figura 5.1: Taula amb la relació especialista - preu/hora

Per calcular el preu hora, s'ha tingut en compte un total de 1.812hores/anuals per jornada completa [4].

Si observem en la Figura 5.2 podem observar quines tasques tenen assignades cadascun dels especialistes, i per tant el número d'hores total associades. Finalment, podem obtenir les estimacions parcials i totals dels costos de les tasques.

Especialista	Nº hores	Preu/Hora	Preu Total
Cap de projecte	205h	21.5 €/h	4.407,5 €
Programador	685h	16.3 €/h	11.165.5 €
Tester	70h	17.6 €/h	1.232 €
			Total: **16.805 €**

Figura 5.2: Taula amb estimacions parcials i total dels costos de les tasques

En la Figura 5.3 podem observar la relació entre totes les tasques definides en l'apartat anterior, les seves dependències, la duració en hores i l'assignació corresponent.

Tasca	Descripció	Dependència	Duració (h)	Assignació
T-11	Reunions		15	CP
T-12	Definir estructura i llenguatge de programació	T-11	5	CP
T-13	Definir i parametrizar amb dades	T-11	10	CP
T-14	Revisió Sprint	T-11,T-12,T-13	10	CP
T-21	Reunions		15	P, CP
T-22	Entendre l'entorn actual (amb tots els microserveis)	T-21	20	P
T-23	Entendre la connexió entre els diferents contenidors (docker)[50]	T-21,T-22	25	P
T-24	Entendre l'estructura de les bases de dades actuals i els valors de les taules corresponents.	T-21,T-22	15	P
T-25	Entendre el concepte de ORM (Object Oriented Mapping)[51] i estudiar l'accés d'aquests a les bases de dades.	T-21,T-22, T-24	25	P
T-26	Revisió Sprint.	T-21, T-22, T-23, T-24, T-25	5	P, CP
T-31	Reunions		15	P, CP
T-32	Creació d'un nou contenidor dins l'entorn actual.	T-31	25	P
T-33	Creació o verificació de la conneixió a les bases de dades i taules corresponents.	T-31	25	P
T-34	Definir patrons de disseny del projecte [52].	T-31, T-32, T-33	25	P
T-35	Revisió Sprint.	T-31, T-32, T-33, T-34	20	P, CP
T-41	Reunions		30	P, CP
T-42	Creació dels objectes i models necessàris pels accesos a les bases de dades	T-41	40	P
T-43	Representació de la disponibilitat de l'atleta en forma de matriu	T-41, T-42	25	P
T-44	Creació i connexió a la taula dels resultats dels entrenaments anteriors dels atletes	T-41	40	P
T-45	Creació de la funció objectiu a omptimitzar	T-41	40	P
T-46	Creació del sistema d'optimització amb l'algorisme més eficient.	T-41, T-45	165	P
T-47	Revisió Sprint.	T-41,T-42,T-43,T-44,T-45,T-46	40	P, CP
T-51	Reunions		15	P, CP
T-52	Migració del servei al servidor.	T-51	20	P
T-53	Instal·lació de dependències necessàries	T-51, T-52	10	P
T-54	Comprovar les connexions del microservei amb totes les parts	T-51,T-52,T-53	20	P
T-55	Revisió Sprint.	T-51, T-52, T-53, T-54	5	P, CP
T-61	Reunions		15	P, CP, T
T-62	Realització de proves amb casos extrems	T-61	30	T
T-63	Anàlisi i estudi dels resultats	T-61, T-62	20	T
T-64	Revisió Sprint.	T-61, T-62, T-63	5	P, CP, T
			Total:735h	

Figura 5.3: Taula de tots els Sprints amb les seves tasques corresponents, la duració en hores i l'assignació.

5.1.2 Costos del Hardware

Per desenvolupar el projecte s'utilitzarà un ordinador amb dues pantalles. L'especificació i costos d'aquests són els següents:

Producte	Quantitat	Preu Unitari	Preu Total	Vida útil	Amortització
Gigabyte GB-BRI5H-8250-BW [14]	1	427,36 €	427,36 €	6 anys	28,9 €
AOC Monitor G2460VQ6 24"	2	129,0 €	258,0 €	6 anys	17,44 €
			Total: 685,36 €		

Figura 5.4: Taula costos del Hardware

Per calcular l'amortització s'ha tingut en compte un total de 10.872 hores laborals a jornada completa durant 6 anys.

5.1.3 Costos del Software

Producte	Quantitat	Preu Unitari	Preu Total	Vida útil	Amortització
Visual Studio Code [14]	1	0 €	0 €	2 anys	0 €
Gitlab	1	0 €	0 €	2 anys	0 €
Overleaf	1	0 €	0 €	2 anys	0 €
Docker	1	0 €	0 €	2 anys	0 €
Python	1	0 €	0 €	2 anys	0 €
Slack	1	0 €	0 €	2 anys	0 €
			Total: 0 €		

Figura 5.5: Taula costos del Software

S'ha utilitzat exclusivament Software obert i gratuït, per tant els costos del Software són negligibles.

5.1.4 Costos indirectes

A continuació es descriuen els costs que afecten de forma indirecte al pressupost del projecte.

Producte	Quantitat	Preu Unitari	Preu Total	Vida útil	Amortització
Preu Coworking [6]	1	128 €/mes	768 €	6 mesos	768 €
			Total: 768 €		

Figura 5.6: Taula costos indirectes

L'empresa treballa en el coworking del Canòdrom de Barcelona. Paga un preu fix de 128 euros mensuals dels quals inclouen tots els costos indirectes.

5.2 Contingències

L'empresa ha especificat un valor específic d'un 4% pel càlcul de les contingències. Per tant, els costos tenint en compte les contingències han estat:

Tipus de cost	Valor (€)	Contingència	Valor amb contingència
Personal	16.805 € 1	4%	17.477,2 €
Hardware	685,36 € 1	4%	712,75 €
Software	0 €	4%	0 €
Indirectes	768 € 1	4%	798,72 €
			Total: 18.988,67 €

Figura 5.7: Taula costos amb contingències

5.3 Imprevistos

En aquest apartat s'afegirà en el pressupost el cost que poden tenir els possibles imprevistos que poden sorgir durant el desenvolupament del projecte. S'ha considerat que els imprevistos, tal com s'expliquen en l'apartat de l'abast, són negligibles donat que en els sprints sempre s'han de realitzar tasques de revisió. A més a més, hi ha alguna tasca per cobrir-los.

Pel que fa als imprevistos materials, pel material hardware s'ha tingut en compte un 1% de possibilitats que deixés de funcionar. Els costos associats són els següents:

Hardware	Valor (€)	Provabilitat	Valor amb l'imprevist
Gigabyte GB-BRI5H-8250-BW	427,36 € 1	1%	4,27 €
AOC Monitor G2460VQ6 24	258,0 € 1	1%	2,58 €
			Total: 6,85 €

Figura 5.8: Taula costos amb imprevistos

A continuació es detalla el pressupost final estimat pel projecte, detallat amb contingències i sense contingències.

	Sense contingències	Amb Contingències
Costos especialistes	16.805 €	17.477,2 €
Costos hardware	685,36 €	712,75 €
Costos software	0 €	0 €
Costos indirectes	768 €	798,72 €
Total	18.258,36 €	18.988,67 €
Imprevistos	6,85 €	
Total amb imprevistos	18.265,21 €	18.995,52 €

Figura 5.9: Taula costos finals

5.4 Pressupost final

A continuació es detalla el pressupost final estimat pel projecte, detallat amb contingències i sense contingències.

	Sense contingències	Amb Contingències
Costos especialistes	16.805 €	17.477,2 €
Costos hardware	685,36 €	712,75 €
Costos software	0 €	0 €
Costos indirectes	768 €	798,72 €
Total	18.258,36 €	18.988,67 €
Imprevistos	6,85 €	
Total amb imprevistos	18.265,21 €	18.995,52 €

Figura 5.10: Taula costos finals

En total el pressupost calculat puja a un total de 18.265,21 € sense tenir en compte les contingències, i a un total de 18.995,52 € si es tenen en compte les contingències.

5.5 Gestió de costos

Per mantenir el control del pressupost en tot moment, cada cop que es tanqui una tasca del corresponent sprint s'actualitzarà el pressupost en base les hores reals consumides i en cas d'imprevistos, també els costos associats a aquests.

Els valors es compararan amb els valors teòrics prevists per tal d'identificar les possibles desviacions per cada tasca i sprint. Un cop s'hagi finalitzat el projecte, s'obtindran les desviacions totals. A continuació s'ha desglossat els indicadors utilitzats.

- *Desviació dels costos humans = (Cost estimat - cost real) * Hores reals*
- *Desviació d'hores consumides per tasca = (Hores estimades - Hores reals) * cost real*
- *Desviació dels costos hardware = Cost estimat hardware - Cost real hardware*
- *Desviació dels costos software = Cost estimat software - Cost real software*
- *Desviació dels costos indirectes = Cost estimat indirectes - Cost real indirectes*
- *Desviació dels costos imprevistos = Cost estimat imprevistos - Cost real imprevistos*
- *Desviació dels costos en recursos humans = Cost estimat en recursos humans - Cost real en recursos humans*
- *Desviació total d'hores = Hores estimades totals - Hores reals totals*
- *Desviació total dels costos = Cost estimat total - Cost real total*

Disseny

Aquest apartat de la memòria pretén donar a conèixer l'estructura del sistema, és a dir, quines estructures o patrons de disseny s'ha utilitat, com es comunica el sistema amb les altres parts, quina base de dades s'ha utilitzat i quina és la seva estructura. Per últim, s'explicarà quin framework [48] s'ha utilitzat per a la interfície i com funciona.

6.1 Arquitectura del sistema

El projecte està desenvolupat en la línia de ser un microservei aïllat, amb tots els avantatges i inconvenients que comporta. S'ha utilitzat Docker per dissenyar i crear l'estructura del sistema en forma de contenidors.

6.1.1 Microservei

El projecte, tal com s'ha explicat en els objectius i requisits, s'ha desenvolupat en forma de microservei, utilitzant Docker. Aquesta estructura s'adapta molt al projecte, principalment pel motiu que el sistema s'ha d'integrar en la plataforma actual de Trainerer, i a més, també presenta molts avantatges com s'ha definit a continuació.

6.1.1.1 Integració amb el sistema actual de Trainerer

Actualment Trainerer ja disposa de la seva pròpia plataforma en funcionament, i per tant, aquest projecte necessita integrar-s'hi. L'estructura de microservei permetrà aïllar el projecte amb totes les seves dependències. Això suposarà que només caldrà establir un sistema i protocol de comunicació entre el projecte i la plataforma actual de l'empresa. Aquest sistema de comunicació serà RabbitMQ, explicat en detall als següents apartats. D'aquesta manera el projecte aportarà unes noves funcionalitats a la plataforma actual de l'empresa.

6.1.1.2 Escalable

Un altre dels avantatges, és l'escalabilitat que dóna un microservei. Actualment, Trainerer es troba en els seus inicis i el nombre d'usuaris no és molt gran. Tot i això, si es segueix la planificació establerta dins l'empresa i el nombre d'usuaris segueix creixent, el sistema ha d'estar pensat per suportar aquest increment d'usuaris. L'estructura de microservei s'enfoca també en aquesta direcció de ser escalable.

Com que el sistema queda encapsulat amb totes les dependències, en cas que es necessiti sempre es pot tenir més d'una instància d'aquest. És a dir, si a l'hora de generar temporades i planificacions esportives un microservei no és suficient, en podem tenir dos realitzant les mateixes tasques.

6.1.1.3 Fiabilitat

La fiabilitat també és un avantatge dels microserveis. Tal com s'ha explicat anteriorment, el projecte serà un sistema aïllat de la plataforma actual. D'aquesta manera, en cas que la plataforma de Trainerer deixes de funcionar per algun motiu, el projecte no es veuria afectat.

6.1.1.4 Millora i actualitzacions

Per últim però no menys important, tenim el factor de mantenir el sistema actualitzat amb les últimes tecnologies. Això significa que si apareix alguna solució nova al mercat que augmenta el rendiment de la generació de planificacions esportives d'aquest projecte, podem actualitzar-ho de manera molt senzilla. Amb la creació d'un nou microservei que realitzi les mateixes funcions i mantingui els mateixos canals de comunicació, però utilitzant noves tècniques més eficients, podem reemplaçar el microservei antic pel nou i mantenir el conjunt del sistema funcionant.

6.1.2 Docker

Per la creació i encapsulació del projecte s'utilitzarà Docker. El Docker [50] és un projecte de codi obert que permet que una aplicació Linux [26] i les seves dependències s'empaquetin en un contenidor. D'aquesta manera cada aplicació que es trobi en un contenidor queda aïllada de la resta.

Una de les funcionalitats del Docker és el *docker-compose*, que tal com explica la documentació oficial [8], s'utilitza per al desplegament i posada en marxa de múltiples contenidors al mateix temps.

Pel projecte s'ha utilitzat 3 contenidors:

- MariaDB: base de dades del projecte. Conté tota la informació necessària per al funcionament d'aquest.
- Projecte (python): contenidor del projecte, preparat per desenvolupar tot el sistema en Python.
- RabbitMQ: s'utilitza com a sistema de comunicació entre el projecte i la plataforma externa de Trainerer. S'explica en profunditat en els següents apartats.

6.1.2.1 Base de dades: mariaDB

El sistema funciona amb la base de dades relacional SQL mariaDB [13]. S'ha decidit tenir la base de dades aïllada en un contenidor separat per protegir les dades. En cas d'haver-hi algun problema amb el projecte o que en algun moment perillés la seva viabilitat, les dades quedarien intactes.

També per la part de seguretat, es fa més fàcil mantenir un sistema de còpies de seguretat.

6.1.2.2 Projecte: Python

En aquest contenidor si desenvoluparà el sistema d'optimització amb restriccions. Conté Python3.7 instal·lat i totes les seves dependències necessàries per poder desenvolupar el projecte.

6.1.2.3 RabbitMq

RabbitMQ [37] és un projecte de codi obert que es coneix en anglès com *message broker*. Actua com a *middleware* entre el consumidor i productor. En el cas d'aquest projecte s'utilitzarà com a consumidor, i la plataforma actual de Trainerer actuarà com a productor. D'aquesta manera quan es necessiti la creació d'alguna temporada o planificació setmanal per algun atleta, el productor (plataforma de Trainerer) emetrà una petició al consumidor (projecte actual). Aquest últim un cop hagi processat i creat la temporada o planificació, enviarà la resposta pel mateix canal.

És sense dubte una peça clau dins del projecte, ja que permet la connexió del sistema amb l'exterior. És el canal a través del qual es demanaran temporades o planificacions setmanals d'entrenament. RabbitMQ suporta molts protocols de connexió, però en el cas d'aquest projecte s'utilitzarà com *message broquer* per connexions asíncrones mitjançant cues. Si en algun moment es realitzessin moltes peticions simultànies, aquestes quedarien empilades a la cua del RabbitMQ i serien tractades amb seqüencialitat. Per aquest motiu la connexió serà asíncrona.

6.1.3 Diagrama dels contenidors

En la Figura 6.2 es pot observar l'arquitectura dels contenidors del sistema i també les connexions amb la plataforma actual de Trainerer mitjançant RabbitMQ.

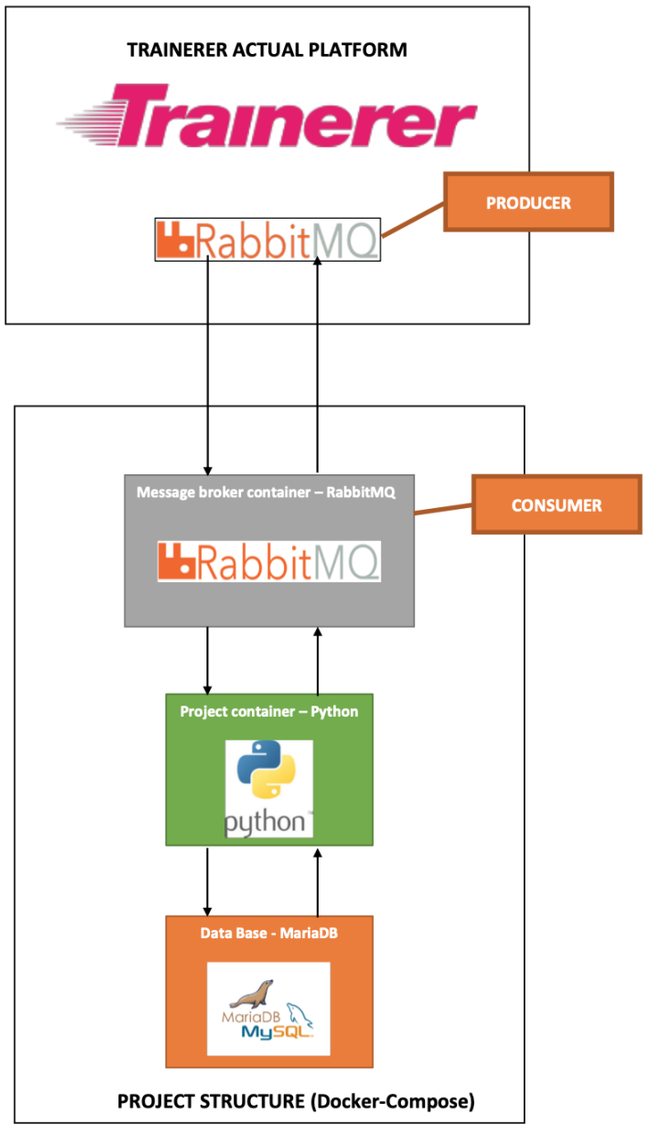


Figura 6.1: Disseny dels contenidors del sistema i connexió amb la plataforma Trainerer

El contenidor Python, on està desenvolupat el microservei d'aquest projecte té connexió amb el contenidor de la base de dades MariaDB. En aquesta base de dades s'hi emmagatzemarà tota la informació necessària per al funcionament d'aquest projecte, com per exemple temporades esportives predefinides, informació relativa als esports o als atletes, planificacions setmanals predefinides....

6.2 Arquitectura de la base de dades

Tal com s'ha explicat en la secció 6.1.2.1 del contenidor de la base de dades, s'utilitza una base de dades relacional SQL amb tots els avantatges que suposa.

6.2.1 Taules

Una base de dades relacional s'organitza en taules i les relacions entre aquestes. En els següents apartats es descriuen les taules del sistema i es defineixen el conjunt de relacions.

6.2.1.1 Taula Athletes

Conté tota la informació dels atletes usuaris del sistema. Aquesta informació consta de les seves dades físiques necessàries per realitzar les planificacions esportives setmanals i el disseny de les temporades.

6.2.1.2 Taula Events

Conjunt de sessions esportives d'entrenament predefinides de tots els esports. Conté informació com la descripció de la sessió, esport, duració i nivell.

6.2.1.3 Taula Events_translation

Conte la traducció a diferents idiomes de les sessions d'entrenament de la taula *events* explicada anteriorment. D'aquesta manera quan s'introdueix una sessió nova al sistema ja s'introdueix la traducció a tots els idiomes.

6.2.1.4 Taula sports_id_generation

Taula amb tots els esports del sistema. Conté el nom de l'esport i un *id* que l'identifica.

6.2.1.5 Taula plannings_generation

Taula per les planificacions setmanals predefinides.

6.2.1.6 Taula season

Taula per les temporades predefinides al sistema. Conté informació de l'esport, i duració mínima i màxima en setmanes.

6.2.1.7 Taula training_period

En català període d'entrenament, subconjunt de setmanes dins d'una temporada. Conté informació de la prioritat dins d'una temporada, el nombre de setmanes de duració i si és prescindible o no.

6.2.1.8 Taula training_period_type

Categoritza tots els tipus de període d'entrenament. S'ha utilitzat una taula a part amb la intenció de poder identificar els tipus de períodes d'entrenament amb més facilitat.

6.2.1.9 Taula season_athlete

Taula que emmagatzema la relació entre un atleta i una temporada. S'utilitza en el sistema per emmagatzemar temporades ja creades i per tenir constància sobre quin punt d'aquesta passa un atleta a l'hora de generar-l'hi una planificació setmanal.

6.2.2 Relació entre taules

En les bases de dades relacionals SQL existeixen principalment tres tipus de relacions:

- Relació un a un: Quan una instància d'una taula li correspon una instància d'una altra.
- Relació un a molts o molts a un: Quan una instància d'una taula li corresponen més d'una instància d'una altra taula o viceversa.
- Relació molts a molts: Quan instàncies d'una taula li corresponen més d'una instància d'una altra, i les instàncies d'aquesta segona taula li corresponen més d'una instància de la primera.

6.2.2.1 Relació events - events_translation

És una relació un a molts, ja que per cada instància d'*events* (és a dir per cada sessió d'entrenament) li correspon més d'una instància d'*events_translation*, és a dir, més d'una traducció. D'aquesta manera una mateixa sessió d'entrenament està traduïda a més d'un idioma.

6.2.2.2 Relació events - sport_id_generation

Es correspon a una relació d'un a molts. Un mateix esport està relacionat amb moltes sessions d'entrenament, mentre que una sessió d'entrenament només és d'un sol esport.

6.2.2.3 Relació plannings_generation - events

Per tal com s'ha definit i treballat es correspon molts a molts, ja que una sessió d'entrenament pot pertànyer a diferents planificacions setmanals predefinides i per l'altra banda, com és evident, una planificació setmanal conté més d'una sessió d'entrenament.

6.2.2.4 Relació plannings_generation - sports_id_generation

Es correspon a una relació un a molts. Una planificació setmanal està destinada a preparar-se per un esport i un esport està relacionat amb més d'una planificació setmanal.

6.2.2.5 Relació plannings_generation - training_period_type

Es correspon a una relació un a molts. Una planificació setmanal es troba dins d'un tipus període d'entrenament, mentre que un període d'entrenament està relacionat amb més d'una planificació setmanal.

6.2.2.6 Relació training_period - season

Es correspon a una relació molts a molts. Una temporada esportiva té més d'un període d'entrenament associat i un període d'entrenament també pot pertànyer a més d'una temporada diferent. Per exemple, un atleta que es vol preparar per a una marató, i un altre per una mitja marató, segurament tots dos dins la seva temporada tindran un període d'entrenament per treballar la resistència.

6.2.2.7 Relació season_athlete - season

Es correspon a una relació un a un. La taula season_atlete s'utilitza per relacionar una temporada personalitzada amb un atleta i informació addicional com per exemple data d'inici i data fi. Com és lògic cada temporada personalitzada té una sola instància de season_athlete, donat que és personalitzada. Per l'altre costat, una temporada personalitzada només s'assigna a un sol atleta i per tant, una instància de season_athlete només li correspon una instància de temporades.

6.2.2.8 Relació season_athlete - athlete

Aquesta és l'altra relació explicada anteriorment, però a diferència de l'altra aquesta és d'un a molts. Per una banda una instància de season_athlete només li correspon a un atleta, pel motiu que una temporada és exclusiva d'un atleta. En canvi, pot ser que l'atleta un cop hagi acabat una temporada en comenci una altra i com és lògic una instància d'atleta pot està relacionada amb més d'una instància de la classe season_athlete.

6.2.3 Diagrama

La Figura 6.2 mostra les taules i relacions explicades en els apartats anteriors.

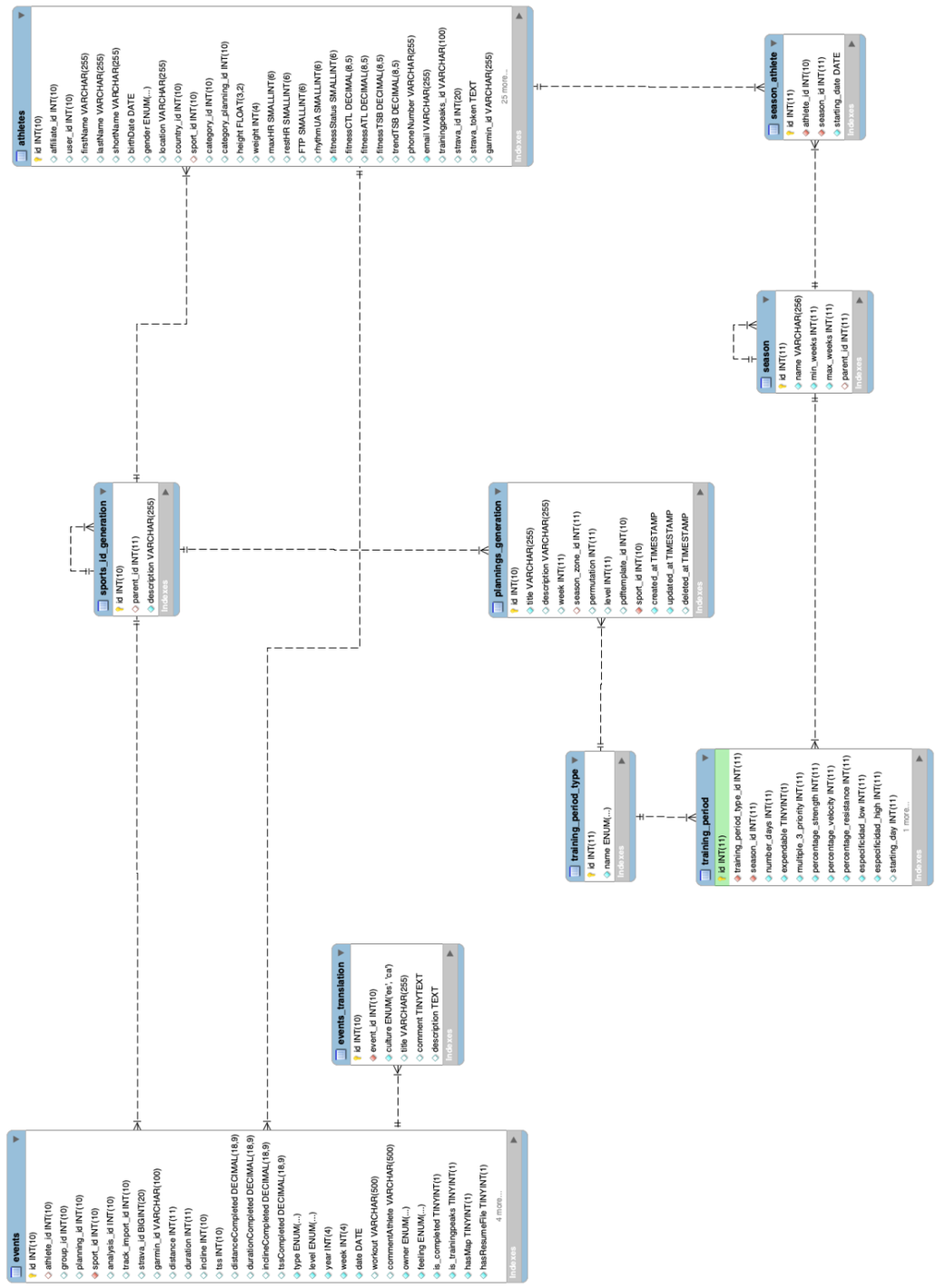


Figura 6.2: Diagrama arquitectura base de dades MariaDB

6.3 Mapatge d'objectes relacional

El mapatge d'objectes relacional, en anglès object-relational mapping [32], és una tècnica de programació molt utilitzada en la programació orientada objectes per convertir informació en objectes. En aquesta tècnica, a l'hora de tractar grans quantitats d'informació, el modelatge, manipulació i creació d'aquesta es realitza d'una manera més senzilla i eficaç.

En aquest sistema, tal com s'ha explicat anteriorment, es treballa amb una base de dades relacional SQL. Per exemple, si volgués obtenir la informació de l'atleta (taula *athletes*) amb id 10, hauria de realitzar la següent consulta SQL:

```
"SELECT id,
first_name,
last_name,
phone,
birth_date,
sex,
age FROM persons WHERE id = 10"
```

En canvi, si utilitzés un framework per realitzar el mapatge d'objectes, un cop s'ha definit l'objecte atleta, la consulta seria:

```
athlete = repository.get_athlete_by_id(id=10)
print(athlete.first_name)
print(athlete.last_name)
print(athlete.phone)
print(athlete.birth_date)
print(athlete.sex)
print(athlete.age)
```

Remarcar que el projecte està programat amb Python3.7 i s'ha utilitzat SQLAlchemy pel mapatge d'objectes a la base de dades.

6.3.1 SQLAlchemy

SQLAchemy [45] és el framework de Python que s'ha utilitzat per realitzar el mapatge d'objectes a la base de dades. És un framework de codi obert, i dins de l'ecosistema de Python, és el que compta amb una comunitat i documentació més extensa. Aquests dos punts junts amb el fet que organitzacions com Reddit [41] o Dropbox [10] l'utilitzin en l'actualitat, ha fet que es decidís escollir SQLAlchemy envers alguna altra opció.

Tal com s'ha explicat, SQLAlchemy converteix informació de la base de dades en objectes de Python. La nomenclatura habitual és definir un objecte per cada taula de la base de dades que es necessiti. Per l'altra banda, en Python els objectes es defineixen en classes tal com s'explica en la documentació oficial del llenguatge [38].

A continuació es mostra un exemple (Figura 6.3) del mapatge d'objectes utilitzant SQLAlchemy, tal com es pot observar en els comentaris consta de 3 parts:

- Part 1: consta de la definició dels camps de la base de dades relacional. S'indica el nom i el tipus.

- Part 2: es defineix un creador d'objectes de Python amb els paràmetres de la Part 1.
- Part 3: es defineix els *getters* i *setters* per obtenir o modificar informació dels objectes.

```
class Athlete(model.Base):
    """Athlete class, corresponds to athletes table on db"""

    # PART 1 -> mariadb database fields
    __tablename__ = 'athletes'
    id            = model.Column(model.Integer, primary_key = True)
    gender        = model.Column('gender', model.Enum(*enum_gender.
                                 enum_gender))
    height        = model.Column('height', model.Integer)
    weight        = model.Column('weight', model.Integer)
    birthDate     = model.Column('birthDate', model.Date)
    restHR        = model.Column('restHR', model.Integer)
    sport_id      = model.Column(model.Integer, model.ForeignKey('
                                 sports_id_generation.id'))
    sport         = model.relationship('Sport', backref='athletes')

    # PART 2 -> python object creation
    def __init__(
        self,
        gender    : str = None,
        height    : str = None,
        weight    : str = None,
        birthDate: model.datetime = None,
        restHR    : int = None,
        sport     : modelSport.Sport =  None
    ):
        self.gender    = gender
        self.height    = height
        self.weight    = weight
        self.birthDate = birthDate
        self.restHR    = restHR
        self.sport     = sport

    # PART 3 -> getters and setters
    def get_gender(self):
        return self.gender

    def set_gender(self, new_gender):
        self.gender = new_gender

    def get_height(self):
        return self.height

    def set_height(self, new_height):
        self.height = new_height

    ...
```

Figura 6.3: Exemple de mapeig d'objectes utilitzant SQLAlchemy en Python3.7

6.4 Patrons de disseny

Com ve ja s'ha dit, aquest projecte no pertany a l'especialitat de Software sinó a la de Computació de la Universitat Politècnica de Catalunya. Tanmateix, a l'hora de dissenyar i implementar un sistema d'optimització amb restriccions en forma de microservei es fa indispensable el disseny d'una arquitectura com a producte Software per l'empresa. En aquest apartat es recullen els patrons de disseny utilitzat i la funció que realitzen dins del sistema.

6.4.1 Patró repositori

Un dels patrons més importants del projecte i que ha ajudat a definir l'estructura d'aquest és el repositary. El patró repositori [2] s'utilitza per aïllar tota la part de gestió de la informació dins del projecte. En el cas d'aquest projecte s'ha utilitzat per separar tota la part de consultes a base de dades. Es fa evident, que s'ha utilitzat en conjunt amb el framework SQLAlchemy per la creació d'objectes a partir de les dades.

Amb aquest patró, el projecte queda dividit en 3 parts:

- Models: objectes de Python que el projecte utilitza. Molts d'aquests tenen la seva instància la a la base de dades mitjançant SQLAlchemy, d'altres simplement són objectes utilitzats en l'execució del servei.
- Repositori: totes les funcions de consulta i guardat d'informació a la base de dades.
- Serveis: s'utilitza per gestionar totes les funcionalitats del sistema. Per exemple, un dels serveis del sistema estar enfocat a gestionar la creació de temporades esportives personalitzades als atletes.

Remarcar que la majoria dels serveis són les funcionalitats principals del projecte, és a dir, hi ha un servei per generar planificacions setmanals personalitzades, un altre per generar temporades...

6.4.2 Patró hereditari

Com el seu nom indica és un patró en què objectes poden heretar informació d'altres objectes. Com es pot veure en la documentació final del projecte s'ha utilitzat aquest patró en diferents parts. Per exemple, a l'hora de crear un nou objecte d'una temporada personalitzada per un atleta, l'objecte final de temporada hereta molta informació sobre l'objecte de temporada predefinida.

6.4.3 Patró middleware

Un patró middleware [27] és un patró de disseny que s'utilitza per a la concatenació d'execució de processos. El número i el tipus de procés es decideix en la inicialització del middleware.

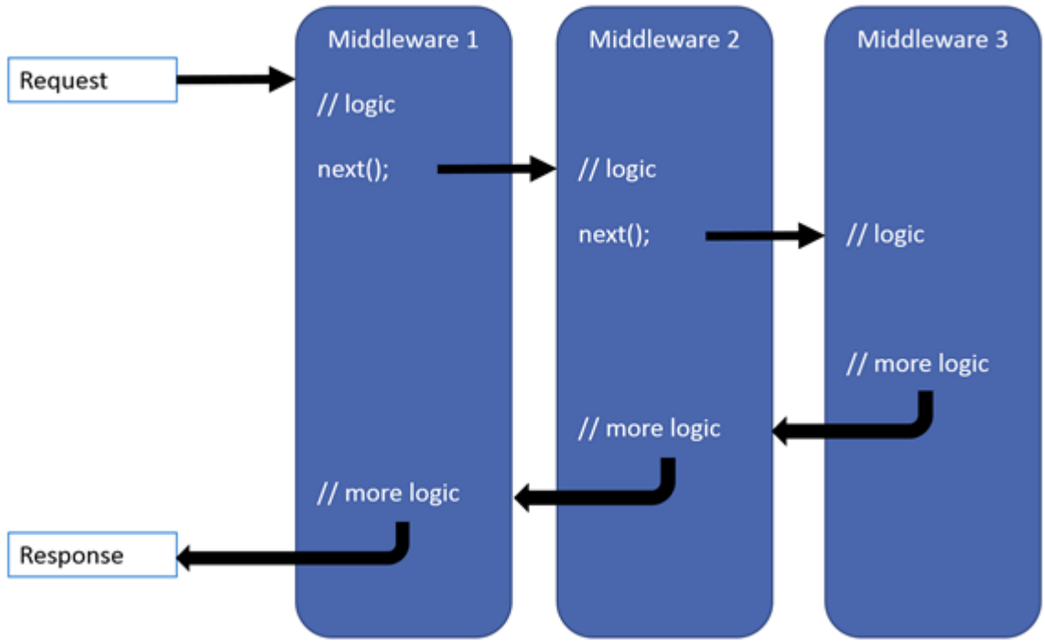


Figura 6.4: Representació patró middleware, documentació Microsoft [34]

Aquest patró s'ha utilitzat en la funció objectiu del sistema d'optimització amb restriccions. Queda explicat i documentat al detall en els apartats següents.

6.4.4 Patró decorador

El patró decorador s'utilitza quan volem afegir una funcionalitat a un objecte i per fer-ho no volem crear un nou objecte amb caràcter hereditari. En aquest projecte s'ha utilitzat en diferents punts. Com a exemple, durant el desenvolupament del projecte es va crear una funció per testejar l'eficiència dels objectes i serveis en temps reals d'execució. El patró decorador en aquest cas s'utilitzava per afegir als objectes i serveis la funció de comprovar l'eficiència.

6.5 Interfície

La interfície és l'última secció dins l'arquitectura del projecte. És una part necessària però no per això la més rellevant d'aquest projecte. Si retrocedim una mica, cal recordar que el projecte té l'estructura de microservei. Aquesta estructura fa que el sistema estigui sempre pendent de possibles peticions, per donar-hi resposta en forma de temporada esportiva i/o planificació setmanal predefinida.

Per facilitar les tasques de manteniment i seguiment a l'empresa Trainerer, s'-ha desenvolupat una interfície per monotonitzar les funcions del microservei. La interfície tindrà les funcions següents:

- Seguiment en directe del microservei. Es podrà saber en tot moment la càrrega de treball del microservei.
- Històric de peticions i respostes. Es podrà fer un seguiment de totes les peticions rebudes i la resposta donada pel sistema.

- Tests del sistema. Es podrà realitzar un test complet al microservei per saber si té tots els serveis operatius.
- Tests de serveis. Es podrà realitzar proves i analitzar els resultats obtinguts. És important com a pas previ a producció.

La interfície és per ús intern de l'empresa Trainerer, ja que com s'ha comentat s'utilitzarà per al seguiment i monitoratge del microservei. Aquest fet és important, ja que es prioritzarà la senzillesa i utilitat, respecte al disseny. Amb aquesta finalitat, s'han contemplat dos possibles frameworks de Python: Flasy i Django. Tots dos compten amb una gran comunitat i una gran trajectòria. A continuació s'expliquen les dues opcions i es justifica la decisió d'escollir Flask per davant de Django.

6.5.1 Django

Django [7], tal com es defineixen en la seva documentació, és un framework Web Python d'alt nivell de codi obert. Està pensat per treballar en estructura de Model-Vista-Controlador i encarat principalment pel desenvolupament web escalable i d'alta qualitat. Utilitza el seu propi mapatge d'objectes relacional, i inicialment conta ja amb una interfície per la configuració i gestió de bases de dades.

6.5.2 Flask

Flask [12] és un framework web Python pel disseny web però més orientat al disseny d'apis. No és tan sofisticat com Django, que pretenia la qualitat i l'escalabilitat, sinó que té un disseny més minimalista i lleuger. Està agrupat amb paquets, de manera que el desenvolupador només instal·larà les funcions que necessita. Segurament gràcies a la senzillesa és més ràpid en comparativa amb Django. No disposa d'un mapatge d'objectes relacional propi, sinó que utilitza SQLAlchemy i no té un sistema d'autentificació tan sofisticat com sí que té Django.

6.5.3 Flask per davant de Django

Un cop s'ha analitzat els punts forts i els punts febles de Django i Flask, s'ha decidit finalment utilitzar Flask. Els arguments per prendre la decisió han estat els següents:

- La interfície web no és la part principal del projecte i no està dirigida als usuaris. Per tant, s'ha valorat més la velocitat, senzillesa i lleugeresa del framework en comptes de l'escalabilitat i qualitat Web de Django.
- Flask utilitza SQLAlchemy com a mapatge d'objectes relacional. Donat que SQLAlchemy ja s'utilitza en altres punts del projecte, la integració és més senzilla.
- Només es necessita autentificació per part de l'equip desenvolupador de Trainerer, per tant no es necessita un servei d'autentificació molt sofisticat.
- La senzillesa de Flask fa que s'hagi pogut integrar la interfície al microservei en hores, a diferència de Django que hagués estat molt més costós.

En l'apèndix queda documentat amb captures de pantalla totes les parts de la interfície web.

Especificació

El projecte està organitzat en repositoris, models i serveis. Els models per definir els objectes o classes de Python del projecte, els repositoris per obtenir la informació de la base de dades en forma de mapatge d'objecte Python, i per últim els serveis que realitzen les funcionalitats del projecte. Afegir que els serveis de generació de temporades i planificacions setmanals personalitzades només se'n donarà una petita definició donat que en els pròxims capítols estan desglossats al detall.

7.1 Descripció de les classes

En Python, com molts dels llenguatges de programació orientats a objectes, els objectes s'utilitzen per representar i manipular estructures de dades més complexes. En el cas concret de Python, els objectes es defineixen a partir de les classes, de manera que una classe en defineix l'estructura i tipus de paràmetres, i l'objecte és una instància amb valors de la classe definida.

```
class Dog:

# Initializer / Instance Attributes
def __init__(self, name, age):
    self.name = name
    self.age = age


# Instantiate the Dog object
philo = Dog("Philo", 5)
mikey = Dog("Mikey", 6)

# Access the instance attributes
print("{} is {} and {} is {}.".format(
    philo.name, philo.age, mikey.name, mikey.age))

# Is Philo a mammal?
if philo.species == "mammal":
    print("{0} is a {1}!".format(philo.name, philo.species))
```

Figura 7.1: Exemple d'instanciació d'objectes de la comumentació de Python [40]

En executar l'exemple de la Figura 7.1 el resultat és el següent:

```
Philo is 5 and Mikey is 6.
Philo is a mammal!
```

Figura 7.2: Resultat d'execució de la Figura 7.1

En el projecte s'ha utilitzat el concepte de *col·lecció*, per referir-se al conjunt de més d'un objecte de la mateixa classe. Per exemple, un conjunt d'objectes d'atletes formaria un Athlete Collection. El concepte de *Collection* és molt interessant sobretot quan es realitza un mapatge d'objectes relacional a base de dades, com és el cas d'aquest projecte. D'aquesta manera quan a través del repositori es realitzi una consulta a base de dades on el resultat sigui més d'una instància, s'estarà realitzant un mapatge en el sistema en forma de col·lecció d'objectes.

En aquest projecte a l'hora de crear un *collection* s'ha heretat les propietats del Collection de SQLAlchemy. Recordem que SQLAlchemy s'utilitza per al mapatge d'objectes a la base de dades. Per temes d'eficiència a l'hora de fer consultes, per crear una col·lecció, totes les classes que representen col·leccions hereten de la classe *collection* de SQLAlchemy. Bàsicament un *collection* de SQLAlchemy és semblant a una llista de Python. Tal com podem veure a la Figura 7.3, quan es defineix la classe SessionCollection s'hereta les propietats de la classe *collections.InstrumentedList* de SQLAlchemy.

```
import sqlalchemy.orm.collections as collections

class SessionCollection(collections.InstrumentedList):
    """Session Collection class. Represents a group of sessions"""
    def __init__(
            self,
            listSession: list = []
        ):
        super(SessionCollection, self).__init__(listSession)


    def add_Session(self, session: modelSession.Session) -> None:
        """Adds session to the session collection"""
        self.append(session)

    def rm_Session(self, session: modelSession.Session) -> None:
        """Removes session from session collection"""
        self.remove(session)
```

Figura 7.3: Representació d'un collection de sessions. Fitxer `modelSessionCollection.py`

7.1.1 Classe Athlete

Una instància de la classe Athlete representa un atleta dins del sistema. La classe està definida en el fitxer `modelAthlete.py` i representada a base de dades per la taula *athletes*.

La classe athlete té els paràmetres següents:

- id: Enter que identifica l'atleta en el sistema.
- gender: Indica el sexe de l'atleta. Pot perdre dos possibles valors: 'MALE', 'FEMALE'.
- height: Alçada de l'atleta.
- weight: Pes de l'atleta.
- birthDate: Dia de naixement de l'atleta. S'utilitza entre d'altres per calcular l'edat de l'atleta.
- restHR: Freqüència cardíaca en repòs de l'atleta.

- sport_id: Id de l'esport principal que entrena l'atleta.

7.1.2 Classe Session

Una instància de la classe Session representa una sessió d'entrenament dins del sistema. La classe està definida en el fitxer `modelSession.py` i representada a la base de dades per la taula *events.*

La classe session té els paràmetres següents:

- id: Enter que identifica la sessió en el sistema.
- sport_id: Id de l'esport de la sessió d'entrenament.
- planning_id: Id de la planificació setmanal predefinida a la qual pertany la sessió d'entrenament.
- duration: Duració en minuts de la sessió d'entrenament.
- distance: Distancia en metres que és recórrer en la sessió d'entrenament. En cas que no sigui rellevant serà *NULL*, com per exemple en les sessions d'entrenament de força en un gimnàs.
- incline: Inclinació de la sessió d'entrenament. Donada en metres, respecte a 1000 metres. En cas que la sessió d'entrenament no necessiti aquest paràmetre, es deixarà a *NULL.*
- session_type: Tipus de sessió d'entrenament. El tipus pot ser: 'STANDARD', 'OPTIONAL', 'IMPORTANT', 'FULLDAY', 'COMPETITION'. En aquest projecte s'utilitza 'IMPORTANT' (sessió de qualitat) i 'STANDARD' (sessió complementària). Els altres tipus tenen funcions especials dins la plataforma de Trainerer, com per exemple marcar dies de cursa.
- workout: Descripció de l'entrenament en format text.
- date: Dia de la setmana, de dilluns a diumenge, que està programat la sessió en la planificació setmanal predefinida.
- translation: Traducció del *workout* a diferents idiomes.

7.1.3 Classe Session Collection

Una instància de la classe Session Collection representa un conjunt de sessions d'entrenament dins del sistema. Tenir un objecte per representar col·leccions de sessions d'entrenament és important pel fet que una planificació setmanal ja és en si un conjunt de sessions, a més a més d'algun altre paràmetre.

7.1.4 Classe Session Translation

Una instància de la classe Session Translation representa una traducció d'una sessió d'entrenament dins del sistema. La classe està definida en el fitxer `modelSessionTranslation.py` i representada a la base de dades per la taula *events_ translation.*

La classe session té els paràmetres següents:

- id: Enter que identifica una traducció en el sistema.
- event_id: Sessió d'entrenament per la qual s'ofereix una traducció.
- culture: Cultura per la qual la traducció està definida. Una cultura seria 'ENG', 'ESP' o 'CAT'.
- title: títol de la sessió d'entrenament en la cultura corresponent.
- description: descripció de la sessió d'entrenament en la cultura corresponent.

7.1.5 Classe Plannings

Una instància de la classe Plannings representa una planificació setmanal predefinida dins del sistema. La classe està definida en el fitxer `modelPlanning.py` i representada a la base de dades per la taula *plannings_generation*.

La classe plannings té els paràmetres següents:

- id: Enter que identifica la planificació en el sistema.
- title: Títol de la planificació setmanal.
- description: Descripció de la planificació setmanal.
- week: Setmana dins del conjunt de setmanes del període d'entrenament al qual pertany.
- season_zone_id: Id del període d'entrenament al qual pertany la planificació.
- permutation: Quina permutació de la planificació predefinida pertany.
- level: Nivell de la planificació setmanal.
- sport_id: Id de l'esport al qual està encarat la planificació temporal.
- created_at: Data de creació de la planificació predefinida.
- plannings_sessions: Referència un *Session Collection*, és a dir, a un conjunt de sessions d'entrenament.

7.1.6 Classe Plannings Collection

Una instància de la classe Plannings Collection representa un conjunt de planificacions setmanals predefinides dins del sistema. Tenir un objecte per representar col·leccions de planificacions és rellevant pel fet que en la majoria dels casos es consulta a base de dades planificacions d'un tipus en concret, i el resultat sol ser més d'una planificació. Tenir el *collection* facilita la lectura de les dades.

7.1.7 Classe Sport

Una instància de la classe Sport representa un esport dins del sistema. La classe està definida en el fitxer `modelSport.py` i representada a la base de dades per la taula *sport_id_generation*.

La classe sport té els paràmetres següents:

- id: Enter que identifica l'esport en el sistema.
- parent_id: Enter que identifica si l'esport decideix d'algun altre esport. Per exemple, bicicleta de carretera o bicicleta de muntanya decideix del ciclisme.
- description: Nom de l'esport en format text.

7.1.8 Classe Sport Collection

Una instància de la classe Sport Collection representa un conjunt d'esports del sistema.

7.1.9 Classe Season

Una instància de la classe Season representa una temporada dins del sistema. La classe està definida en el fitxer `modelSeason.py` i representada a la base de dades per la taula *season*.

La classe season té els paràmetres següents:

- id: Enter que identifica la temporada en el sistema.
- name: Nom de la temporada o objectiu esportiu.
- min_weeks: Nombre mínim de setmanes pels quals la temporada es pot adaptar.
- max_weeks: Nombre màxim de setmanes per realitzar la temporada.
- parent_id: Id de la temporada de la qual descendeix, en cas que existeixi.
- training_periods: Instància de la classe Training Period Collection. Col·lecció de períodes d'entrenament que formen una temporada.

7.1.10 Classe Season Collection

Una instància de la classe Season Collection representa un conjunt de temporades del sistema.

7.1.11 Classe Training Period

Una instància de la classe Training Period representa un període d'entrenament dins del sistema. La classe està definida en el fitxer `modelTrainingPeriod.py` i representada a la base de dades per la taula *training_period*.

La classe Training Period té els paràmetres següents:

- id: Enter que identifica el període d'entrenament dins del sistema.
- training_period_type_id: Enter que identifica una instància de tipus de període d'entrenament.

- season_id: Enter que identifica la temporada de la qual el període d'entrenament forma part.
- weeks: Duració en nombre de setmanes del període d'entrenament.
- expendable: Binary que indica si el període d'entrenament és prescindible o no en la temporada a la qual pertany.
- multiple_3_priority: Importància del període d'entrenament dins la temporada. Un període d'entrenament està definit com a múltiple de 4 setmanes. Aquest paràmetre indica la prioritat dins la temporada a reduir-se en duració a múltiple de 3 setmanes.
- percentage_strength: Percentatge temporal de força que es treballa respecte al 100%.
- percentage_velocity: Percentatge temporal de velocitat que es treballa respecte al 100%.
- percentage_resistencia: Percentatge temporal de resistència que es treballa respecte al 100%.
- especificidad_low: Nivell d'intensitat mínima a la qual es treballarà.
- especificidad_high: Nivell d'intensitat màxima a la qual es treballarà.
- starting_day: Data que indica el moment que es comença el període d'entrenament. Si no ha sigut assignat a cap atleta, romandrà *NULL*.
- color: Color del període d'entrenament en format RGB. Fins a purament d'interfície.

Remarcar que en el projecte no s'utilitzen els paràmetres de percentage_strength, percentage_velocity, percentage_resistance, especificidad_low i especificidad_high. Són utilitzats per la plataforma Trainerer.

7.1.12 Classe Training Period Collection

Una instància de la classe Training Period Collection representa un conjunt de períodes d'entrenament del sistema.

7.1.13 Classe Training Period Type

Una instància de la classe Training Period Type representa un tipus de període d'entrenament dins del sistema. La classe està definida en el fitxer `modelTrainingPeriodType.py` i representada a la base de dades per la taula *training_period_type*.

Remarcar que s'ha decidit representar els tipus de períodes d'entrenament en un objecte i una taula de la base de dades a part, per la relació que tenen amb altres objectes a banda dels períodes d'entrenament. Per exemple, un període d'entrenament de força per preparar-se per una marató o per preparar un Ironman no seran iguals, però tots dos períodes d'entrenament seran del tipus força. A més a més les sessions d'entrenament seran per força.

La classe Training Period Type té els paràmetres següents:

- id: Enter que identifica el tipus de període d'entrenament dins del sistema.
- name: Nom en format text del tipus de període d'entrenament.

7.1.14 Classe Training Period Type Collection

Una instància de la classe Training Period Type Collection representa un conjunt de tipus de períodes d'entrenament del sistema.

7.1.15 Classe Season Athlete

Una instància de la classe Season Athlete representa una assignació d'una temporada a un atleta. La classe està definida en el fitxer `modelObjective.py` i representada a la base de dades per la taula *season_ athlete*.

La classe Season Athlete té els paràmetres següents:

- id: Enter que identifica l'assignació d'una temporada a un atleta.
- dateFrom: Data d'inici de la temporada personalitzada per l'atleta.
- dateTo: Data de finalització de la temporada personalitzada per l'atleta.
- season_id: Id que representa la temporada personalitzada per l'atleta.
- athlete_id: Id que representa l'atleta que se l'hi assigna una temporada personalitzada.
- percentage_strength: Percentatge de força de la temporada.
- percentage_resistance: Percentatge de resistència de la temporada.
- percentage_velocity: Percentatge de velocitat de la temporada.

7.1.16 Classe Season Athlete Collection

Una instància de la classe Season Athlete Collection representa un conjunt de temporades assignades a atletes del sistema.

7.1.17 Classe Solution

Una instància de la classe Solution representa una solució del servei d'assignació de planificacions setmanals personalitzades a un atleta. La classe està definida en el fitxer `modelSolution.py`.

Com s'ha explicat al principi d'aquesta secció, una classe o objecte en la programació orientada a objectes representa estructures de dades més complexes. En el cas de generar una planificació setmanal personalitzada són uns quants els paràmetres que s'han de gestionar i guardar. El fet d'incorporar-ho dins d'una classe de Python ha facilitat molt les tasques, sobretot per un algorisme recursiu que la solució ha d'anar recursivament iterant.

7.2 Descripció dels serveis

Els serveis són l'última part de l'estructura del projecte, que recordem estava dividida en models (classes/objectes definits en el sistema), repositori (manipular la informació amb el mapatge d'objectes relacional a la base de dades) i serveis.

Els serveis realitzen totes les funcionalitats d'aquest projecte, és a dir, cada servei està enfocat a atendre una funcionalitat concreta dins del sistema.

7.2.1 Servei llegir normes disponibilitat

Aquest servei, està enfocat a llegir la disponibilitat de l'atleta i convertir-ho en un format perquè el sistema ho pugui interpretar.

Tal com s'explicarà en detall en les següents seccions del sistema d'optimització amb restriccions, la disponibilitat de l'atleta és un paràmetre d'entrada en format JSON [46] i per tant s'ha de convertir a un format que sigui manipulable pel microservei.

Remarcar que com s'explica en detall en els següents apartats, el servei de realitzar planificacions setmanals personalitzades ha anat evolucionant per millorar l'eficiència i a conseqüència d'aquests canvis també s'ha vist afectat aquest servei. Tots els canvis queden documentats en la implementació.

7.2.2 Servei generar permutacions de les temporades

Servei que a partir d'una temporada predefinida calcula l'adaptació de la temporada per totes les possibles duracions d'aquesta. Les possibles duracions vénen determinades pels paràmetres de duració mínima i màxima de la temporada.

Aquest procés es realitza mitjançant un algorisme iteratiu i queda documentat al detall en la secció d'implementació de temporades personalitzades.

7.2.3 Servei per generar planificacions setmanals personalitzades

Servei que a partir del moment de la temporada en el que passa l'atleta i la seva disponibilitat, entre altres paràmetres, calcula la planificació setmanal de sessions esportives.

Aquesta és la funcionalitat principal del microservei. Tal com queda documentat en la implementació, el servei ha anat evolucionant per millorar l'eficiència temporal.

7.2.4 Servei per preparar informació per la generació de planificacions setmanals

A l'hora de generar planificacions setmanals personalitzades, es necessiten molts paràmetres d'entrada: planificacions setmanals predefinides, moment de l'atleta en la temporada que passa, disponibilitat, criteris per seleccionar la millor planificació...

S'ha separat en un servei a part el procés d'obtenir i preparar tota la informació necessària per generar les planificacions, per temes estructurals i visuals. D'aquesta manera el servei de generar planificacions setmanals personalitzades realitza únicament aquesta tasca, ja que les dades necessàries per realitzar-ho ja venen obtingudes i preparades per aquest altre servei.

Implementació

El nucli del projecte és el disseny de temporades personalitzades a llarg termini i la creació de planificacions setmanals de sessions d'entrenament. Donada la complexitat dels aspectes a tractar i el fet que molts conceptes siguin esportius, s'ha decidit explicar la implementació utilitzant un exemple. D'aquesta manera es contextualitzarà una mica més el seguiment i l'explicació de totes les funcionalitats.

Aquest projecte no està focalitzat a cap esport en concret, de fet està desenvolupat amb la idea de ser aplicat a qualsevol esport. Tot i això, a l'empresa Trainerer la majoria dels esportistes són ciclistes de carretera i per tant l'explicació anirà orientada a aquest esport. També remarcar que s'ha escollit el ciclisme de carretera pel fet que es disposa de més volum de dades.

Per últim, recordar que es dona per descomptat en l'explicació que es diferencien i reconeixen alguns termes esportius, com per exemple temporada esportiva, període d'entrenament dins la temporada, sessió d'entrenament... Tots els conceptes esportius rellevants es troben definits en el glossari de la memòria.

8.1 Explicació mitjançant execució d'un exemple

Un ciclista de carretera vol començar utilitzar el sistema del projecte per assolir els seus objectius esportius. El primer pas del procés és obtenir les dades com a nou atleta. Haurà d'omplir totes les dades d'acord amb la Classe Athlete definida en l'apartat 7.1.1.

Dades de l'atleta:

- *id*: 288.
- *gender*: 'MALE'.
- *height*: 180 cm.
- *weight*: 78 kg.
- *birthDate*: 1993-07-19 (YYYY-MM-DD).
- *restHR*: 80.
- *sport_id*: 8.

Podem observar que *sport_id* és 8. En el sistema el ciclisme de carretera està identificat amb l'enter 8.

A continuació s'enregistrarà l'objectiu esportiu de l'atleta i el nombre de setmanes de les quals disposa per a preparar-se. En la majoria de les ocasions, i sobretot en els atletes més experimentats, acostumen a definir com a objectiu la carrera o prova més important de l'any. D'aquesta manera totes les altres competicions, sortides i entrenaments estan orientades a preparar-se per l'objectiu.

Seguint amb l'exemple anterior, definim doncs l'objectiu principal de la temporada i el nombre de setmanes completes restants per assolir-lo:

- *Objectiu*: Marxa cicloturista.
- *Temps per assolir objectiu*: 24 setmanes (6 mesos)

Finalment, l'atleta ha d'introduir la seva disponibilitat setmanal. Aquesta disponibilitat es pot modificar en qualsevol punt i el sistema s'actualitzarà per adaptar-se a la nova disponibilitat. La disponibilitat d'un atleta, tal com queda definit en el glossari, és per cada dia de la setmana quins esports pot entrenar i quina duració.

- *Dilluns*: Gimnàs o Agilitat (estiraments) o Ciclisme o Córrer, durant 2 hores.
- *Dimarts*: Gimnàs o Agilitat (estiraments) o Ciclisme o Córrer, durant 2 hores.
- *Dimecres*: Gimnàs o Agilitat (estiraments) o Ciclisme o Córrer, durant 2 hores.
- *Dijous*: Gimnàs o Agilitat (estiraments) o Ciclisme o Córrer, durant 2 hores.
- *Divendres*: Gimnàs o Agilitat (estiraments) o Ciclisme o Córrer, durant 3 hores.
- *Dissabte*: Ciclisme 4 hores.
- *Diumenge*: Ciclisme 4 hores.

Una disponibilitat semblant a l'anterior és molt típica en els atletes presents actualment al sistema de Trainerer. De dilluns a dijous, l'atleta pot sortir a córrer, al gimnàs, realitzar estiraments o sortir amb bicicleta durant un període de 2 hores. El divendres el mateix, però amb una duració de 3 hores. Finalment el cap de setmana pot fer sortides amb bicicleta de no més de 4 hores.

Un cop definit l'exemple, podem procedir amb el disseny d'una temporada en el següent apartat.

8.2 Implementació del disseny de temporades

En aquest capítol es profunditzarà en el servei de disseny de temporades pels esportistes. Com a tot servei, té uns paràmetres d'entrada, un procés o algorisme on es manipulen les dades, i finalment uns paràmetres de sortida. Com s'ha dit en el capítol anterior, se seguirà amb l'exemple del ciclista que es prepara per a una marxa cicloturista per l'explicació.

8.2.1 Paràmetres d'entrada i sortida

Els paràmetres d'entrada representen totes les dades necessàries per a l'execució del servei de disseny de temporades, mentre que els paràmetres de sortida representa la resposta d'aquest. A continuació es detallen el conjunt d'aquests paràmetres.

8.2.1.1 Paràmetres d'entrada

El ciclista de carretera es vol preparar per al seu objectiu, una marxa cicloturista (vegeu definició al glossari) que tindrà lloc d'aquí 24 setmanes. Els paràmetres d'entrada al servei de dissenyar una temporada a llarg termini a mida de les necessitats de l'atleta són:

- Id del atleta en el sistema.
- Objectiu.
- Nombre de setmanes per assolir l'objectiu.

8.2.1.2 Paràmetres de sortida

El servei retorna un únic paràmetre de sortida:

- Una temporada (objecte de la classe Season 7.1.9) o NULL.

El paràmetre de sortida és NULL en cas que el sistema cregui que el nombre de setmanes de l'atleta per preparar-se per l'objectiu no són suficients. Aquest fet es detecta quan el nombre de setmanes és inferior al nombre mínim de setmanes associades a l'objectiu dins del sistema.

8.2.1.3 Diagrama

Figura 8.1: Diagrama del servei amb les entrades i sortides.

8.2.2 Algorisme

Els passos que segueix l'algorisme per generar les temporades personalitzades a llarg termini són els següents:

1. Obtenir la temporada predefinida que s'adapta a l'objectiu de l'atleta.
2. Comprovar que el temps de preparació sigui l'adequat.
3. Generar permutacions amb els períodes d'entrenament prescindibles.
4. Reduir la duració dels períodes d'entrenament per ordre de prioritat, per tal d'obtenir totes les temporades per les diferents setmanes possibles.
5. Resoldre casos aïllats.
6. Retornar la temporada entre totes les generades que s'ajusta a la disponibilitat de l'atleta.

En la Figura 8.2 es pot observar el procés en forma de diagrama de flux.

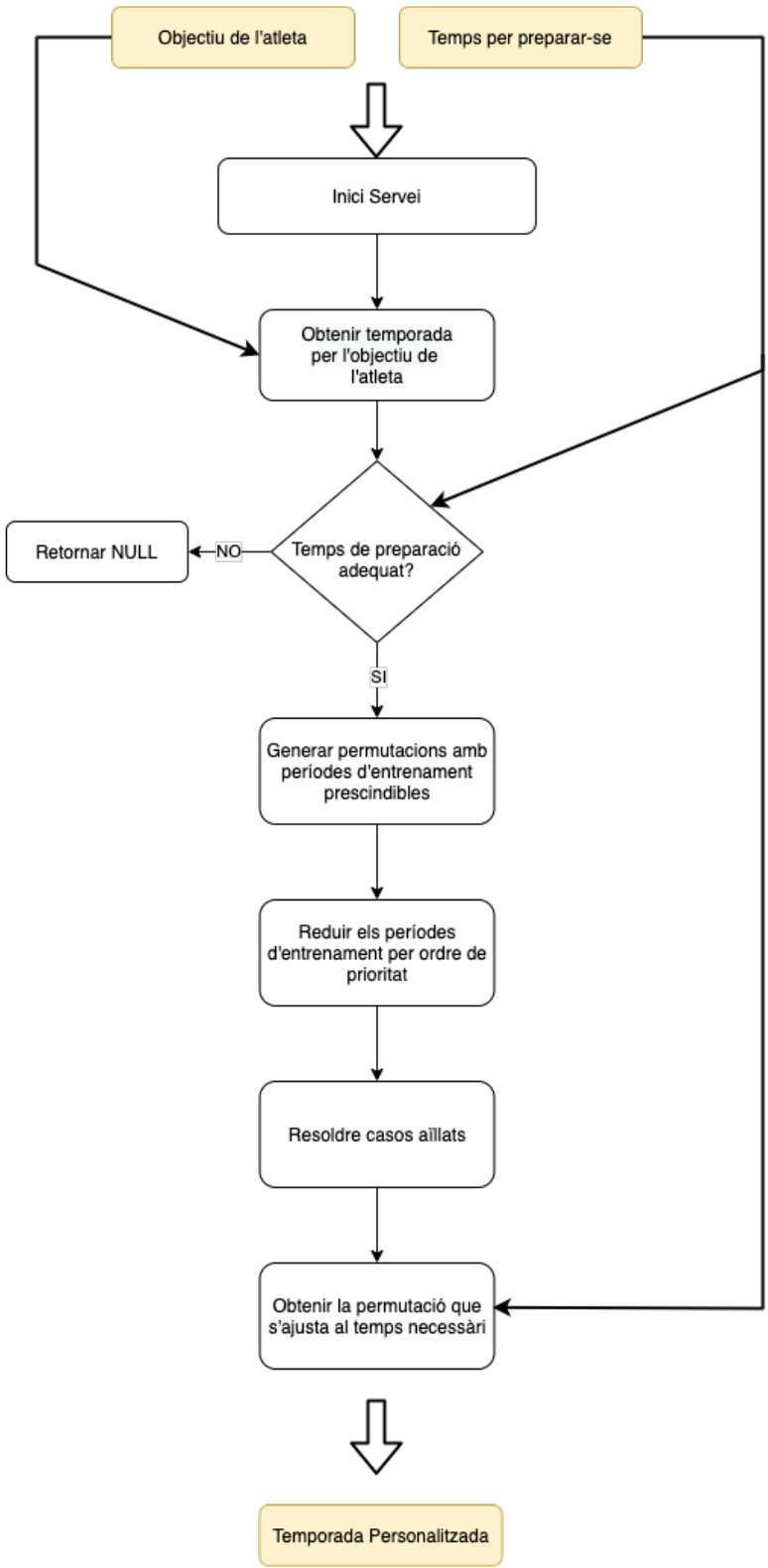


Figura 8.2: Diagrama de flux de l'algorisme per generar temporades.

8.2.2.1 Obtenció de la temporada relacionada amb l'objectiu

Aquest primer pas és un pas trivial. El sistema està pensat per tenir enregistrats tots els tipus d'objectius i de proves esportives. D'aquesta manera si un ciclista es vol preparar una prova en concret, el sistema tindrà enregistrat una temporada de preparació per la prova. De la mateixa manera, un corredor que es vol preparar per a una marató, el sistema tindrà una temporada predefinida per preparar-ser per realitzar una marató.

En el cas del ciclista de l'exemple, es vol preparar per a una marxa cicloturista. En el sistema, per preparar-se pel seu objectiu, trobem la temporada predefinida següent (Figura 8.3 i Figura 8.4).

Temporada ciclisme - Marxa cicloturista			
Nom	Núm. setmanes	Prescindible	Prioritat
Base	8 setmanes	Si	1
Velocitat	4 setmanes	No	2
Resistència	4 setmanes	No	3
Força-Resistència	4 setmanes	No	5
Especialització	4 setmanes	No	6
Competició	4 setmanes	No	4
	28 setmanes		

Figura 8.3: Temporada estàndard del sistema per marxes cicloturistes.

Temporada ciclisme - Marxa cicloturista	
Min. Setmanes	Max. Setmanes
15	28

Figura 8.4: Mínim i màxim de setmanes per preparar-se per una marxa cicloturista.

Observem doncs que la temporada estàndard del sistema consta de 6 períodes d'entrenament: Base, Velocitat, Resistència, Força-Resistència, Especialització i Competició, i un número mínim i un nombre màxim de setmanes per preparar-se.

8.2.2.2 Comprovació del temps de preparació de l'atleta en base l'objectiu

Un cop s'ha seleccionat la temporada estàndard en funció de l'objectiu de l'atleta, s'entra en el segon pas del procés per dissenyar temporades. Aquest segon pas és imprescindible i consisteix a verificar que el temps de preparació de l'atleta per assolir el seu objectiu esportiu està entre el mínim i el màxim nombre de setmanes establertes per la temporada estàndard.

En el cas del ciclista de l'exemple, s'ha definit que disposa d'un total de 24 setmanes per preparar-se per l'objectiu. Com que es troba entre el mínim de setmanes i el màxim de setmanes, 15 i 28 per la marxa cicloturista, es pot procedir amb el següent pas.

En cas contrari, el servei retorna el valor NULL i un error que indica que no es pot procedir pel fet que el nombre de setmanes de l'atleta no és l'adequat pel seu objectiu.

8.2.2.3 Generar permutacions en funció dels períodes d'entrenament prescindibles

Un cop arribat en aquest tercer pas, el sistema ja pot assegurar que disposa d'una temporada estàndard per l'objectiu de l'atleta, i que aquest disposa d'un nombre de setmanes adequat per entrenar-se.

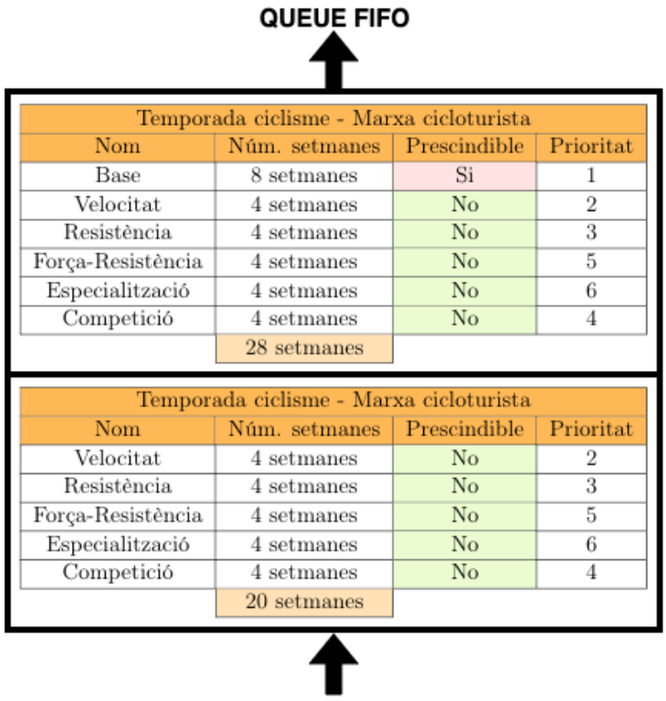
El següent pas consisteix a crear totes les possibles temporades, basant-se en la temporada estàndard de l'objectiu, de manera que s'obtenen totes les permutacions

d'aquesta eliminant-hi els períodes d'entrenament prescindibles. En el projecte s'ha fet ús d'una cua FIFO amb l'objectiu d'anar empilant totes les possibles temporades. Una cua FIFO permet al sistema mantenir l'ordre de temporades, de manera que el principi queden les temporades sense eliminar els períodes prescindibles i al final les que prescindeixen d'aquests.

El procés que segueix el sistema utilitzant cues és el següent:

1. Inserir a la cua la temporada original.
2. Comprovar si hi ha cap període d'entrenament prescindible. En cas contrari, saltar al pas 6.
3. Crear una nova temporada eliminant el període d'entrenament prescindible.
4. Inserir a la cua la nova temporada creada.
5. Saltar al pas 2.
6. Fi.

En l'exemple del ciclista, la temporada estàndard original és de 28 setmanes amb 1 període d'entrenament prescindible (Base). El resultat d'aplicar el procés anterior és una cua amb les següents temporades:



Temporada ciclisme - Marxa cicloturista			
Nom	Núm. setmanes	Prescindible	Prioritat
Base	8 setmanes	Si	1
Velocitat	4 setmanes	No	2
Resistència	4 setmanes	No	3
Força-Resistència	4 setmanes	No	5
Especialització	4 setmanes	No	6
Competició	4 setmanes	No	4
	28 setmanes		

Temporada ciclisme - Marxa cicloturista			
Nom	Núm. setmanes	Prescindible	Prioritat
Velocitat	4 setmanes	No	2
Resistència	4 setmanes	No	3
Força-Resistència	4 setmanes	No	5
Especialització	4 setmanes	No	6
Competició	4 setmanes	No	4
	20 setmanes		

Figura 8.5: Cua amb les permutacions de totes les temporades.

8.2.2.4 Reduir la duració de temporades seguint ordre de prioritat

Un cop el procés del servei ha arribat en aquest punt, el sistema ja comença a definir possibles temporades definitives, a partir de la informació de les temporades de la cua anterior. Un període d'entrenament d'una temporada és empre inicialment múltiple de 4 setmanes (vegeu definició al glossari). A l'hora d'adaptar una temporada per un atleta, és a dir, reduir una temporada a la disponibilitat en nombre de setmanes

de l'atleta, es redueixen progressivament els períodes d'entrenament a múltiples de 3 setmanes.

Per exemple, en la temporada estàndard per preparar-se per una marxa cicloturista, trobem un període d'entrenament Base de 8 setmanes (4 setmanes + 4 setmanes). Pel procés anterior, es redueixen els períodes d'entrenament de 4 setmanes a 3 setmanes, i per tant, al reduir el nombre de setmanes s'obtindria un període d'entrenament de 7 setmanes (4 setmanes + 3 setmanes) o un període d'entrenament de 6 setmanes (3 setmanes + 3 setmanes).

El procés que segueix el sistema, definint la prioritat a l'hora de seleccionar els períodes d'entrenament a reduir, és el següent:

1. Es crea una estructura de dades key:value (Dict a Python), i s'inicialitza amb keys igual al valor entre el mínim i el màxim nombre de setmanes.
2. Mentre la cua no sigui buida o no s'hagin assignat una temporada a totes les claus de l'estructura anterior, anar pas 3. En cas contrari, anar pas 10.
3. Obtenir una temporada de la cua.
4. Si no existeix cap temporada amb key igual el nombre total de setmanes de la temporada, assignem la temporada a la key corresponent.
5. Mentre es pugui reduir algun període d'entrenament, anar pas 6. En cas contrari al pas 9.
6. Crear una nova temporada reduir a 3 setmanes el període d'entrenament amb prioritat més petita.
7. Si no existeix cap temporada amb key igual el nombre total de setmanes de la temporada, assignem la temporada a la key corresponent.
8. Anar pas 5.
9. Anar pas 2.
10. Fi.

Es pot observar del procés anterior que només es guarda una nova temporada, en el cas que no se n'hagi creat cap amb el mateix nombre de setmanes prèviament. D'aquesta manera s'estableix un ordre de prioritat en cas de tenir dues temporades amb el mateix nombre de setmanes. Recordar que en la cua de temporades, a l'inici d'aquesta hi havia les temporades amb més períodes d'entrenament. Com a conseqüència, s'estableix la preferència a temporades amb més nombre de períodes d'entrenament que a temporades amb menys.

El resultat obtingut en aplicar el procés anterior a la cua de temporades estàndards pel ciclista que es prepara per a una marxa cicloturista com a objectiu, es mostra en la Figura 8.6.

Temporada ciclisme - Marxa cicloturista			
Nom	Núm. setmanes	Prescindible	Prioritat
Base	8 setmanes	Si	1
Velocitat	4 setmanes	No	2
Resistència	4 setmanes	No	3
Força-Resistència	4 setmanes	No	5
Especialització	4 setmanes	No	6
Competició	4 setmanes	No	4
	28 setmanes		

Temporada ciclisme - Marxa cicloturista			
Nom	Núm. setmanes	Prescindible	Prioritat
Base	7 setmanes	Si	1
Velocitat	4 setmanes	No	2
Resistència	4 setmanes	No	3
Força-Resistència	4 setmanes	No	5
Especialització	4 setmanes	No	6
Competició	4 setmanes	No	4
	27 setmanes		

Temporada ciclisme - Marxa cicloturista			
Nom	Núm. setmanes	Prescindible	Prioritat
Base	6 setmanes	Si	1
Velocitat	4 setmanes	No	2
Resistència	4 setmanes	No	3
Força-Resistència	4 setmanes	No	5
Especialització	4 setmanes	No	6
Competició	4 setmanes	No	4
	26 setmanes		

Temporada ciclisme - Marxa cicloturista			
Nom	Núm. setmanes	Prescindible	Prioritat
Base	6 setmanes	Si	1
Velocitat	3 setmanes	No	2
Resistència	4 setmanes	No	3
Força-Resistència	4 setmanes	No	5
Especialització	4 setmanes	No	6
Competició	4 setmanes	No	4
	25 setmanes		

Temporada ciclisme - Marxa cicloturista			
Nom	Núm. setmanes	Prescindible	Prioritat
Base	6 setmanes	Si	1
Velocitat	3 setmanes	No	2
Resistència	3 setmanes	No	3
Força-Resistència	4 setmanes	No	5
Especialització	4 setmanes	No	6
Competició	4 setmanes	No	4
	24 setmanes		

Temporada ciclisme - Marxa cicloturista			
Nom	Núm. setmanes	Prescindible	Prioritat
Base	6 setmanes	Si	1
Velocitat	3 setmanes	No	2
Resistència	3 setmanes	No	3
Força-Resistència	4 setmanes	No	5
Especialització	4 setmanes	No	6
Competició	3 setmanes	No	4
	23 setmanes		

Temporada ciclisme - Marxa cicloturista			
Nom	Núm. setmanes	Prescindible	Prioritat
Base	6 setmanes	Si	1
Velocitat	3 setmanes	No	2
Resistència	3 setmanes	No	3
Força-Resistència	3 setmanes	No	5
Especialització	4 setmanes	No	6
Competició	3 setmanes	No	4
	22 setmanes		

Temporada ciclisme - Marxa cicloturista			
Nom	Núm. setmanes	Prescindible	Prioritat
Base	6 setmanes	Si	1
Velocitat	3 setmanes	No	2
Resistència	3 setmanes	No	3
Força-Resistència	3 setmanes	No	5
Especialització	3 setmanes	No	6
Competició	3 setmanes	No	4
	21 setmanes		

Temporada ciclisme - Marxa cicloturista			
Nom	Núm. setmanes	Prescindible	Prioritat
Velocitat	4 setmanes	No	2
Resistència	4 setmanes	No	3
Força-Resistència	4 setmanes	No	5
Especialització	4 setmanes	No	6
Competició	4 setmanes	No	4
	20 setmanes		

Temporada ciclisme - Marxa cicloturista			
Nom	Núm. setmanes	Prescindible	Prioritat
Velocitat	3 setmanes	No	2
Resistència	4 setmanes	No	3
Força-Resistència	4 setmanes	No	5
Especialització	4 setmanes	No	6
Competició	4 setmanes	No	4
	19 setmanes		

Temporada ciclisme - Marxa cicloturista			
Nom	Núm. setmanes	Prescindible	Prioritat
Velocitat	3 setmanes	No	2
Resistència	3 setmanes	No	3
Força-Resistència	4 setmanes	No	5
Especialització	4 setmanes	No	6
Competició	4 setmanes	No	4
	18 setmanes		

Temporada ciclisme - Marxa cicloturista			
Nom	Núm. setmanes	Prescindible	Prioritat
Velocitat	3 setmanes	No	2
Resistència	3 setmanes	No	3
Força-Resistència	4 setmanes	No	5
Especialització	4 setmanes	No	6
Competició	3 setmanes	No	4
	17 setmanes		

Temporada ciclisme - Marxa cicloturista			
Nom	Núm. setmanes	Prescindible	Prioritat
Velocitat	3 setmanes	No	2
Resistència	3 setmanes	No	3
Força-Resistència	3 setmanes	No	5
Especialització	4 setmanes	No	6
Competició	3 setmanes	No	4
	16 setmanes		

Temporada ciclisme - Marxa cicloturista			
Nom	Núm. setmanes	Prescindible	Prioritat
Velocitat	3 setmanes	No	2
Resistència	3 setmanes	No	3
Força-Resistència	3 setmanes	No	5
Especialització	3 setmanes	No	6
Competició	3 setmanes	No	4
	15 setmanes		

Figura 8.6: Permutacións de temporades seguint l'algorisme de reducció de períodes d'entrenament.

8.2.2.5 Resoldre casos aïllats

L'últim pas del procés consisteix en la resolució dels casos aïllats. El concepte de casos aïllats en el sistema fa referència a possibles excepcions. És a dir, en alguns casos concrets, es podria donar el cas que no hi hagués una temporada vàlida per un nombre concret de setmanes només reduint de períodes de 4 a 3 setmanes. En aquests casos, es procedeix a reduir a períodes de 2 setmanes aquell o aquells períodes d'entrenament de menys rellevància dins la temporada.

En l'exemple del ciclista, si volgués també una temporada de duració 14 set-

manes, el sistema es trobaria davant d'un cas aïllat d'aquestes característiques. El procediment que seguiria el sistema per tractar-lo és:

1. Començant per l'excepció de nombre de setmanes més grans (si hi ha més d'una), agafem la temporada de duració per sobre de l'excepció.
2. Reduïm a 2 setmanes el període amb una prioritat més baixa que tingui duració de 3 setmanes.

El resultat de l'execució per obtenir el cas aïllat de la temporada de 14 setmanes és:

Temporada ciclisme - Marxa cicloturista			
Nom	Núm. setmanes	Prescindible	Prioritat
Velocitat	2 setmanes	No	2
Resistència	3 setmanes	No	3
Força-Resistència	3 setmanes	No	5
Especialització	3 setmanes	No	6
Competició	3 setmanes	No	4
	14 setmanes		

Figura 8.7: Temporada per l'objectiu d'una marxa cicloturista de 14 setmanes

8.2.2.6 Obtenir permutació que s'ajusta el temps de preparació de l'atleta

L'últim pas del servei implica seleccionar la temporada que s'ajusta al nombre de setmanes que disposa l'atleta. Com a paràmetre d'entrada, el ciclista de l'exemple havia introduït que disposava de 24 setmanes per preparar-se, i per tant el resultat del servei serà el de la Figura 8.8.

Temporada ciclisme - Marxa cicloturista			
Nom	Núm. setmanes	Prescindible	Prioritat
Base	6 setmanes	Si	1
Velocitat	3 setmanes	No	2
Resistència	3 setmanes	No	3
Força-Resistència	4 setmanes	No	5
Especialització	4 setmanes	No	6
Competició	4 setmanes	No	4
	24 setmanes		

Figura 8.8: Resultat del servei de generar temporades.

8.2.3 Cost temporal generació de temporades

Per últim, s'analitzarà el cost del temps d'execució del servei de disseny de planificacions. En l'exemple del ciclista que es prepara per a la marxa cicloturista, el temps d'execució per crear totes les permutacions de temporades i seleccionar la temporada de 26 setmanes ha estat de **0.02377605438232422** segons.

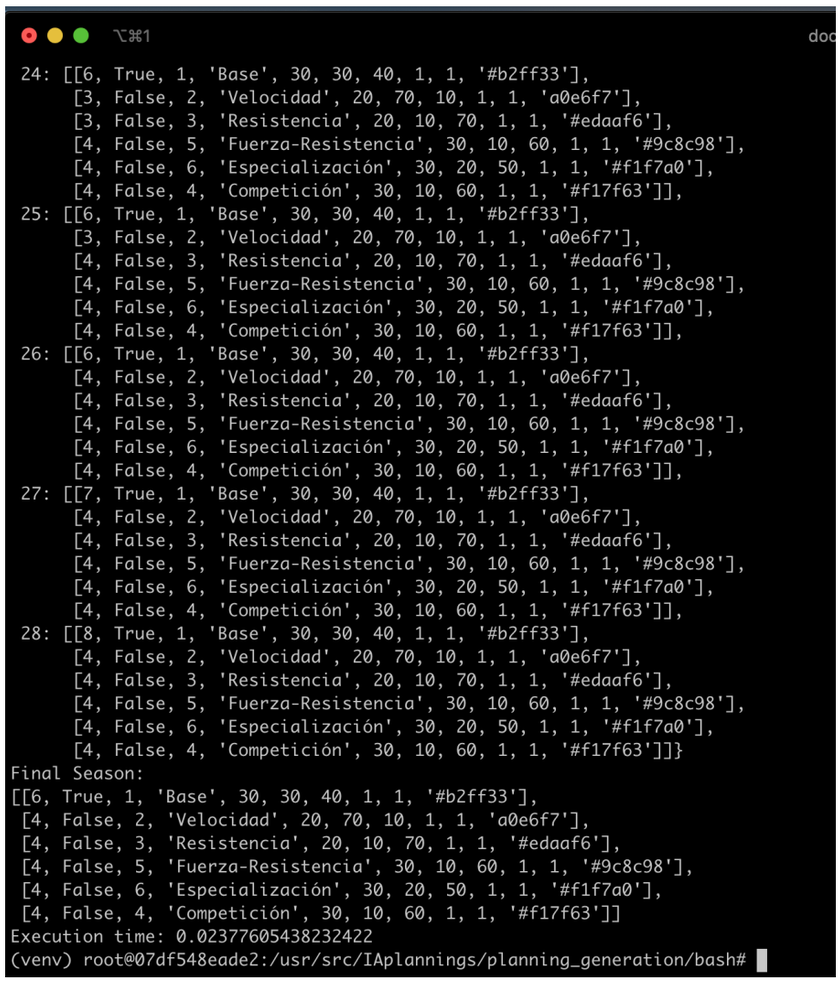


Figura 8.9: Temps d'execució del servei des del terminal del contenidor del projecte.

8.2.4 Anàlisi final del cost

Per últim, analitzarem el cost de l'algorisme anterior. El cost està dividit en diferents parts que es desglossen a continuació.

Per facilitar els càlculs es defineixen els paràmetres N, n i D.

- N: nombre de períodes d'entrenament de la temporada.
- n: nombre de períodes d'entrenament prescindibles.
- D: diferència entre el màxim i mínim nombre de setmanes en què es pot dissenyar una temporada.

8.2.4.1 Cost obtenció temporada estàndard per l'objectiu

És el primer pas de l'algorisme. Totes les temporades estàndards es troben en la base de dades relacional, dins del contenidor de MariaDB. Aquest fet provoca que aquest cos estigui lligat a la consulta a base de dades, i per tant donarem el cost en forma de constant:

$$O(\text{obtenir temporada estàndard})$$

8.2.4.2 Cost generar permutacions amb períodes prescindibles.

Com s'ha vist consisteix en posar a la cua les diferents permutacions de temporades, eliminant els períodes d'entrenament prescindibles successivament.

$$O(N * n)$$

En el pitjor cas, quan $n = N$ seria:

$$O(N^2)$$

8.2.4.3 Cost reduir períodes d'entrenament a múltiples de 3

. Per cadascuna de les possibles setmanes, s'han de reduir tots els períodes d'entrenament a múltiples de 4.

$$O(D * N * n)$$

En el pitjor cas, el nombre de períodes prescindibles seria N i D seria igual o superior a N. Per tant, es pot expressar com:

$$O(N^3)$$

8.2.4.4 Cost tractar casos aïllats

A l'hora de tractar els casos aïllats el primer pas és identificar-los. Aquest primer pas es fa amb cost constant $O(1)$, ja que l'estructura de dades que emmagatzema les temporades té la forma key:value. A continuació s'ha de localitzar el període d'entrenament que toca reduir $O(N)$ i emmagatzemar els resultats. El cost total és:

$$O(N)$$

8.2.4.5 Cost de seleccionar la temporada de longitud adequada

Com s'ha explicat anteriorment, les temporades s'emmagatzemen en una estructura de key:value. Això significa que es pot obtenir els valors amb cost $O(1)$ (Python dictionary [9]. El cost total és:

$$O(1)$$

8.2.4.6 Cost total

El cost total de l'algorisme serà la suma del cost de tots els apartats analitzats anteriorment.

$$\text{Cost} = O(\text{obtenir temporada estàndard}) + O(N^2) + O(N^3) + O(N) + O(1)$$

$$\text{Cost} = O(\text{obtenir temporada estàndard}) + O(N^3)$$

Cal remarcar que N (nombre de períodes d'entrenament és un paràmetre que acostuma a ser molt baix. En la plataforma actual de Trainerer la temporada amb un número més gran de períodes d'entrenament és 7. Aquest fet significa que el principal cost es troba amb la connexió i execució de la consulta a la base de dades.

8.3 Implementació planificacions setmanals personalitzades

La segona part del projecte consisteix a realitzar les planificacions setmanals personalitzades pels atletes. En aquest punt es fa imprescindible que l'atleta tingui una temporada ja definida pel seu objectiu, ja que saber en quin punt es troba l'atleta de la temporada és la base per generar les planificacions setmanals. Com és lògic, és diferent generar una planificació setmanal per un ciclista que es troba en el moment de la temporada d'entrenar la velocitat, que la planificació setmanal del moment que treballa la resistència.

Tal com es pot observar, aquesta memòria recull dues implementacions per generar planificacions setmanals, les quals utilitzen mètodes i implementacions completament diferents. L'objectiu d'aquest capítol serà posar context el problema de generar planificacions setmanals personalitzades, és a dir, definir el problema, definir els paràmetres d'entrada i sortida, i introduir els enfocaments per resoldre'l.

Tota explicació d'aquest capítol i de les dues implementacions posteriors, se seguirà amb l'exemple del ciclista que es prepara per a la marxa cicloturista del qual ja se'n disposa de la seva temporada de 24 setmanes.

8.3.1 Definició del problema

Com a primer pas, es comença definint i contextualitzant el problema que presenta la realització de planificacions setmanals personalitzades a partir de planificacions setmanals predefinides. Si s'enllaça amb l'exemple del ciclista, se sap que es prepara per a una marxa cicloturista i per aquest motiu se li ha associat una temporada a mida per preparar-se. Per tant, en aquest punt el sistema ja té constància de quins aspectes es treballaran cada setmana fins al final de la temporada (vegeu Figura 8.10).

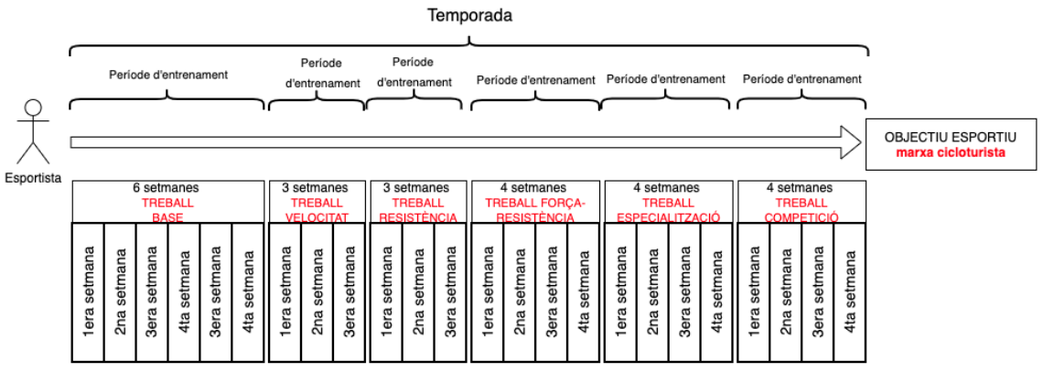


Figura 8.10: Temporada ciclista 24 setmanes marxa cicloturista.

Com es pot observar a la figura anterior, cada setmana, s'haurà de realitzar per l'atleta la planificació setmanal que millor s'ajusti al moment de la temporada al qual es troba i s'ajusti també a la seva disponibilitat.

8.3.1.1 Scheduling Problems

El problema a resoldre es pot incloure dins del grup de problemes coneguts com a *Scheduling Problems* [3], més en concret, a un problema de *Task Scheduling*. De

manera senzilla, tenim un conjunt de sessions d'entrenament que el sistema ha de distribuir a la disponibilitat setmanal de l'atleta basant-se en un grup de criteris (setmana dins la temporada, preferències de l'atleta, ...).

Els problemes de *Task Scheduling* es classifiquen en dos grups:

- Trobar una distribució de tasques que satisfaci una condició.
- Trobar la millor distribució de tasques en funció d'una heurística o sistema de puntuació.

El problema de *Task Scheduling* d'aquest projecte està inclòs en el segon grup. Com s'estudiarà en detall a continuació, el sistema busca la millor planificació setmanal personalitzada a cada setmana de la temporada per l'atleta. Per trobar la millor planificació setmanal personalitzada, s'ha creat un sistema de puntuació de planificacions que permet saber quan de bona és una planificació i també comparar unes amb altres. Es pot observar com clarament el projecte està en el segon grup, en què l'objectiu és trobar la millor distribució de tasques en base una funció heurística.

8.3.1.2 Problema NP-Difícil

El problema de trobar la millor distribució de sessions d'entrenament en funció d'un sistema de puntuació de planificacions, és a dir el conjunt de problemes del segon grup, és que són problemes NP-Difícils. Aquest fet fa que a l'hora de resoldre'l o verificar una solució, sigui molt costós.

Un problema NP-Difícil [49] es pot reduir a un problema NP-Complet, és a dir, és un problema que no es pot resoldre en temps polinòmic. Per l'altra banda, pel fet que és un problema NP-Difícil, donada una distribució de tasques (planificació setmanal), tampoc podem saber en temps polinòmic que aquesta sigui la solució (millor distribució segons el sistema de puntuació).

Aquesta dificultat a l'hora de crear planificacions setmanals personalitzades ha fet que s'explorés més d'una alternativa com a implementació i resolució del problema.

8.3.2 Sistema de puntuació per comparar planificacions

Com ja s'ha dit en l'apartat anterior, per poder definir el problema i resoldre'l trobant la millor distribució de sessions d'entrenament en la disponibilitat setmanal de l'atleta, cal saber quan una planificació setmanal és millor que una altra. S'ha decidit utilitzar un sistema de puntuació de planificacions per a valorar amb un enter positiu quan de bona és una planificació. Per dissenyar el sistema de puntuació de planificacions es relacionen factors d'una planificació amb una puntuació. Aquesta puntuació forma part dels paràmetres d'entrada del servei de generar planificacions setmanals personalitzades, de manera que cada atleta tindrà el seu sistema de puntuació de planificacions propi.

Els factors del sistema de puntuació de planificacions setmanals són els següents:

- Punts per sessió assignada.
- Punts per dies concrets de descans.

- Punts per assignar la sessió de qualitat.
- Punts per assignar sessió de qualitat dimecres o dijous.
- Punts per mantenir patró de ciclisme setmana anterior.
- Punts per mantenir patró de córrer setmana anterior.
- Punts per mantenir patró de gimnàs setmana anterior.
- Punts per mantenir patró d'agilitat/estiraments setmana anterior.

8.3.2.1 Punts per sessió assignada

Es calcularà el nombre de sessions assignades a la planificació setmanal. Es retornà el nombre de sessions multiplicat pels punts per sessió assignada.

8.3.2.2 Punts per dies concrets de descans

S'avaluarà quants dies de descans coincideixen amb els dies de descans especificats. Es retornarà tants punts com dia de descans mantingut.

8.3.2.3 Punts per assignar la sessió de qualitat

La sessió de qualitat és la més important de la setmana (vegeu sessió de qualitat al glossari), ja que es treballen realment els aspectes més rellevants de la setmana que es troba l'atleta dins la temporada. Es retornarà els punts per sessió de qualitat assignada en cas que es produeixi.

8.3.2.4 Punts per assignar sessió de qualitat dimecres o dijous

Com ja s'ha comentat, la sessió de qualitat és la més rellevant de la setmana. Aquest fet fa que sigui aconsellable que es planifiqui a mitjans de la setmana, ja que el cap de setmana normalment es realitzen les sessions d'entrenament més llargues i a principi de setmana sessions suaus o de gimnàs. Es retornarà els punts especificats en cas que la sessió de qualitat es trobi assignada el dimecres o dijous de la planificació.

8.3.2.5 Punts per mantenir patró de ciclisme setmana anterior

Els atletes els agrada mantenir un patró d'esports setmana rere setmana dins la temporada, és a dir, si a un atleta les últimes setmanes se l'hi ha programat una sessió de gimnàs els dilluns i sortides de córrer els dimecres, en cas que a la setmana vinent s'hagi d'assignar sessions de córrer i gimnàs, que s'assignin el dilluns i dimecres respectivament. El patró del ciclisme permet saber al sistema de planificacions, quins dies de la setmana seria bo o aconsellable que se li assignessin sessions de ciclisme a l'atleta. Es retorna els punts corresponents per cada dia del patró mantingut amb una sessió de ciclisme.

8.3.2.6 Punts per mantenir patró de córrer setmana anterior

La mateixa situació que el cas anterior. Es retorna els punts corresponents per cada dia del patró mantingut amb una sessió de córrer.

8.3.2.7 Punts per mantenir patró de gimnàs setmana anterior

La mateixa situació que el cas anterior. Es retorna els punts corresponents per cada dia del patró mantingut amb una sessió de gimnàs.

8.3.2.8 Punts per mantenir patró d'agilitat/estiraments setmana anterior

La mateixa situació que el cas anterior. Es retorna els punts corresponents per cada dia del patró mantingut amb una sessió d'estiraments.

8.3.3 Paràmetres d'entrada i sortida

Un cop s'ha arribat aquest punt, on ja s'ha contextualitzat el problema de la generació de planificacions setmanals, es procedirà a definir els paràmetres d'entrada i sortida del servei que realitza aquesta funcionalitat en el projecte.

8.3.3.1 Paràmetres d'entrada

Els paràmetres d'entrada al servei de generar planificacions setmanals personalitzades són:

- Disponibilitat setmanal de l'atleta.
- Setmana de la temporada en la qual es troba.
- Sistema de puntuació de planificacions:
  - Punts per sessió assignada.
  - Punts per dies concrets de descans.
  - Punts per assignar la sessió de qualitat.
  - Punts per assignar sessió de qualitat dimecres o dijous.
  - Punts per mantenir patró de ciclisme setmana anterior.
  - Punts per mantenir patró de córrer setmana anterior.
  - Punts per mantenir patró de gimnàs setmana anterior.
  - Punts per mantenir patró d'agilitat/estiraments setmana anterior.

8.3.3.2 Paràmetres de sortida

Com a paràmetre de sortida del servei s'obtindrà la planificació setmanal personalitzada per assignar a l'atleta.

8.3.3.3 Diagrama

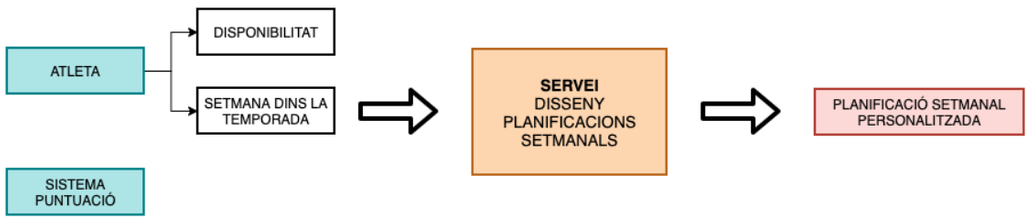


Figura 8.11: Diagrama del server amb entrades i sortides.

8.3.4 Enfocaments per resoldre el problema

Donada la complexitat del problema a resoldre explicada prèviament, ha generat que s'estudiés més d'una via d'implementació. En els capítols posteriors a l'actual, s'analitzarà en profunditat cadascuna de les implementacions realitzades i els resultats obtinguts. Finalment es realitzarà una comparativa dels resultats i temps d'execució.

La primera està basada en un algorisme recursiu i la segona en un model d'optmització amb restriccions.

8.3.4.1 Algorisme Recursiu

La primera implementació que es va realitzar va ser mitjançant un algorisme recursiu. La idea general és recórrer totes les possibles permutacions de setmanes, és a dir, totes les possibles combinacions de les sessions d'entrenament en totes les disponibilitats, per tal d'obtenir la millor solució de totes. La solució està basada en el sistema de puntuació de planificacions específic de l'atleta.

8.3.4.2 Problema d'optimització amb restriccions

La segona implementació que es va realitzar va ser formulant el problema com un problema d'optimització amb restriccions. La idea general és utilitzar el paquet de Python Pyomo per desenvolupar un model d'optimització amb les variables i restriccions necessàries per a maximitzar la funció objectiu, utilitzant el sistema de puntuació de planificacions establert. Aquests models es poden executar en una gran varietat de Solvers [54], des dels comercials com Gurobi [24] i CPLEX [28] o els de codi obert CBC [5] o GLPK [19].

8.3.5 Implementació planificacions setmanals: mètode recursiu

En aquest apartat s'explica el funcionament, procediment, resultats i temps d'execució de la primera implementació explorada per generar planificacions setmanals personalitzades. De la mateixa manera que s'ha realitzat en l'explicació del disseny de temporades, se seguirà amb l'exemple del ciclista que es prepara per a una marxa cicloturista. Recordar que ja s'havia obtingut una temporada de 24 setmanes per preparar-se per l'objectiu del ciclista.

8.3.5.1 Procediment

A continuació queda detallat el procés que segueix la primera implementació de generació de planificacions setmanals. Recordar que en el capítol anterior ja s'han definit els paràmetres d'entrada i sortida (apartat 8.3.3.

1. Obtenir les planificacions setmanals predefinides que s'ajusten a la setmana de la temporada que passa l'atleta.
2. Llegir la disponibilitat de l'atleta i transformar-ho en tota la possible combinatòria de setmanes possibles. Per exemple, si un atleta dilluns pot anar a córrer o al gimnàs 1 hora, es crearan 2 setmanes, una per cada combinació.
3. Per cadascuna de les setmanes creades a partir de la disponibilitat, es realitzaran totes les possibles combinacions amb les sessions d'entrenament de cadascuna de les planificacions setmanals predefinides. El sistema es quedarà amb la millor basant-se amb el sistema de puntuació de planificacions.
4. Seleccionar la millor setmana amb la millor combinatòria de sessions.

El fet que hi hagi una doble iteració, per cadascuna de les planificacions predefinides i per cadascuna de les possibles setmanes de disponibilitat, dificulta i enreda el procés. Per aclarir el procediment veure la Figura 8.12.

Diagrama

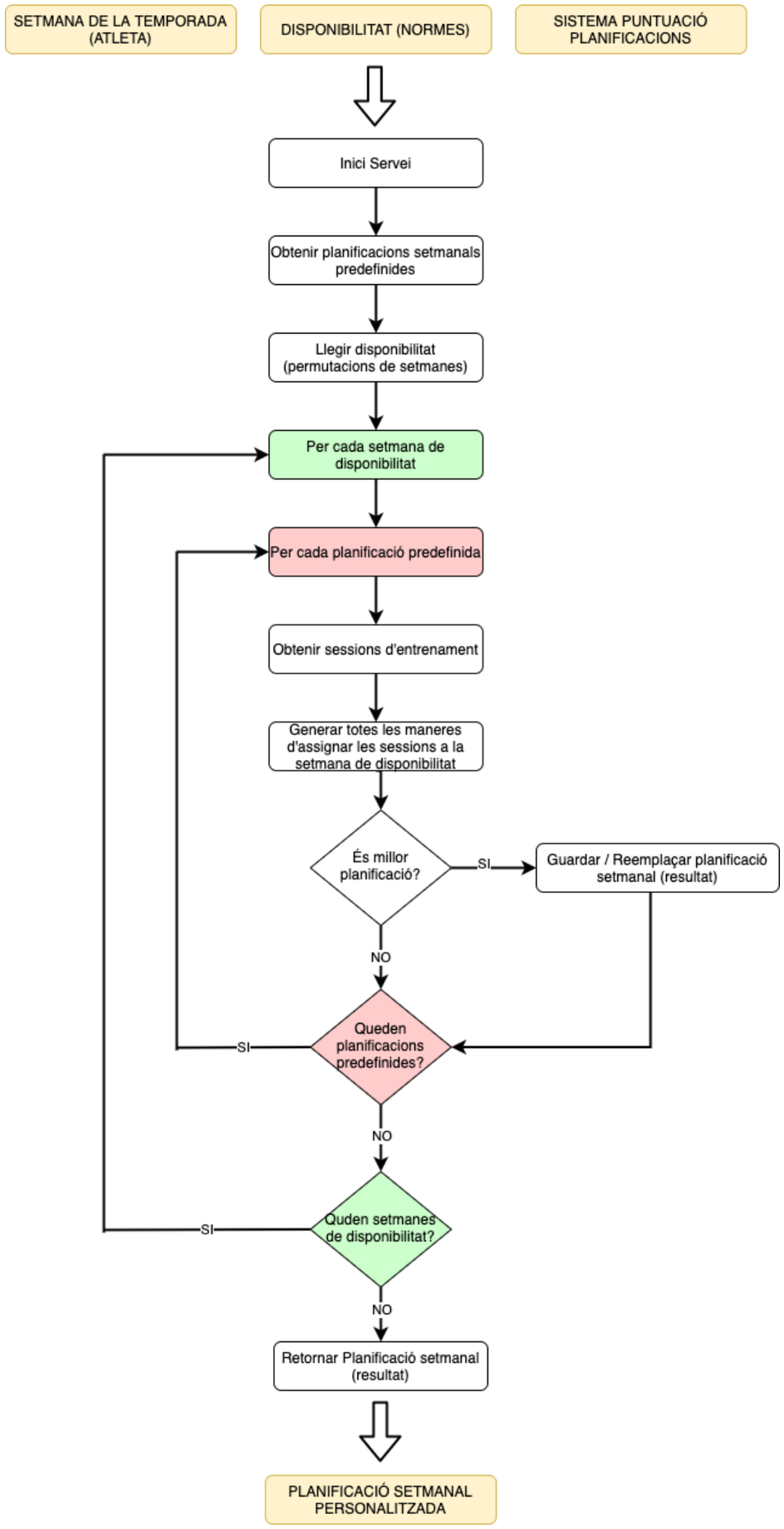


Figura 8.12: Diagrama de flux del procés de generar planificacions setmanals amb implemantació recursiva.

8.3.5.2 Planificacions setmanals predefinides

El primer pas del procés d'aquesta implementació consisteix a obtenir les planificacions setmanals predefinides que es corresponen amb la setmana de l'atleta dins la temporada. Aquestes planificacions s'obtenen de la base de dades i es realitza el mapatge corresponent a la classe de planificació. Per aquest motiu, un dels paràmetres d'entrada imprescindible d'aquest servei és el moment de la temporada en el qual l'atleta es troba.

Si tornem a l'exemple del ciclista que es prepara per a una marxa cicloturista, suposarem a partir d'aquest moment que ens trobem en el punt que s'ha sol·licitat al servei la planificació setmanal de la **segona setmana de velocitat**:

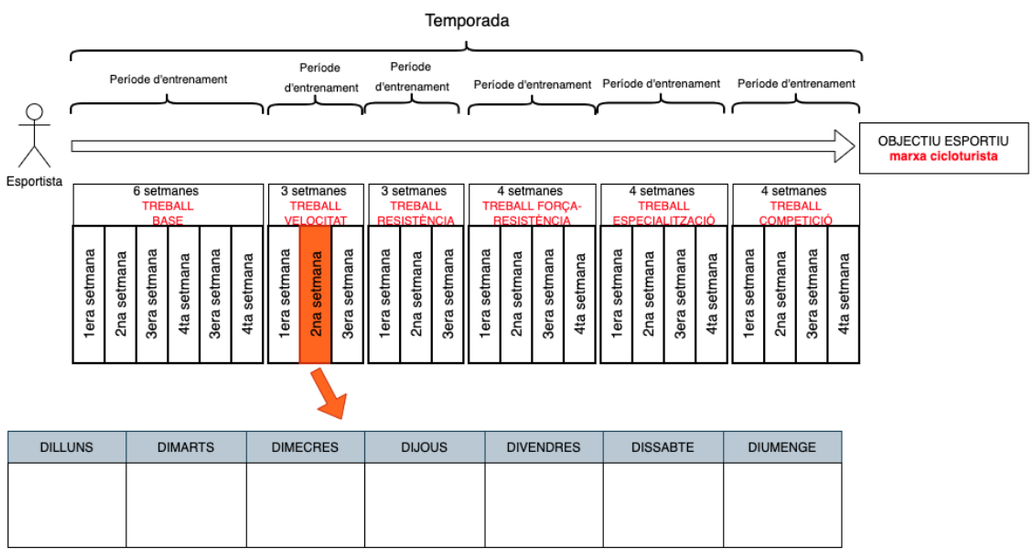


Figura 8.13: Selecció de la setmana a planificar dins la temporada.

El sistema obtindrà totes les planificacions de ciclisme de la segona setmana dins del període d'entrenament de velocitat i les emmagatzemarà per les tasques següents.

8.3.5.3 Disponibilitat

El segon pas, un cop s'ha obtingut el conjunt de planificacions setmanals predefinides de la setmana corresponent, és llegir la disponibilitat de l'atleta. La disponibilitat setmanal, també anomenada regles de disponibilitat pel fet que hi pot haver disjuncions d'esports, és la relació entre els dies de la setmana amb el nombre de minuts de cada esport. Tornant amb l'exemple del ciclista, tenim la disponibilitat setmanal següent:

- *Dilluns*: Gimnàs o Agilitat (estiraments) o Ciclisme o Córrer, durant 2 hores.
- *Dimarts*: Gimnàs o Agilitat (estiraments) o Ciclisme o Córrer, durant 2 hores.
- *Dimecres*: Gimnàs o Agilitat (estiraments) o Ciclisme o Córrer, durant 2 hores.
- *Dijous*: Gimnàs o Agilitat (estiraments) o Ciclisme o Córrer, durant 2 hores.
- *Divendres*: Gimnàs o Agilitat (estiraments) o Ciclisme o Córrer, durant 3 hores.

- *Dissabte*: Ciclisme 4 hores.
- *Diumenge*: Ciclisme 4 hores.

Es pot observar les disjuncions d'esports en els dies d'entre setmana, on només es pot realitzar un dels quatre esports.

A partir de la informació anterior, el sistema crea les permutacions de totes les possibles disponibilitats setmanals, que en el cas del ciclista resulten en un total de 1024 possibles setmanes de disponibilitat diferents.

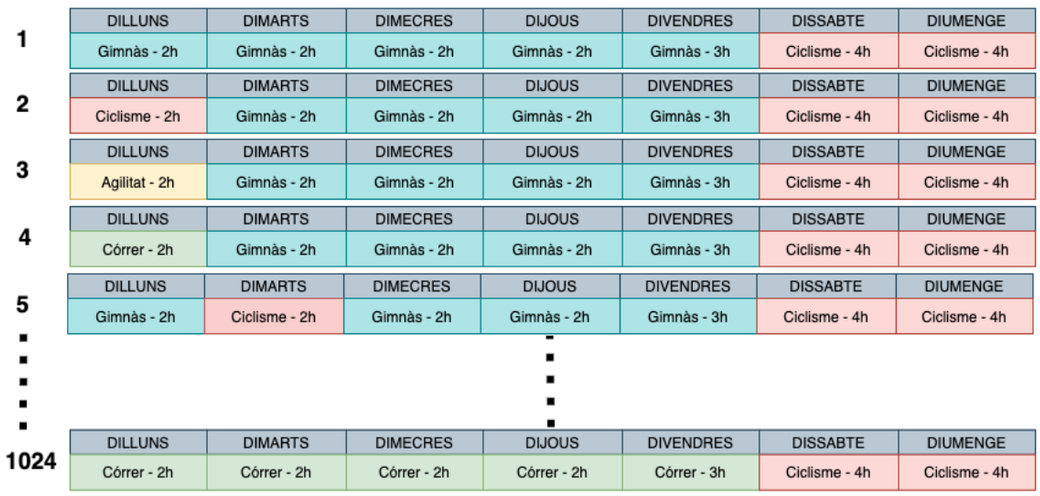


Figura 8.14: Possibles permutacions de setmanes de disponibilitat del ciclista de l'exemple.

8.3.5.4 Algorisme recursiu

El següent pas del servei ja és aplicar l'algorisme de recursivitat per generar les planificacions setmanals. En aquest punt, ja es disposa del conjunt de planificacions setmanals predefinides, del conjunt de setmanes de disponibilitat creades a partir de les normes i el sistema de puntuació de planificacions per saber quan de bona és una planificació respecte a una altra.

El procés recursiu segueix els passos següents:

1. Obtenir una setmana de disponibilitat.
2. Obtenir una planificació setmanal predefinida i extreure'n les sessions d'entrenament.
3. Provar tota la combinació possible de distribuir les sessions d'entrenament en la setmana de disponibilitat, si alguna solució és millor que el resultat, guardar-la.
4. Si queden planificacions setmanals anar a 2.
5. Si queden setmanes de disponibilitat anar a 1.
6. Fi.

Mitjançant la recursivitat podem comprovar totes la possibilitat de combinacions de cada planificació setmanal predefinida amb cadascuna de les setmanes de disponibilitat, i quedar-nos amb la que tingui una puntuació més alta. La puntuació ve donada pels paràmetres del sistema de puntuació.

8.3.5.5 Exemple generació planificació setmanal

Per entendre millor el procediment que segueix la implementació recursiva d'aquest servei, es complementarà l'explicació amb l'exemple del ciclista que té per objectiu realitzar una marxa cicloturista.

Del ciclista ja en disposem de les planificacions setmanals predefinides de la segona setmana de força, del conjunt de combinacions setmanals de disponibilitat (Figura 8.14) i el sistema de puntuació de planificacions següent:

- 10 punts per sessió assignada.
- 50 punts per mantenir diumenge descans.
- 20 punts per mantenir gimnàs els dilluns.
- 20 punts per mantenir agilitat els dilluns. Donat que els estiraments també es realitzen al gimnàs, també ho situem el dilluns.
- 50 punts per mantenir sessions de ciclisme els dimarts, dimecres, divendres o dissabte.
- 20 punts per mantenir sessions de córrer els dimarts, dimecres o divendres. Donat que entre setmana és indiferent sortir a córrer que amb bici, mantenim els dies.

El sistema de puntuació de planificacions està pensat perquè un expert esportiu o entrenador personal pugui ajustar els paràmetres necessaris dels atletes. A més a més, a mesura que l'atleta avanci en la temporada també s'anirà obtenint més dades i tendències que ajudaran a seguir ajustant-ho.

En el cas del ciclista, han estat un total de tres les planificacions setmanals predefinides obtingudes per generar la planificació de la segona setmana del període d'entrenament de velocitat pel ciclista:

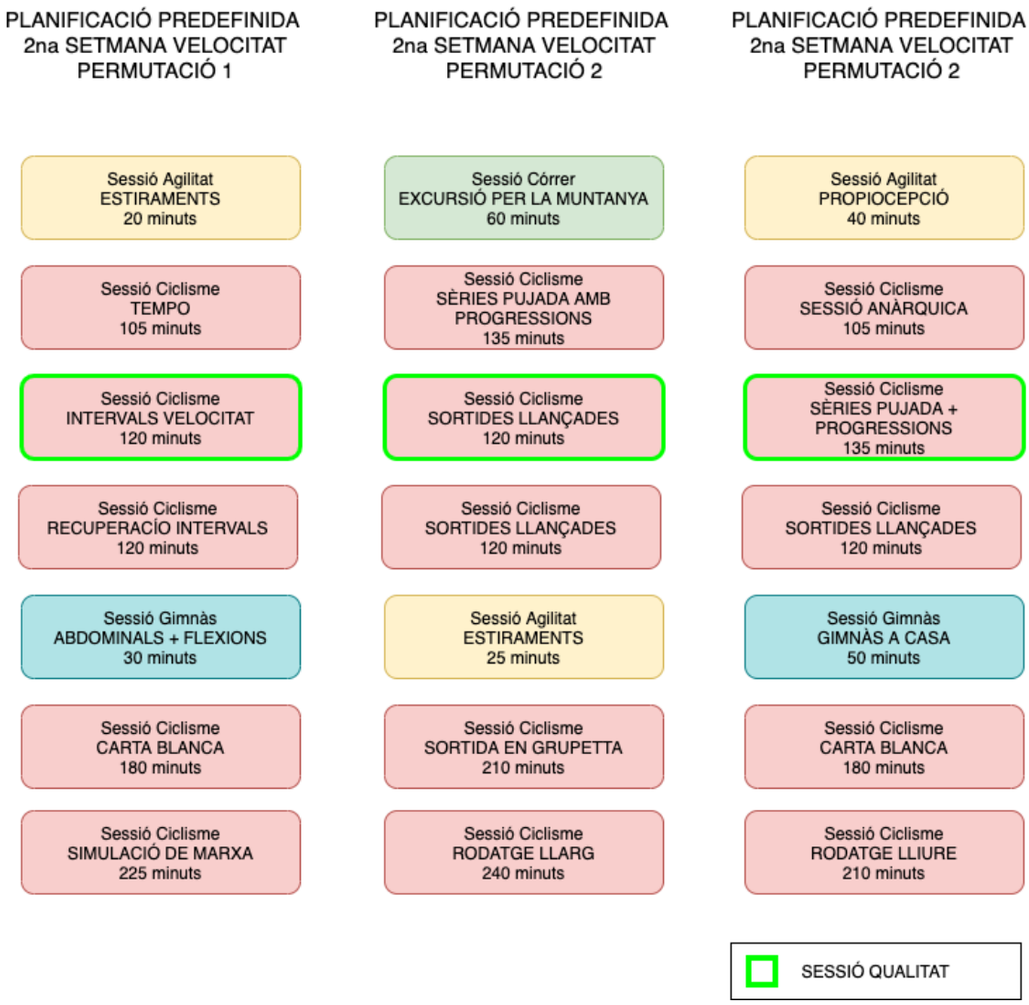


Figura 8.15: Conjunt planificacions predefinides 2na setmana velocitat, marxa cicloturista

Si s'analitza les tres planificacions setmanals predefinides per la segona setmana de velocitat del ciclista (Figura 8.15), es pot observar diferents característiques. Cadascuna de les planificacions setmanals predefinides té una sessió de qualitat per treballar la velocitat. La resta de sessions d'entrenament complementàries poden variar una mica, per exemple, incorporà sessions de córrer, sessions de gimnàs o fins i tot alguna sessió de ciclisme més llarga. La finalitat serà que si un atleta li agrada també sortir a córrer, el sistema disposi d'una planificació setmanal predefinida que contempli aquesta opció, o si un altre prefereix anar més el gimnàs, també tenir la possibilitat de contemplar-ho.

Donada la disponibilitat de l'usuari, el sistema de puntuacions definit prèviament i les planificacions setmanals predefinides, s'ha executat el servei pel ciclista i s'ha obtingut la següent planificació setmanal:

DILLUNS	DIMARTS	DIMECRES	DIJOUS	DIVENDRES	DISSABTE	DIUMENGE
ESTIRAMENTS 20 min	RECUPERACIÓ INTERVALS 120 min	INTERVALS VELOCITAT 120 min	ABDOMINALS + FLEXIONS 30 min	CARTA BLANCA 180 min	TEMPO 105 min	

Figura 8.16: Planificació setmanal obtinguda pel ciclista de l'exemple.

La planificació setmanal obtinguda pel ciclista, clarament ha estat dissenyada a partir de la permutació 1 de les planificacions setmanals predefinides. La planificació ha obtingut un total de **380 punts**, basant-se en el sistema de puntuació de

planificacions anterior.

En l'àmbit esportiu, la planificació setmanal del ciclista també té molta coherència. En primer lloc s'ha respectat el descans al diumenge i la setmana comença amb una sessió d'entrenament suau d'estiraments. Dimarts i dimecres s'han assignat les dues sessions específiques per treballar fort la velocitat, la segona de les quals és la sessió de qualitat. I per últim, s'ha reservat per divendres i dissabte les sessions d'entrenament més llargues en relació amb la disponibilitat de l'atleta.

8.3.5.6 Segon exemple generació planificació setmanal

Per últim en aquest capítol, i per tal d'acabar d'analitzar el comportament d'aquesta implementació, contemplarem un nou escenari pel ciclista de l'exemple. En aquest cas, suposarem que de tant en tant, al ciclista també li agrada sortir a córrer per la muntanya. Com a conseqüència, s'ajustarà el sistema de puntuació de planificacions per tal de tenir en compte aquest cas:

- 10 punts per sessió assignada.
- 50 punts per mantenir diumenge descans.
- 20 punts per mantenir gimnàs els dilluns.
- 20 punts per mantenir agilitat els dilluns. Donat que els estiraments també es realitzen al gimnàs, també ho situem el dilluns.
- 50 punts per mantenir sessions de ciclisme els dimarts, dimecres, divendres o dissabte.
- **25 punts** per mantenir sessions de córrer els dimarts, dimecres o divendres. Donat que entre setmana és indiferent sortir a córrer que amb bici, mantenim els dies.

Per completar l'exemple, també es modifica la disponibilitat del ciclista reduint la disponibilitat dels dissabtes i diumenges de la manera següent:

- *Dilluns*: Gimnàs o Agilitat (estiraments) o Ciclisme o Córrer, durant 2 hores.
- *Dimarts*: Gimnàs o Agilitat (estiraments) o Ciclisme o Córrer, durant 2 hores.
- *Dimecres*: Gimnàs o Agilitat (estiraments) o Ciclisme o Córrer, durant 2 hores.
- *Dijous*: Gimnàs o Agilitat (estiraments) o Ciclisme o Córrer, durant 2 hores.
- *Divendres*: Gimnàs o Agilitat (estiraments) o Ciclisme o Córrer, durant 3 hores.
- *Dissabte*: Ciclisme **2,5 hores.**
- *Diumenge*: Ciclisme **2,5 hores.**

Tenint en compte l'ajust realitzat en el sistema de puntuacions de planificacions, i també la reducció de la disponibilitat els dissabtes i diumenges, s'ha obtingut com a resultat la planificació setmanal següent:

DILLUNS	DIMARTS	DIMECRES	DIJOUS	DIVENDRES	DISSABTE	DIUMENGE
ESTIRAMENTS 25 min	SORTIDES LLANÇADES 120min	SORTIDES LLANÇADES 120 min		EXCURSIÓ PER LA MUNTANYA 60 min	SÈRIES PUJADA + PROGRESSIONS 135 min	

Figura 8.17: Planificació setmanal obtinguda pel ciclista de l'exemple (preferència córrer).

En aquest segon cas, la planificació setmanal obtinguda ha estat dissenyada pel sistema a partir de la permutació 2 del conjunt de planificacions predefinides. La puntuació obtinguda per la planificació setmanal resultat ha estat de **345 punts**.

Es pot observar que en l'àmbit esportiu, igual que en el cas anterior, també té molt sentit i s'ha respectat les preferències de sessions de córrer. La setmana comença suau amb una sessió d'estiraments. El dimarts i dimecres es manté les sessions de treball de velocitat, sent una d'elles la sessió de qualitat de la setmana. El divendres s'ha incorporat la sessió d'entrenament de córrer per la muntanya, respectant les prioritats de l'atleta. Finalment el dissabte s'ha programat la sessió de ciclisme més llarga aprofitant les dues hores i mitja de disponibilitat de ciclisme.

Per últim, analitzar el fet que el sistema de puntuació de planificacions és realment molt rellevant i que gran part del resultat està condicionat pels seus paràmetres.

8.3.5.7 Cost temporal generació de planificacions - implementació recursiva

El temps d'execució d'aquesta implementació, tal com es pot observar, és molt elevat. En els apartats següents es desglossa per cadascun dels dos exemples del ciclista anterior. L'anàlisi a fons del cost d'execució s'ha separat en un capítol a part, posterior al capítol de la segona implementació.

Cost permutacions de setmanes de disponibilitat

El primer pas consta de crear totes les possibles setmanes a partir de la disponibilitat de l'atleta. Si recordem en el cas del ciclista que es prepara per a una marxa cicloturista, hem obtingut un total de 1024 permutacions de setmanes de disponibilitat (vegeu Figura 8.14).

Generar les permutacions ha tingut un cost d'execució de **0.2179875373840332** segons en la primera setmana i **0.21312212944030762** segons en la segona setmana (preferència córrer).

Cost generar planificació setmanal

Per últim, el temps d'execució del conjunt de tots els passos d'aquest servei per generar la planificació setmanal personalitzada per la segona setmana de velocitat del ciclista, ha tingut uns temps d'execució de **514.7610321044922** segons en el primer cas, mentre que per la segona setmana de l'exemple el temps ha estat de **161.13166332244873** segons.

```
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash# python3 testRecursive.py
temps crear permutacions -> 0.2179875373840332
Number of permutations -> 1024

FINAL
{'2020-02-03': [[20, 7, 0, 0, 'ESTIRAMENTS']],
 '2020-02-04': [[120, 8, 3, 0, 'RECUPERACIÃ“  INTERVALS']],
 '2020-02-05': [[120, 8, 2, 1, 'INTERVALS  VELOCITAT']],
 '2020-02-06': [[30, 6, 4, 0, 'ABDOMINALS + FLEXIONS']],
 '2020-02-07': [[180, 8, 6, 0, 'CARTA  BLANCA']],
 '2020-02-08': [[105, 8, 1, 0, 'TEMPO']],
 '2020-02-09': []}
obj value -> 380
temps total -> 514.7610321044922
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash#
```

Figura 8.18: Execució terminal planificació setmanal exemple ciclista 1.

```
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash# python3 testRecursive.py
temps crear permutacions -> 0.21312212944030762
Number of permutations -> 1024

FINAL
{'2020-02-03': [[25, 7, 0, 0, 'ESTIRAMENTS']],
 '2020-02-04': [[120, 8, 2, 0, 'SORTIDES  LLANÃ‡ADES']],
 '2020-02-05': [[120, 8, 2, 1, 'SORTIDES  LLANÃ‡ADES']],
 '2020-02-06': [],
 '2020-02-07': [[60, 3, 4, 0, 'EXCURSIÃ“  PER  LA  MUNTANYA']],
 '2020-02-08': [[135, 8, 1, 0, 'SÃˆRIES PUJADA + PROGRESSIONS']],
 '2020-02-09': []}
obj value -> 345
temps total -> 161.13166332244873
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash#
```

Figura 8.19: Execució terminal planificació setmanal exemple ciclista 2.

Es pot observar la diferència entre els temps d'execució entre el primer i el segon exemple del ciclista. Aquesta diferència és causada pel fet de la reducció de la combinatòria de sessions d'entrenament del segon exemple. Cal recordar que en el segon exemple s'ha reduït la disponibilitat del dissabte i diumenge de quatre a dues hores i mitja. Aquest fet suposa que algunes de les sessions d'entrenament de les planificacions predefinides ja no hi càpiguen i per tant, la combinatòria de possibles distribucions és més petita.

8.3.5.8 Anàlisi final del cost

Pel que fa a l'anàlisi del cost algorítmic, és realment molt difícil de calcular per la gran quantitat de factors que intervenen:

- Diverses connexions a base de dades per obtenir informació de l'atleta, planificacions predefinides i sessions d'entrenament.
- Generar les permutacions de setmanes de disponibilitat, que depèn estrictament del nombre d'esports. És a dir, s'obtindrien tantes setmanes com:

  $$\text{Num. esports dilluns} * \text{Num. esports dimarts} * .... * \text{Num. esports diumenge}$$

  Per tant, si suposem que durant la setmana l'atleta pot practicar un número fixa d'esports igual a tots els dies, el resultat és:

  $$O(\text{generar permutacions disponibilitat}) = (\text{Nombre d'esports})^7$$

- Finalment, per cadascuna de les permutacions de disponibilitat i per cadascuna de les planificacions predefinides, obtenir les sessions d'entrenament d'aquesta última i comprovar totes les possibles maneres i ordres d'assignar les sessions d'entrenament. Aquest últim pas també inclou la possiblitat de no assignar-les totes.

El resultat acaba sent un cost exponencial, que depèn d'una gran quantitat de factors i que en el pitjor dels casos podria ser un procés interminable.

8.3.6 Implementació planificacions setmanals: sistema d'optimització

En aquest capítol s'explicarà el procediment i funcionament de la segona implementació del servei per generar les planificacions setmanals personalitzades. Igual que en la implementació anterior, l'explicació es realitzarà lligada amb l'exemple del ciclista que es prepara per a una marxa cicloturista. Això permetrà també que al final es pugui realitzar una comparació dels resultats i del temps d'execució.

Aquesta segona implementació s'ha desenvolupat a posteriori de la implementació recursiva del capítol anterior. Donats els temps d'execució desmesurats, prop de més de 600 segons per l'exemple del cicloturista, s'ha decidit replantejar el problema i per fer-ho s'ha tingut en compte la complexitat del problema a resoldre i quines tècniques hi ha al mercat per resoldre problemes d'aquesta mena. A més a més, un cost d'execució tan elevat fa que sigui inviable introduir aquest servei a la plataforma de l'empresa Trainerer, ja que es buscava escalabilitat i rendibilitat.

Per tal d'explorar possibles noves vies d'implementació per a resoldre el problema, és bàsic tornar a la idea que ens trobem davant d'un problema de complexitat NP-Difícil. En concret, generar planificacions setmanals personalitzades es troba al subgrup de problemes complexos de *Task Scheduling*. Aquest fet és molt rellevant perquè permet estudiar les solucions existents al mercat per a resoldre aquest tipus de problema. Avui en dia, un dels mètodes que obté millor resultat, per no dir el que més bons resultats obté per resoldre problemes molt costosos com aquest, és la utilització de *solvers* per a resoldre models d'optimització basats en aquest problema.

8.3.6.1 Procediment

El procediment d'aquesta segona implementació difereix bastant sobre la primera, a conseqüència de l'enfocament diferent utilitzant models d'optimització amb restriccions i resolts en el que es coneix com a solvers. Com és lògic, molts dels passos del procés de la primera implementació difereixen molt respecta aquesta segona. Com s'estudiarà a continuació, passos com llegir la disponibilitat de l'atleta tenen un enfocament completament diferent amb la idea de potenciar el màxim possible l'ús del sistema d'optimització.

Els passos que segueix el servei de generar planificacions setmanals personalitzades són:

1. Obtenir les planificacions setmanals predefinides que s'ajusten a la setmana de la temporada que passa l'atleta.
2. Llegir la disponibilitat de l'atleta en forma de minuts amb restriccions.
3. Crear el model d'optimització i seleccionar el *solver*.
   (a) Crear i definir les variables del model.
   (b) Crear i definir les restriccions del model.
   (c) Crear i definir la funció objectiu.
4. Executar el model utilitzant el *solver* seleccionat.
5. Llegir la planificació setmanal obtinguda del model.

Diagrama

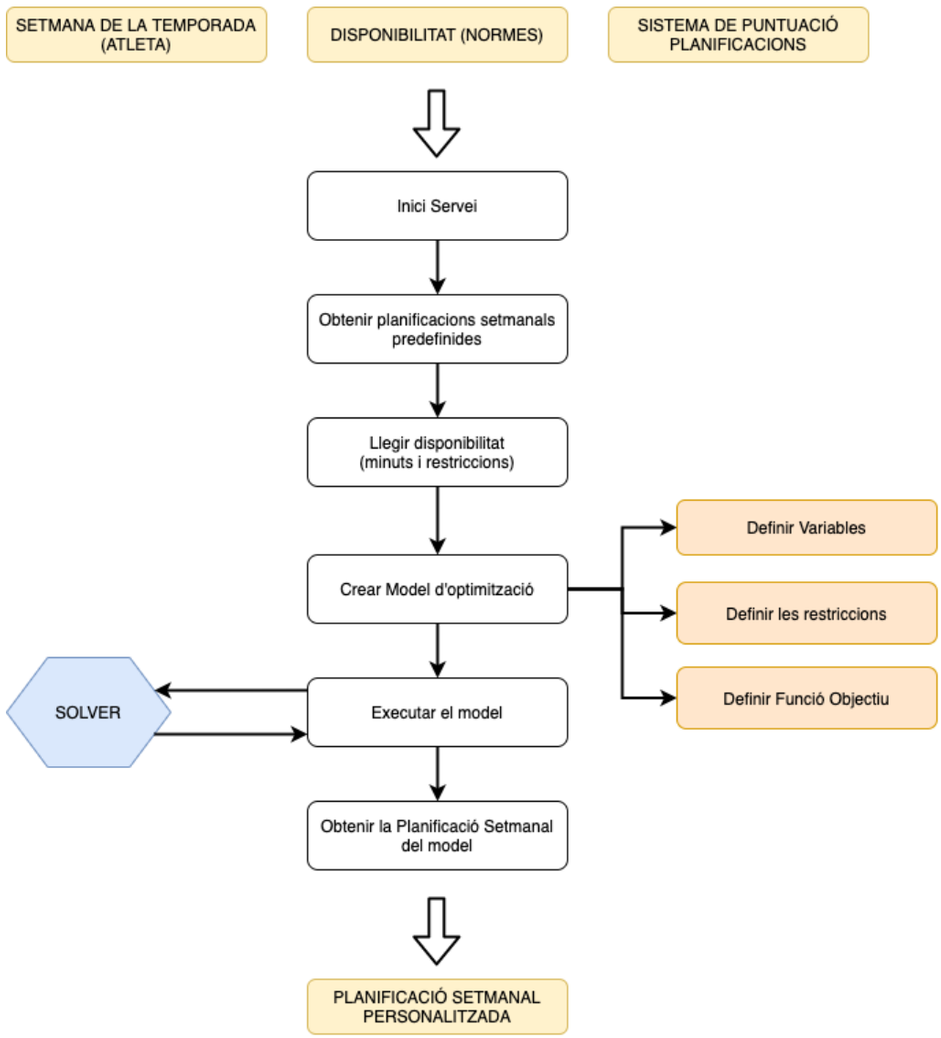


Figura 8.20: Diagrama de flux del procés de generar planificacions setmanals amb sistema d'optimització amb restriccions.

8.3.6.2 Disponibilitat

Un cop obtingut les planificacions setmanals predefinides pel moment de la temporada el qual l'atleta es troba, el següent pas consisteix a llegir la disponibilitat. Aquesta disponibilitat, també anomenada "regles"de disponibilitat pel fet que hi poden aparèixer disjuncions d'esports als dies, segueix un procés totalment diferent el de la implementació anterior en què s'utilitzava per crear permutacions de setmanes de disponibilitat. Si recordem la disponibilitat del ciclista de l'exemple:

- *Dilluns*: Gimnàs o Agilitat (estiraments) o Ciclisme o Córrer, durant 2 hores.
- *Dimarts*: Gimnàs o Agilitat (estiraments) o Ciclisme o Córrer, durant 2 hores.
- *Dimecres*: Gimnàs o Agilitat (estiraments) o Ciclisme o Córrer, durant 2 hores.
- *Dijous*: Gimnàs o Agilitat (estiraments) o Ciclisme o Córrer, durant 2 hores.
- *Divendres*: Gimnàs o Agilitat (estiraments) o Ciclisme o Córrer, durant 3 hores.
- *Dissabte*: Ciclisme 4 hores.

- *Diumenge*: Ciclisme 4 hores.

Com s'ha comentat anteriorment el procés que seguirà la disponibilitat és molt diferent el de la implementació del servei anterior. Si prèviament es transformava les "regles"de disponibilitat en tota la combinatòria de setmanes de disponibilitat (Figura 8.14), en aquest procés es guardarà en forma de minuts i restriccions. És a dir, si tenim que l'atleta el dilluns té disponibilitat per córrer o per ciclisme, si anteriorment es creaven dues setmanes diferents per cadascuna de les opcions, en aquesta implementació es guardarà que dilluns l'atleta té disponibilitat per córrer i ciclisme, però amb la restricció que només pot realitzar un dels dos esports.

Seguint el procés, la disponibilitat pel ciclista de l'exemple que s'obtindria seria la següent:

```
{
"dilluns": {Gimnas:120, Agilitat:120, Cilisme:120, Correr:120},
"dimarts": {Gimnas:120, Agilitat:120, Cilisme:120, Correr:120},
"dimecres": {Gimnas:120, Agilitat:120, Cilisme:120, Correr:120},
"dijous": {Gimnas:120, Agilitat:120, Cilisme:120, Correr:120},
"divendres":{Gimnas:180, Agilitat:180, Cilisme:180, Correr:180},
"dissabte": {Cilisme:240},
"diumenge": {Cilisme:240}
}
```

Figura 8.21: Disponibilitat del ciclista en forma de disponibilitat amb restriccions.

Tot i això, el ciclista només pot realitzar un esport cada dia. Per aquest motiu, el sistema també enregistra les restriccions següents:

```
{
"dilluns": [Gimnas, Agilitat, Cilisme, Correr],
"dimarts": [Gimnas, Agilitat, Cilisme, Correr],
"dimecres": [Gimnas, Agilitat, Cilisme, Correr],
"dijous": [Gimnas, Agilitat, Cilisme, Correr],
"divendres":[Gimnas, Agilitat, Cilisme, Correr],
"dissabte": [],
"diumenge": []
}
```

Figura 8.22: Restriccions de la disponibilitat del ciclista en forma de disponibilitat amb restriccions.

Les restriccions de la figura anterior (Figura 8.22) reflecteixen el nou procés de la disponibilitat per aquesta implementació. Per exemple, el dilluns el ciclista no pot realitzar simultàniament Gimnàs, Agilitat, Ciclisme i Córrer, i en comptes de crear quatre permutacions de setmanes més, s'estableix la restricció que al dilluns no es poden realitzar simultàniament aquests esports.

8.3.6.3 Eines

A l'hora de realitzar la implementació d'aquest servei per generar planificacions setmanals utilitzant models d'optimització, es fa imprescindible l'ús de noves eines compatibles amb Python per crear i executar models d'optimització en solvers. Actualment el mercat per Python, hi ha dues eines molt populars per resoldre aquest tipus de problemes mitjançant models d'optimització, per una banda OR-Tools de Google, i per l'altra Pyomo. Els dos apartats de continuació es realitza una comparativa d'aquestes dues eines i s'argumenta quina ha estat l'elecció.

OR-Tools

Google OR-Tools [22] és una eina de codi obert de Google orientada a l'optimització dels problemes més difícils de resoldre, com per exemple dissenyar rutes de vehicles, problemes de flux, problemes de distribució de tasques, resoldre sistemes lineals... Forma part del grup d'eines de Google AI [20] i ha estat reconegut amb diverses medalles al MiniZinc Challenge [35] els últims anys. La principal especialitat, i per la qual ha rebut els premis anteriors, de Google OR-Tools és la de resoldre models d'optimització de satisfacció de restriccions.

Aquesta eina permet crear el model d'optimització en Python, C++, DotNet i Java, per després ser executats en *solvers* comercials com CPLEX [28] o Gurobi [24], en *solvers* de codi obert com CBC [5] o Glpk [19], i en els *solvers* propis de Google com són GLOP [23] i CP-SAT [21].

Pyomo - Eina utilitzada

Pyomo [25] és la segona alternativa contemplada per creació i execució de models d'optimització en Python. Com es pot deduir del seu nom, és una eina exclusiva de Python, de codi lliure i orientada a la creació i execució de models d'optimització en *solvers*. És una eina molt semblant a OR-Tools de Google, però a part de ser exclusiva de Python, els models necessiten un *solver* extern. A diferència de l'eina anterior de Google que ja incorporava molts solvers, alguns de propis i altres d'externs, en la seva instal·lació.

S'ha decidit desenvolupar el model d'optimització amb restriccions utilitzant l'eina de Pyomo. A l'hora de realitzar la creació del model, és molt més simplificat utilitzant Pyomo que OR-Tools, potser pel fet que Pyomo és exclusiu de Python i més optimitzat en aquesta direcció. Per altra banda, Pyomo és una eina molt lleugera que permet anar-la completant instal·lant al sistema els solvers que es necessitin. Aquest punt també és rellavant, ja que en la línia de desenvolupar el microservei per l'empresa Trainerer, el cost d'emmegatzematge és important. Com a tercer punt, ja s'ha dit que Google OR-Tools és especialment interessant per resoldre models d'optimització de satisfacció de restriccions, ja que Google disposa de solver propis per a resoldre-ho. En el cas d'aquest servei, l'objectiu no és trobar una distribució de sessions d'entrenament que satisfaci unes restriccions, sinó que es busca la millor combinació i per tant, encara que s'utilitzés Google OR-Tools s'hauria d'utilitzar un solver extern.

8.3.6.4 Solvers

Al llarg d'aquest capítol s'ha utilitzat en nombroses vegades el terme "solver"sense especificar una definició clara, així doncs, un solver [54] és un programa o una llibreria que resolt problemes matemàtics. Són d'especial interès per resoldre problemes d'optimització costosos pel fet que estan entrenats per trobar solucions òptimes en temps d'execució molt més reduïts que utilitzant algorismes tradicionals. Els solvers més utilitzats, siguin de codi obert o privats, es troben llistats a continuació.

Gurobi

Gurobi [24] és un solver d'optimització per la resolució de problemes en programació lineal (LP), programació quadràtica (QP), programació quadràtica amb restriccions (QCP) i programació lineal entera (MILP). És el solver d'ús comercial més utilitzat

en problemes d'optimització. Una llicència individual per ús empresarial costa al voltant de 10,000 $.

S'ha descartat pel cost de la llicència.

CPLEX

La segona gran alternativa és el solver CPLEX [28] d'IBM. És especialment rellevant per resoldre problemes de programació entera i programació lineal. A diferència de Gurobi, CPLEX compte amb el seu propi entorn de desenvolupament, a part del solver. La llicència per desenvolupament més senzilla costa 192 € mensuals, és a dir, 2,300 € anuals.

Igual que en el cas del Gurobi, s'ha descartat per l'elevat cost de la llicència.

CBC

CBC [5] és un solver lliure de programació lineal i programació lineal entera desenvolupat en C++. Està basat en l'algorisme símplex [29] i és el més utilitzat per recerca i investigació. És possiblement un dels millors solvers de programari lliure.

És el que s'ha utilitzat finalment en el projecte, pel seu cost nul i per l'eficiència respecte al Glpk.

Glpk

Per últim el solver Glpk (GNU Linear Programming Kit) [19], és un solver per resoldre problemes de programació lineal, programació entera i algun d'altre. Utilitza una combinatòria de mètodes i algorismes, simplex methods, primal-dual interior-point method [43] i branch-and-cut method [55].

S'ha acabat descartant pel fet que CBC era més eficient en temps d'execució.

8.3.6.5 Sistema d'optimització

Arribat aquest punt, ja s'ha definit el concepte de solver i quin s'utilitzarà, s'ha definit també el procés de la disponibilitat i s'ha obtingut les planificacions setmanals predefinides. A continuació, el següent pas ja és definir el model d'optimització en si, és a dir, totes les seves variables, restriccions necessàries i per últim la funció objectiu a optimitzar.

En l'exemple del ciclista que es prepara per a una marxa cicloturista, ja s'ha obtingut la disponibilitat (Figura 8.21 i Figura 8.22), i les planificacions setmanals predefinides per la segona setmana de força que si es recorda són les següents:

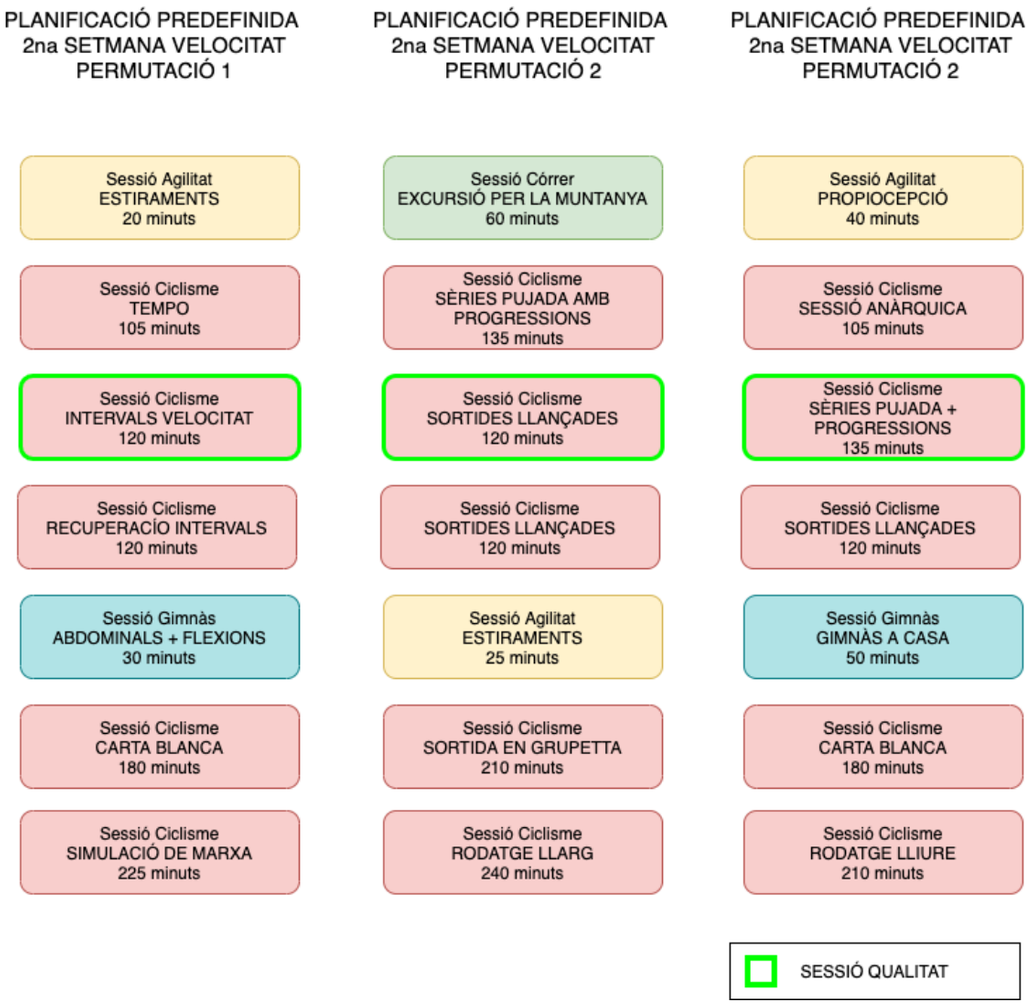


Figura 8.23: Conjunt planificacions predefinides 2na setmana velocitat, marxa cicloturista

Com ja s'ha comentat, un model d'optimització està format per un conjunt de variables que defineixen el domini del model, unes restriccions sobre aquestes variables que s'han de complir i per últim, una funció objectiu a optimitzar sobre el conjunt de variables anterior.

Variables

Com a model d'optimització, es necessita variables sobre les quals definir el problema. És el primer pas a l'hora de definir un model, ja que les restriccions i la funció objectiu, estan definides sobre les mateixes variables.

Les variables del model d'optimització s'han anat definint progressivament durant el desenvolupament per anar cobrint les necessitats que sorgien. La variable principal, com s'explicarà a continuació, és una matriu tridimensional booleana que relaciona les planificacions setmanals predefinides, amb les corresponents sessions d'entrenament i dies de la setmana. D'aquesta manera si algun valor de la matriu És *"True"*, significa que a un dia de la setmana en concret se li assigna una sessió d'entrenament que pertany a una planificació setmanal predefinida específica.

Observar que totes les variables s'han declarat en format booleà, pel fet que del model només es necessita saber si es produeixen o no algunes accions i un booleà és la unitat d'emmagatzematge mínim que accepta Pyomo.

Variable llista booleans: dia sessions d'entrenament. Llista de set booleans, un per cada dia de la setmana. L'objectiu d'aquesta variable és identificar els dies de la setmana que hi ha assignada una sessió d'entrenament.

Variable llista booleans: dia sessions de qualitat. Llista de set booleans, un per cada dia de la setmana. L'objectiu d'aquesta variable és identificar els dies de la setmana que hi ha assignada una sessió d'entrenament de qualitat.

Variable llista booleans: planificacions predefinida. Llista de booleans, una per cada planificació predefinida que hi hagi com a paràmetre d'entrada. L'objectiu d'aquesta variable és identificar quina de les planificacions setmanals predefinides s'utilitza.

Variable llista booleans: sessions ciclisme. Llista de set booleans, un per cada dia de la setmana. L'objectiu d'aquesta variable és identificar els dies de la setmana que hi ha assignada una sessió d'entrenament de ciclisme.

Variable llista booleans: sessions Gimnàs. Llista de set booleans, un per cada dia de la setmana. L'objectiu d'aquesta variable és identificar els dies de la setmana que hi ha assignada una sessió d'entrenament de gimnàs.

Variable llista booleans: sessions Agilitat. Llista de set booleans, un per cada dia de la setmana. L'objectiu d'aquesta variable és identificar els dies de la setmana que hi ha assignada una sessió d'entrenament d'agilitat.

Variable llista booleans: sessions Córrer. Llista de set booleans, un per cada dia de la setmana. L'objectiu d'aquesta variable és identificar els dies de la setmana que hi ha assignada una sessió d'entrenament de córrer.

Variable matriu booleans: relació planificació, dia, sessió d'entrenament. Variable més important del model, ja que identifica la planificació setmanal personalitzada del resultat. És una matriu booleana tridimensional, ja que identifica la relació de planificació setmanal predefinida, sessió d'entrenament i dia de la setmana. L'objectiu és identificar quines sessions d'entrenament de quina planificació setmanal predefinida, s'assignen a cada dia de la setmana.

Restriccions

El segon pas per desenvolupar el model d'optimització és definir les restriccions que afecten les variables anteriors. Es diferencien dos grups de restriccions, per una banda les restriccions a mantenir la coherència entre les variables definides, i les restriccions que defineixen les característiques d'una planificació setmanal esportiva i les seves funcionalitats. Per exemple, en el primer grup tenim una variable booleana de set elements per indicar si algun dia de la setmana s'ha assignat una sessió de ciclisme, i per tant es necessita una restricció que en assignar-se una sessió de ciclisme modifiqui aquesta variable booleana de set elements per indicar el dia de l'assignació. Per l'altra banda, el segon grup per definir una planificació setmanal, hi haurà una restricció del tipus que una sessió d'entrenament només pot ser assignada un cop.

A continuació es comença per definir les restriccions que s'utilitzaran per mantenir la coherència entre variables.

Restricció assignació d'una sessió a un dia en concret. Si a la matriu booleana tridimensional (planificació predefinida, sessió d'entrenament i dia de la setmana) hi ha alguna sessió d'entrenament assignada algun dia, és a dir, si hi ha alguna variable amb valor *"True"*, la variable de llista de booleans dels dies de la setmana ha de tenir el valor del dia corresponent a *"True"*. Per altra banda, si no hi

ha cap sessió assignada a un dia, és a dir, que la variable tridimensional que manté el resultat no té cap sessió assignada a un dia, aquest dia ha d'estar amb valor de *"False"* en la variable dels dies de la setmana.

Restricció assignació sessió de qualitat a un dia concret. Si a la matriu booleana tridimensional (planificació predefinida, sessió d'entrenament i dia de la setmana) hi ha alguna sessió d'entrenament de qualitat assignada algun dia, la variable de llista de booleans dels dies de la setmana sessió de qualitat, ha de tenir el valor del dia corresponent a *"True"*. Per altra banda, si no hi ha cap sessió de qualitat assignada a un dia, aquest dia ha d'estar amb valor de *"False"* en la variable dels dies de sessió de qualitat.

Restricció assignació sessió de ciclisme a un dia concret. Si a la matriu booleana tridimensional (planificació predefinida, sessió d'entrenament i dia de la setmana) hi ha alguna sessió d'entrenament de ciclisme assignada a un dia concret, la variable de llista booleana dels dies de sessions de ciclisme, també ha de tenir el dia de la sessió de ciclisme a *"True"*. Per altra banda, si algun dia no hi ha cap sessió de ciclisme assignada, la variable dels dies de sessions de ciclisme ha de tenir el dia a *"False"*.

Restricció assignació sessió de gimnàs a un dia concret. Si a la matriu booleana tridimensional (planificació predefinida, sessió d'entrenament i dia de la setmana) hi ha alguna sessió d'entrenament de gimnàs assignada a un dia concret, la variable de llista booleana dels dies de sessions de gimnàs, també ha de tenir el dia de la sessió de gimnàs a *"True"*. Per altra banda, si algun dia no hi ha cap sessió de gimnàs assignada, la variable dels dies de sessions de gimnàs ha de tenir el dia a *"False"*.

Restricció assignació sessió d'agilitat a un dia concret. Si a la matriu booleana tridimensional (planificació predefinida, sessió d'entrenament i dia de la setmana) hi ha alguna sessió d'entrenament d'agilitat assignada a un dia concret, la variable de llista booleana dels dies de sessions d'agilitat, també ha de tenir el dia de la sessió d'agilitat a *"True"*. Per altra banda, si algun dia no hi ha cap sessió d'agilitat assignada, la variable dels dies de sessions d'agilitat ha de tenir el dia a *"False"*.

Restricció assignació sessió de córrer a un dia concret. Si a la matriu booleana tridimensional (planificació predefinida, sessió d'entrenament i dia de la setmana) hi ha alguna sessió d'entrenament de córrer assignada a un dia concret, la variable de llista booleana dels dies de sessions de córrer, també ha de tenir el dia de la sessió de córrer a *"True"*. Per altra banda, si algun dia no hi ha cap sessió de córrer assignada, la variable dels dies de sessions de córrer ha de tenir el dia a *"False"*.

Per l'altra banda, tenim les restriccions pròpies del sistema que defineixen les propietats d'una planificació setmanal personalitzada.

Restricció assignació única de sessions d'entrenament. Restricció del sistema per assegurar que una sessió d'entrenament només pot estar assignada un cop. És a dir, a la matriu tridimensional booleana (planificació predefinida, sessió d'entrenament i dia de la setmana), una sessió d'entrenament d'una planificació predefinida només pot tenir un valor igual a *"True"* per cada dia de la setmana.

Restricció assignació de sessions d'entrenament amb disponibilitat. Restricció del sistema per assegurar que una sessió d'entrenament només s'assigna a un dia al qual l'atleta té disponibilitat de l'esport en minuts superior a la duració de la sessió. Per cada sessió d'entrenament el model comprova la disponibilitat en minuts del mateix esport que la sessió, i en cas que algun dia no es pugui

assignar, ja s'estableix el valor de *"False"* a la matriu tridimensional booleana per la sessió i dia.

Restricció una sola planificació setmanal predefinida. Restricció del sistema per assegurar que només s'assignen sessions d'entrenament d'una planificació setmanal predefinida. A la matriu tridimensional booleana d'assignació de sessions, no hi pot haver dues sessions assignades (valor *"True"* algun dia de la setmana) que siguin de planificacions setmanals predefinides diferents.

Restricció esports disponibilitat. Restriccions del sistema per assegurar que no s'assignen dues sessions d'esports diferents a un mateix dia, quan només se'n pot realitzar un d'ells (restriccions de disponibilitat). Per exemple, en cas que un dia la disponibilitat sigui de 120 minuts de ciclisme o 120 minuts córrer, es definirà una restricció que estableixi que en aquell dia es pot realitzar sessions de ciclisme o sessions de córrer, però no les dues al mateix temps.

Funció objectiu

L'últim pas per desenvolupar el sistema d'optimització consta de la definició de la funció objectiu, que en el cas d'aquest projecte el solver haurà de maximitzar. La funció objectiu s'utilitza per a valorar una planificació setmanal, i com és lògic, estableix un criteri per seleccionar la millor. Un dels paràmetres d'entrada d'aquest servei és el sistema de puntuació de planificacions, un sistema que defineix a la funció objectiu el criteri de puntuació de les planificacions.

Si es recorda en l'exemple del ciclista que es prepara per a una marxa cicloturista, el sistema de puntuació de planificacions que s'havia dissenyat era el següent:

- Punts per sessió assignada.
- Punts per dies concrets de descans.
- Punts per assignar la sessió de qualitat.
- Punts per assignar sessió de qualitat dimecres o dijous.
- Punts per mantenir patró de ciclisme setmana anterior.
- Punts per mantenir patró de córrer setmana anterior.
- Punts per mantenir patró de gimnàs setmana anterior.
- Punts per mantenir patró d'agilitat/estiraments setmana anterior.

La funció objectiu fa ús de les variables del model per obtenir informació sobre les sessions d'entrenament assignades, i d'aquesta manera aplicar el sistema de puntuació de planificacions per saber quan de bona és la solució. El solver per la seva part, el seu objectiu és maximitzar el valor que s'obté de la funció objectiu amb la millor combinatòria de les variables del model definides, i respectant sempre les restriccions. El solver finalitza l'execució quan ha arribat a una disposició de les variables del model que maximitza el sistema de puntuació de planificacions, és a dir, quan ha trobat la millor planificació setmanal personalitzada.

8.3.6.6 Exemple generació planificació setmanal

Per complementar l'explicació de la creació del model d'optimització, se seguirà amb l'exemple del ciclista que es prepara per a la marxa cicloturista. Com a recordatori, les dades que disposa el sistema en el moment previ a la creació de la planificació personalitzada són les següents:

- 3 planificacions setmanals predefinides (Figura 8.23)
- Disponibilitat:

```
{
    "dilluns":  {Gimnas:120, Agilitat:120, Cilisme:120, Correr:120}
                                          ,
    "dimarts":  {Gimnas:120, Agilitat:120, Cilisme:120, Correr:120}
                                          ,
    "dimecres": {Gimnas:120, Agilitat:120, Cilisme:120, Correr:120}
                                          ,
    "dijous":   {Gimnas:120, Agilitat:120, Cilisme:120, Correr:120}
                                          ,
    "divendres":{Gimnas:180, Agilitat:180, Cilisme:180, Correr:180}
                                          ,
    "dissabte": {Cilisme:240},
    "diumenge": {Cilisme:240}
}
```

Figura 8.24: Disponibilitat del ciclista en forma de disponibilitat amb restriccions.

- Restriccions disponibilitat:

```
{
    "dilluns":  [Gimnas, Agilitat, Cilisme, Correr],
    "dimarts":  [Gimnas, Agilitat, Cilisme, Correr],
    "dimecres": [Gimnas, Agilitat, Cilisme, Correr],
    "dijous":   [Gimnas, Agilitat, Cilisme, Correr],
    "divendres":[Gimnas, Agilitat, Cilisme, Correr],
    "dissabte": [],
    "diumenge": []
}
```

Figura 8.25: Restriccions de la disponibilitat del ciclista en forma de disponibilitat amb restriccions.

El primer pas per crear el model d'optimització consta en definir la variable més important, la matriu tridimensional que relaciona les sessions d'entrenament de les planificacions predefinides amb l'assignació als dies de la setmana.

Matriu assignació sessions

Com s'ha definit prèviament, la matriu tridimensional booleana relaciona les planificacions setmanals predefinides, les sessions d'entrenament de cadascuna d'elles i els dies de la setmana. En el cas d'aquest exemple, en la primera dimensió tenim les 3 planificacions setmanals predefinides, en la segona dimensió 7 sessions d'entrenament per cada planificació anterior, i finalment en l'última dimensió els 3 dies de la setmana. Per tant, el procés resulta en una matriu booleana 3x7x7.

Això significa que si algun valor de la matriu és *"True"*, una sessió d'entrenament d'una planificació setmanal predefinida ha sigut assignada algun dia de la setmana.

Restriccions del sistema

- Cadascuna de les sessions d'entrenament de les 3 planificacions predefinides només pot estar assignada un sol cop.
- Només es poden assignar sessions d'entrenament d'una sola planificació setmanal predefinida.
- Només es pot assignar una sessió d'entrenament a un dia que l'atleta pugui practicar l'esport de la sessió, i la disponibilitat sigui més gran o igual que la duració de la sessió.
- Per cadascuna de les restriccions de disponibilitat (vegeu Figura 8.25), assegurar-se que per cada dia no es realitzin simultàniament els esports de les restriccions.

Restriccions coherència entre variables

- Si durant l'execució del model en el solver, s'assigna alguna sessió d'entrenament d'alguna planificació predefinida a un dia en concret, la variable de dies de la setmana tindrà el dia marcat també com a *"True"*.
- Si durant l'execució del model en el solver, s'assigna alguna sessió d'entrenament de qualitat d'alguna planificació predefinida a un dia en concret, la variable de dies de la setmana de sessions de qualitat tindrà el dia marcat també com a *"True"*.
- si durant l'execució del model en el solver, s'assigna alguna sessió d'entrenament, la variable llista de l'esport de la sessió d'entrenament, es marcarà amb *"True"* el dia que s'hagi assignat la sessió. En l'exemple tenim l'esport de ciclisme, córrer, gimnàs i agilitat (estiraments o relaxació muscular).

Creació Funció Objectiu a maximitzar

De la manera que s'ha explicat abans, la funció objectiu s'utilitza per valorar quan de bona és una planificació setmanal. L'objectiu del solver és trobar una combinació de valors de les variables explicades anteriorment, que maximitzi la funció objectiu. Com a resultat s'obtindrà la millor planificació setmanal basant-se en el sistema de puntuació:

- 10 punts per sessió assignada.
- 50 punts per mantenir diumenge descans.
- 20 punts per mantenir gimnàs els dilluns.
- 20 punts per mantenir agilitat els dilluns. Donat que els estiraments també es realitzen al gimnàs, també ho situem el dilluns.
- 50 punts per mantenir sessions de ciclisme els dimarts, dimecres, divendres o dissabte.
- 20 punts per mantenir sessions de córrer els dimarts, dimecres o divendres. Donat que entre setmana és indiferent sortir a córrer que amb bici, mantenim els dies.

Obtenció del resultats

El solver és l'encarregat d'executar el model d'optimització fins a trobar una distribució de les variables que maximitzi la funció objectiu, és a dir, fins a trobar una assignació de sessions d'entrenament que obtinguin una puntuació màxima en el sistema de puntuació de planificacions. El resultat obtingut és la configuració de variables del model en si, i per tant, s'ha de llegir aquestes variables per obtenir-ne la planificació setmanal personalitzada.

Resultats obtinguts amb el solver

Donada la disponibilitat de l'usuari, el sistema de puntuacions definit prèviament i les planificacions setmanals predefinides, l'execució del sistema d'optimització en el solver ha tingut com a resultat la següent planificació setmanal:

DILLUNS	DIMARTS	DIMECRES	DIJOUS	DIVENDRES	DISSABTE	DIUMENGE
GIMNÀS A CASA 50 min	SESSIÓ ANÀRQUICA 120 min	SORTIDES LLANÇADES 120 min	PROPIOCEPCIÓ 40 min	SÈRIES PUJADA + PROGRESSIONS 135 min	RODATGE LLIURE 210 min	

Figura 8.26: Planificació setmanal obtinguda utilitzant el solver per resoldre el sistema d'optimització.

La planificació ha obtingut un total de **380 punts**, a través de la maximització de la funció objectiu. Pertany a la planificació predefinida amb permutació 3 (Figura 8.23).

8.3.6.7 Segon exemple generació planificació setmanal

De la mateixa manera que s'ha procedit amb la primera implementació, es realitza una modificació de l'exemple del ciclista per tal d'analitzar la implementació una mica més a fons. Per aquest segon exemple, es modificarà una mica el sistema de puntuació de planificacions i la disponibilitat de l'atleta.

Igual que en el cas de la implementació recursiva, es modifica el sistema de puntuació de planificacions de la manera següent:

- 10 punts per sessió assignada.
- 50 punts per mantenir diumenge descans.
- 20 punts per mantenir gimnàs els dilluns.
- 20 punts per mantenir agilitat els dilluns. Donat que els estiraments també es realitzen al gimnàs, també ho situem el dilluns.
- 50 punts per mantenir sessions de ciclisme els **dimarts, dimecres o dissabte**.
- **25 punts** per mantenir sessions de córrer els dimarts, dimecres o divendres. Donat que entre setmana és indiferent sortir a córrer que amb bici, mantenim els dies.

Igual que en el cas de la implementació recursiva, es modifica la disponibilitat de ciclisme de dissabtes i diumenges de l'atleta:

- *Dilluns*: Gimnàs o Agilitat (estiraments) o Ciclisme o Córrer, durant 2 hores.
- *Dimarts*: Gimnàs o Agilitat (estiraments) o Ciclisme o Córrer, durant 2 hores.
- *Dimecres*: Gimnàs o Agilitat (estiraments) o Ciclisme o Córrer, durant 2 hores.
- *Dijous*: Gimnàs o Agilitat (estiraments) o Ciclisme o Córrer, durant 2 hores.
- *Divendres*: Gimnàs o Agilitat (estiraments) o Ciclisme o Córrer, durant 3 hores.
- *Dissabte*: Ciclisme **2.5 hores.**
- *Diumenge*: Ciclisme **2.5 hores.**

Si s'executa per segon cop el model, tenint en compte les modificacions anteriors, la planificació setmanal obtinguda és la següent:

DILLUNS	DIMARTS	DIMECRES	DIJOUS	DIVENDRES	DISSABTE	DIUMENGE
ESTIRAMENTS 25 min	SORTIDES LLANÇADES 120min	SORTIDES LLANÇADES 120 min		EXCURSIÓ PER LA MUNTANYA 60 min	SÈRIES PUJADA + PROGRESSIONS 135 min	

Figura 8.27: Planificació setmanal obtinguda utilitzant el solver per resoldre el sistema d'optimització, exemple 2.

La planificació anterior ha obtingut **345 punts** amb el sistema de puntuació de planificacions de la funció objectiu. Si observem el resultat, veiem que a diferència de la planificació anterior tenim una sessió d'entrenament de córrer. Aquests canvis han vingut causats a conseqüència de la modificació del sistema de puntuacions de planificacions, ja que s'ha donat 25 punts per sessió de córrer que s'assigni a un dimarts, dimecres o divendres. A més a més, s'ha eliminat el divendres els punts de sessió de ciclisme.

Si comparem la planificació setmanal obtinguda en el primer exemple amb l'obtinguda en el segon, podem observar que en cas que fossin dos atletes diferents, tot i preparar-se pel mateix objectiu i està el mateix moment de la temporada, no tindrien la mateixa planificació setmanal. És un dels grans avantatges que té la utilització d'un sistema de puntuació de planificacions que s'adapta a les necessitats i preferències dels atletes.

8.3.6.8 Cost temporal generació de planificacions - implementació sistema d'optmització

Com a últim apartat dins del capítol de la segona implementació per generar planificacions setmanals, s'analitzarà els temps d'execució obtinguts en ambdós exemples anteriors.

En el primer cas (Figura 8.26), el temps per llegir la disponibilitat setmanal i crear les restriccions d'esports ha estat de **0.00016260147094726562 segons**, i el sistema d'optimització ha trobat una solució que maximitza la funció objectiu en un total de **1.5885634422302246 segons**.

```
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash# python3 testNewSolver.py
Processing availability time -> 0.00016260147094726562
RESULTS:

objective value -> 380.0
Planning -> 2
sessions 4 in day 0 of planning 2  ------ [50, 6, 4, 0, True, 'GIMNÃ€S A CASA']
sessions 1 in day 1 of planning 2  ------ [120, 8, 2, 0, False, 'SESSIÃ“  ANÃ€RQUICA']
sessions 5 in day 2 of planning 2  ------ [120, 8, 3, 1, False, 'SORTIDES  LLANÃ‡ADES']
sessions 0 in day 3 of planning 2  ------ [40, 7, 0, 0, True, 'PROPIOCEPCIÃ“']
sessions 3 in day 4 of planning 2  ------ [135, 8, 1, 0, False, 'SÃˆRIES PUJADA + PROGRESSIONS']
sessions 6 in day 5 of planning 2  ------ [210, 8, 6, 0, False, 'RODATGE  LLIURE']
Execution time: 1.5885634422302246
```

Figura 8.28: Sortida per terminal execució solver per l'exemple 1 del ciclista per la marxa cicloturista.

Per l'altra banda, el segon exemple al qual s'ha suposat que el ciclista tenia interès per sessions d'entrenament de córrer (Figura 8.27), el temps per llegir la disponibilitat setmanal i crear les restriccions d'esports ha estat de **0.00018453598022460938 segons**. El sistema d'optimització ha trobat una solució que maximitza la funció objectiu en un total de **1.6186678409576416 segons**.

```
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash# python3 testNewSolver.py
Processing availability time -> 0.00018453598022460938
RESULTS:

objective value -> 345.0
Planning -> 1
sessions 2 in day 0 of planning 1  ------ [25, 7, 0, 0, True, 'ESTIRAMENTS']
sessions 6 in day 1 of planning 1  ------ [120, 8, 2, 0, False, 'SORTIDES  LLANÃ‡ADES']
sessions 5 in day 2 of planning 1  ------ [120, 8, 2, 1, False, 'SORTIDES  LLANÃ‡ADES']
sessions 0 in day 4 of planning 1  ------ [60, 3, 4, 0, False, 'EXCURSIÃ“  PER  LA  MUNTANYA']
sessions 3 in day 5 of planning 1  ------ [135, 8, 1, 0, False, 'SÃˆRIES PUJADA + PROGRESSIONS']
Execution time: 1.6186678409576416
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash#
```

Figura 8.29: Sortida per terminal execució solver per l'exemple 2 del ciclista per la marxa cicloturista.

8.3.6.9 Anàlisi final del cost

En aquest cas, es fa impossible realitzar un anàlisi algorítmic del servei de generació de planificacions setmanals utilitzant el model d'optimització i el solver. Tota la part del solver i la creació del model d'optimització són paquets de programari externs desenvolupat per organitzacions o persones externes a aquest projecte i per tant no es pot valorar el cost algorítmic.

Avaluació del sistema generació planificacions

En aquest capítol s'analitzarà en detall els resultats que s'obtenen utilitzant cadascuna de les dues implementacions per generar planificacions d'entrenament setmanals personalitzades. En els últims apartats del capítol, es realitzarà una comparació del temps d'execució i un test d'estrès [53] per tal d'observar el comportament en ambdues implementacions quan es tenen moltes planificacions setmanals predefinides o disponibilitats més complexes.

9.1 Avaluació dels resultats

A continuació es realitzarà una comparativa dels resultats obtinguts en l'exemple del ciclista que es prepara per a la marxa cicloturista utilitzant les dues implementacions.

9.1.1 Múltiples possibles solucions

En el primer cas, amb l'algorisme recursiu i amb el sistema d'optimització s'ha obtingut setmanes diferents, com es pot veure a la Figura 9.1.

Algorisme recursiu - Puntuació: 380

DILLUNS	DIMARTS	DIMECRES	DIJOUS	DIVENDRES	DISSABTE	DIUMENGE
ESTIRAMENTS 20 min	RECUPERACIÓ INTERVALS 120 min	INTERVALS VELOCITAT 120 min	ABDOMINALS + FLEXIONS 30 min	CARTA BLANCA 180 min	TEMPO 105 min	

Sistema d'optimització - Puntació: 380

DILLUNS	DIMARTS	DIMECRES	DIJOUS	DIVENDRES	DISSABTE	DIUMENGE
GIMNÀS A CASA 50 min	SESSIÓ ANÀRQUICA 120 min	SORTIDES LLANÇADES 120 min	PROPIOCEPCIÓ 40 min	SÈRIES PUJADA + PROGRESSIONS 135 min	RODATGE LLIURE 210 min	

Figura 9.1: Comparativa planificació setmanal obtinguda exemple 1 amb l'algorisme recursiu i amb el sistema d'optimització.

Les planificacions setmanals personalitzades tenen sessions d'entrenament diferents, però per l'altra banda segueixen quasi el mateix patró d'esports. Els dilluns comencen amb una sessió de gimnàs o agilitat, dimarts una sessió de ciclisme, dimecres la sessió de qualitat de ciclisme, dijous una sessió de gimnàs o agilitat, i divendres i dissabte les sessions de ciclisme més llargues.

Les dues solucions són correctes i resolen el problema de generar una planificació setmanal personalitzada, ja que les dues planificacions obtenen la màxima puntuació basant-se en el sistema de puntuació de planificacions. En el cas de l'algorisme recursiu, està implementat per tal de quedar-se amb la primera de les múltiples solucions. En el cas del sistema d'optimització resolt amb un solver, no podem saber quina de les solucions obté, però sabem que la solució obtinguda maximitza la funció objectiu i per tant obté la màxima puntuació basant-se en el sistema de puntuació de planificacions.

Des del punt de vista d'aquest projecte, les dues solucions serien vàlides. Vist des del punt de vista esportiu també té sentit, en les dues setmanes hi ha quatre sessions d'entrenament de ciclisme per treballar la força, de les quals una és la sessió de qualitat. També tenim una sessió de gimnàs i una sessió d'agilitat que complementen les quatre sessions de ciclisme. Per tant, les dues planificacions tenen la sessió de qualitat al mig de la setmana, les sessions de ciclisme més llargues al divendres i dissabte, les sessions de gimnàs i agilitat per complementar la setmana, i per últim el diumenge descans.

9.1.2 Solució única

En el segon cas, el segon exemple d'ambdues implementacions, ha obtingut com a resultat la mateixa planificació setmanal personalitzada (Figura 9.2).

Algorisme recursiu - Puntuació: 345

DILLUNS	DIMARTS	DIMECRES	DIJOUS	DIVENDRES	DISSABTE	DIUMENGE
ESTIRAMENTS 25 min	SORTIDES LLANÇADES 120min	SORTIDES LLANÇADES 120 min		EXCURSIÓ PER LA MUNTANYA 60 min	SÈRIES PUJADA + PROGRESSIONS 135 min	

Sistema d'optimització - Puntació: 345

DILLUNS	DIMARTS	DIMECRES	DIJOUS	DIVENDRES	DISSABTE	DIUMENGE
ESTIRAMENTS 25 min	SORTIDES LLANÇADES 120min	SORTIDES LLANÇADES 120 min		EXCURSIÓ PER LA MUNTANYA 60 min	SÈRIES PUJADA + PROGRESSIONS 135 min	

Figura 9.2: Comparativa planificació setmanal obtinguda exemple 2 amb l'algorisme recursiu i amb el sistema d'optimització.

Com es pot observar les dues implementacions han arribat a la mateixa solució tot i la gran diferència en temps d'execució. En aquest segon exemple doncs, hi ha una solució única que maximitza la puntuació obtinguda i per aquest motiu totes dues implementacions han obtingut el mateix resultat.

Com veurem en el següent apartat, la diferència de temps entre la primera implementació i la segona és molt gran. Tot i això, la implementació recursiva per generar planificacions setmanals pot ser útil a l'hora de verificar que el sistema d'optimització (segona implementació) està funcionant correctament.

9.2 Avaluació del temps d'execució

A continuació s'avaluarà el temps d'execució dels exemples del ciclista que es prepara per a una marxa cicloturista. Es finalitzarà amb un anàlisi més complex per tal d'analitzar els temps d'execució en ambdós sistemes.

En totes les proves que es realitzaran a continuació, estan basades en l'exemple del ciclista que es prepara per a una marxa cicloturista. L'exemple en si, utilitzat per explicar la implementació del projecte, ja era molt complex en disponibilitat i en el sistema de puntuació de les planificacions.

9.2.1 Avaluació temps d'execució exemple cicloturista

En aquest primer apartat s'avaluarà els temps d'execució obtinguts en els dos exemples del ciclista utilitzant les implementacions, la recursiva i el model d'optimització. Els temps d'execució obtinguts es mostren a la taula següent:

	Exemple 1		Exemple 2	
	Algorisme recursiu	Model d'optmització	Algorisme recursiu	Model d'optmització
Número permutacions disponibilitat	1024	-	1024	-
Número planificacions predefinides	3	-	3	3
Temps d'execució	514.761032	1.588563	161.131663	1.618667

Figura 9.3: Taula temps d'execució exemple cicloturista

Com ja s'ha explicat al detall en el seu corresponent capítol, l'algorisme recursiu utilitza permutacions de setmanes de disponibilitat per comprovar totes les possibles combinacions, mentre que el model d'optimització utilitza restriccions d'esports. Podem observar que els temps d'execució de l'algorisme recursiu estan molt lluny del model executat en un solver i reflecteix la complexitat del problema a resoldre. També dóna credibilitat a la idea d'utilitzar models d'optimització amb restriccions que s'executin en solvers, és una bona implementació per resoldre problemes de *Task Scheduling*.

També es pot observar com en els dos exemples, el temps d'execució en el model d'optimització utilitzant un solver per resoldre-ho no ha variat pràcticament. Per l'altra banda, utilitzant l'algorisme recursiu s'ha reduït quasi per quatre vegades. La gran reducció del temps d'execució ve donat pel fet que en el segon exemple s'havia reduït la disponibilitat de ciclisme dels dissabtes i diumenges de quatre hores a dos i mitja. Això suposa que la combinatòria de setmanes a comprovar per l'algorisme recursiu no sigui tan gran, ja que no es poden assignar les sessions de ciclisme més llargues de les planificacions predefinides.

Després de l'anàlisi i experimentació utilitzant ambdues implementacions, és fàcil veure que hi ha dos grans factors que afecten el temps d'execució: la disponibilitat setmanal dels atletes i el nombre de planificacions predefinides. En el següent apartat, es realitzaran proves d'estrès par tal d'observar el comportament del sistema d'optimització i l'algorisme recursiu davant d'un increment d'aquests dos factors tan rellevants.

9.2.2 Avaluació temps d'execució en funció de la disponibilitat

En aquest apartat s'estudia la relació entre la disponibilitat setmanal de l'atleta i el temps d'execució en cadascuna de les dues implementacions. S'ha utilitzat el mateix exemple del ciclista que es prepara per a una marxa cicloturista i que pot realitzar 4 esports diaris.

La disponibilitat anirà augmentan des de 30 minuts diaris per un dels quatre esports, fins arribar a 240 minuts (4 hores) a una escala de 30 minuts.

9.2.2.1 Implementació recursiva

El temps d'execució en funció de la disponibilitat en la implementació recursiva ha obtingut els resultats següents:

Temps diari (minuts	Temps d'execució (segons)
30	4.55883526802063
60	6.540103912353516
90	6.304154872894287
120	105.5096206665039
150	196.28121733665466
180	377.770446062088
210	715.8507051467896
240	1347.317682504654

Figura 9.4: Taula evaluació del temps d'execució en funció de la disponibilitat. Implementació recursiva.

Si s'observen els resultats anteriors en forma de gràfica:

Figura 9.5: Gràfica evaluació del temps d'execució en funció de la disponibilitat. Implementació recursiva.

Es pot observar la tendència exponencial i lo realment costós que és el problema a resoldre quan la disponibilitat de l'atleta augmenta.

9.2.2.2 Implementació sistema d'optimització

El temps d'execució en funció de la disponibilitat en la implementació utilitzant un sistema optimització amb restriccions executat en solvers ha obtingut els resultats

següents:

Temps diari (minuts	Temps d'execució (segons)
30	1.6024277210235596
60	2.012683629989624
90	1.918635368347168
120	1.8944292068481445
150	1.9150919914245605
180	1.8879997730255127
210	2.1039202213287354
240	2.031836986541748

Figura 9.6: Taula evaluació del temps d'execució en funció de la disponibilitat. Implementació model d'optmització.

Si s'observen els resultats anteriors en forma de gràfica:

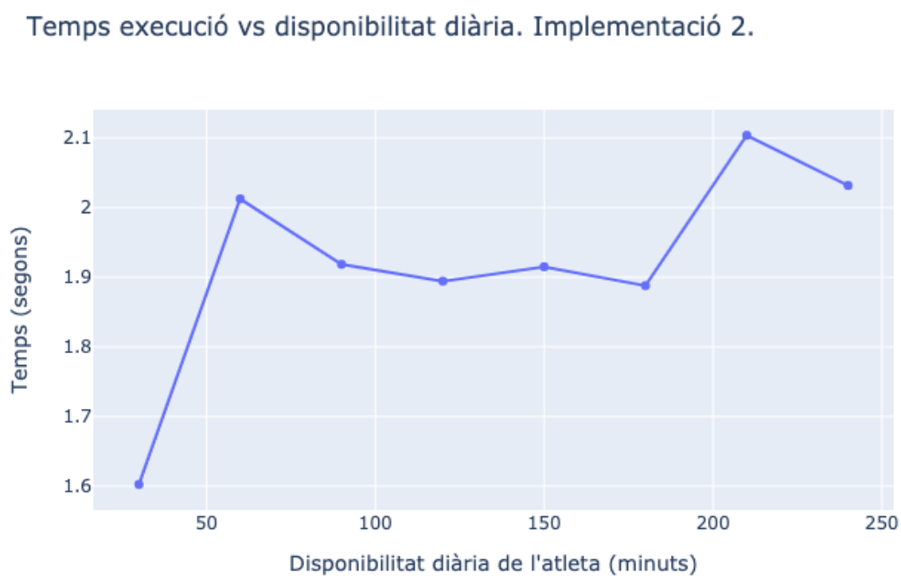


Figura 9.7: Gràfica evaluació del temps d'execució en funció de la disponibilitat. Implementació model d'optimització.

En aquest cas, tal com es pot observar en la gràfica, la disponibilitat de l'atleta no té un efecte trascendent en el temps d'execució del model d'optmització. Si ens fixem, gran part dels valors es troben entre 2.1 i 1.9 segons.

9.2.3 Avaluació temps d'execució en funció del nombre de planificacions predefinides

En aquest apartat s'estudia la relació entre el nombre de planificacions predefinides, és a dir el nombre de sessions d'entrenament, i el temps d'execució de cadascuna de les implementacions.

9.2.3.1 Implmentació recursiva

Els temps d'execució obtinguts en la implementació recursiva per generar les planificacions setmanals personalitzades, en funció del nombre de planificacions setmanals predefinides ha estat:

Nombre de planificacions setmanals predefinides	Temps d'execució (segons)
1	311.09212493896484
2	565.6994073390961
3	578.6686291694641
4	678.0229954719543
5	836.4802956581116
6	930.1142101287842
7	1046.8166780471802
8	1144.5462653636932
9	1214.0364747257233
10	1579.4103796482086
11	1634.0669090747833
12	1641.4639601707458

Figura 9.8: Taula evelució del temps en funció del número de planificacions setmanal predefinides. Implementació recursiva.

Si observem els resultats anteriors obtinguts en una gràfica, es pot veure la tendència clara d'augment del temps d'execució a mesura que s'incrementen el nombre de planificacions setmanals predefinides.

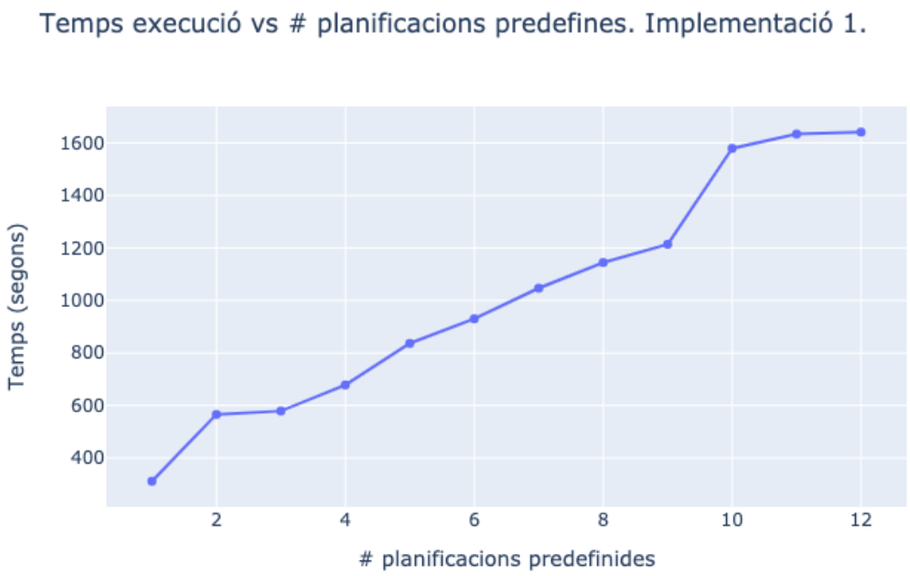


Figura 9.9: Gràfica relació temps d'execució i nombre planificacions predefinides. Implementació recursiva.

L'augment en el temps és clarament lineal. El resultat és l'esperat ja que afegir una planificació setmanal més en l'algorisme recursiu significa duplicar el problema a resoldre i quedar-se amb la millor planificació.

9.2.3.2 Implementació model d'optimització

Els temps d'execució obtinguts en la implementació utilitzant el model d'optimització per generar les planificacions setmanals personalitzades, en funció del nombre de planificacions setmanals predefinides:

Nombre de planificacions setmanals predefinides	Temps d'execució (segons)
1	1.6667015552520752
2	1.6960971355438232
3	1.6783747673034668
4	1.797853708267212
5	1.890709400177002
6	2.0540966987609863
7	2.3360326290130615
8	2.662358283996582
9	2.7656757831573486
10	2.775928497314453
11	3.015747308731079
12	3.483834981918335

Figura 9.10: Taula evelució del temps en funció del número de planificacions setmanal predefinides. Implementació sistema optimització.

Finalment, si ho observem en forma de gràfica, es pot apreciar la tendència a l'augment del temps d'execució a mesura que s'incrementa les planificacions setmanals predefinides. Figura 9.11.

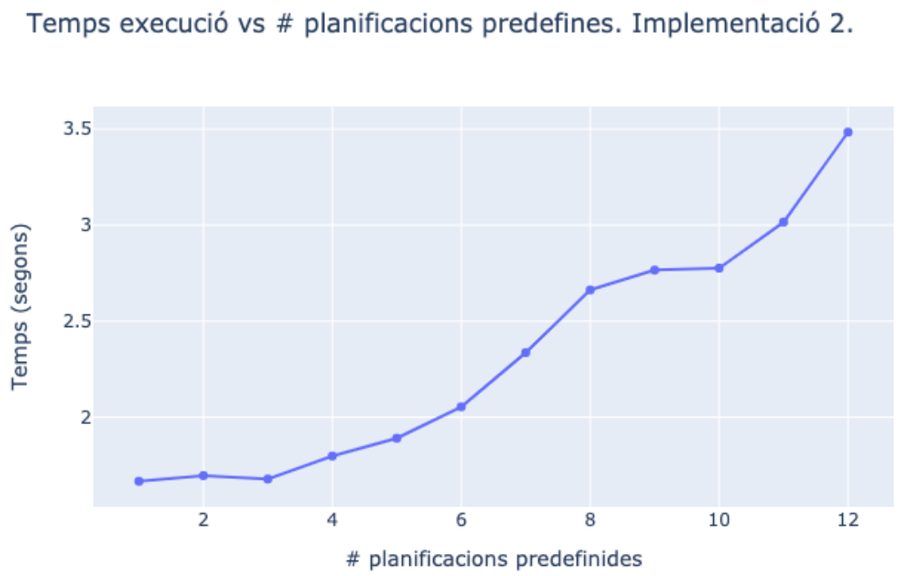


Figura 9.11: Gràfica relació temps d'execució i nombre planificacions predefinides. Implementació sistema optimització.

Es pot observar que en una disponibilitat i sistema de puntuació de planificacions complexes, fins a sis planificacions predefinides no se supera els 2 segons d'execució. A partir de les set planificacions setmanals predefinides el temps es dispara a l'alça notablement.

Per tant, en el cas del sistema d'optimització, augmentar el nombre de planificacions setmanals predefinides si que té un efecte directe amb el temps d'execució.

Sostenibilitat i compromis social

10.1 Sostenibilitat econòmica

Per la realització del projecte s'ha realitzat un pressupost on s'han tingut en compte els recursos humans, els recursos del hardware, els recursos del software, els costos indirectes, els imprevistos i les contingències. Com s'ha pogut observar en l'apartat de pressupostos corresponent, el factor més gran ha estat els recursos humans. Per la part del hardware i software també s'ha tingut en compte l'amortització, basada en la vida útil material definida per Hisenda.

Els costs del projecte són molt raonables donada la dimensió i complexitat del projecte. També remarcar que els costos indirectes s'han vist àmpliament reduïts gràcies al fet que el desenvolupament del projecte es realitza en un co-working en comptes d'una oficina. Els recursos també han estat mínims i es creu que no es podria desenvolupar el projecte en un període de temps més reduït. Per aquests fets es conclou que el pressupost s'ajusta molt a les necessitats de l'empresa.

10.2 Sostenibilitat social

Aquest projecte ha col·laborat en l'aprenentatge tant en la part tècnica de desenvolupar microserveis en Python, com en la part d'organització i planificació de projectes.

Pel que fa a la societat, cada cop hi ha persones amb més sobrepès i sobretot a les grans ciutats, es fa difícil la pràctica d'esport regular. Aquest projecte intentarà aportar a la societat una eina més per totes aquelles persones que tenen la necessitat o les ganes de realitzar esport, i ho volen fer de manera guiada, progressiva i sense lesions. Per aquesta raó, es creu que el projecte millorarà la qualitat de vida del col·lectiu descrit anteriorment. El col·lectiu d'entrenadors personals es podria veure perjudicat pel desenvolupament del projecte, ja que s'estaria substituint la seva feina.

10.3 Sostenibilitat ambiental

Per la realització del projecte s'ha utilitzat els recursos materials descrits en l'apartat anterior de recursos materials, els quals són mínims. Es necessita de l'electricitat pel funcionament de l'ordinador i les pantalles, i en un futur, l'electricitat necessària per fer funcionar un petit servidor. A més a més, l'ordinador es troba en el co-working del canòdrom de Barcelona, un espai compartit que intenta reduïr l'impacte mediambiental.

A més a més, el correcte desenvolupament d'aquest projecte pot afavorir el mediambient donat que s'està promocionant la realització d'esport en col·lectius amb alguna dificultat per realitzar-ne. D'aquesta manera es podrien substituir activitats més contaminants per entrenaments d'esport a l'aire lliure.

Conclusions

Arribat aquest punt de la memòria, es dóna per completada tota l'explicació i anàlisi del projecte *Aplicatiu de Creació Automàtica de Planificacions d'Entrenament Personalitzat*. L'objectiu d'aquest últim capítol és en primer lloc analitzar els resultats que s'han obtingut amb el desenvolupament d'aquest projecte. A continuació, un petit anàlisi de la retrospectiva i quines possibles ampliacions es poden realitzar en un futur. Finalment, quin ha estat en conjunt l'aprenentatge que s'ha adquirit en el desenvolupament del projecte.

11.1 Resultat

En les fases inicials d'aquest projecte, es va detallar tot el conjunt de funcionalitats que s'havia d'assolir per tal de portar un producte nou i diferent el mercat. Com s'ha explicat molt bé en la memòria del projecte, al llarg del desenvolupament s'han trobat diferents obstacles i inconvenients, tot i això, es pot afirmar que el projecte ha estat tot un èxit.

Generar planificacions setmanals no ha estat gens fàcil, a conseqüència de la complexitat del problema a resoldre. Tot i això, la millora en la segona implementació ha estat molt gran i permet que l'empresa Trainerer pugui començar els testos reals a la seva plataforma. Per altra banda, també hi ha hagut el repte de treballar amb dades físiques i fisiològiques del cos humà. El cos humà és un món per si sol, i tot i que es pot quantificar és difícil saber quin afecte tindrà una sessió d'entrenament o una altra. Per últim, la pandèmia ha dificultat molt el desenvolupament. Cal recordar que aquest és un projecte d'un treball de fi de grau universitari amb el qual es pretenia un seguiment proper del tutor a l'empresa i el ponent de la universitat. En aquest sentit el confinament ha afectat aquest seguiment, però s'han buscat alternatives i s'ha tirat endavant.

Tot i els inconvenients que han sorgit en el desenvolupament d'aquest projecte, el més rellevant és que s'ha seguit endavant per acabar complint els terminis previstos. A continuació es llisten els resultats que s'han aconseguit del projecte:

- Crear i desenvolupar el projecte en forma de microservei per tal de que sigui escalable, fiable i segur.
- El projecte sigui compatible amb la plataforma actual de l'empresa Trainerer.
- Dissenyar i implementar un servei dins del projecte per dissenyar temporades esportives en funció de l'objectiu principal de l'atleta i el nombre de setmanes de les quals disposa per preparar-se.
- Poder gestionar la disponibilitat dels atletes setmana rere setmana dins d'una temporada.

- Dissenyar i implementar un servei dins del projecte per generar planificacions setmanals personalitzades als atletes al llarg de la temporada esportiva.
- El punt anterior inclou poder dissenyar un sistema per quantificar quan de bona és una planificació per cadascun dels atletes del sistema.

Com s'ha dit, es pot afirmar que el desenvolupament ha estat un èxit pel fet que s'ha complert l'objectiu principal de dissenyar un sistema per generar planificacions setmanals personalitzades en forma de microservei de Python.

11.2 Futur

Tot i l'èxit del desenvolupament del projecte, hi ha dues grans extensions que s'han tingut presents dins de l'empresa Trainerer des dels inicis. Aquestes extensions millorarien notablement la qualitat de les planificacions setmanals que generaria el sistema, però al mateix temps també suposen grans reptes en l'àmbit tecnològic i esportiu.

11.2.1 Generar temporades modificables d'acord amb l'estat i objectiu de l'atleta.

La primera extensió del projecte està enfocada a generar una temporada tenint en compte l'estat de l'atleta. Com ja s'ha analitzat, actualment es disposa de plantilles de temporades predefinides pels objectius dels atletes per generar les temporades. El que pretén aquesta extensió és quantificar l'estat de l'atleta a l'inici, quantificar l'objectiu amb les mateixes unitats que l'atleta i d'aquesta manera poder generar una temporada per l'atleta acord amb les seves mancances envers l'objectiu.

Aquesta extensió obre una nova porta al projecte, ja que incloure l'estat de l'atleta a l'hora de generar temporades seria un gran pas pels atletes. A més a més, també hi hauria la possibilitat de realitzar un seguiment al llarg de la temporada per tal d'analitzar com de bé o malament està progressant l'atleta. Per últim, també permetria modificar la temporada a mesura que l'atleta està avançant, ja que si per assolir l'objectiu esportiu l'atleta en un moment donat necessita treballar més un aspecte físic que un altre, es modificaria la temporada per destinar més setmanes a entrenar l'aspecte físic que necessita.

Dos atletes de la mateixa edat que es volen preparar per al mateix objectiu esportiu, poden tenir estats físics molt diferents. Aquest fet fa que la temporada per preparar-se per l'objectiu tingui sentit que sigui diferent per tots dos. És en aquest aspecte en el qual aquesta extensió pretén millorar el projecte actual.

11.2.2 Utilitzar models de predicció en la generació de planificacions setmanals

Aquesta segona extensió del projecte és posterior a l'anterior, ja que requereix la possibilitat de poder quantificar l'estat de l'atleta. La idea general consta que si el sistema és capaç de quantificar l'estat d'un atleta, es pot saber quin afecte ha tingut sobre aquest una sessió d'entrenament o fins i tot una planificació setmanal.

L'objectiu seria poder crear un sistema intel·ligent que a mesura que els atletes avancen al llarg de les seves temporades, va aprenent quines han estat les millors

sessions d'entrenament per complir cadascun dels objectius. D'aquesta manera, les planificacions esportives que es realitzin per a futurs atletes seran molt més precises i s'adequaran millor a l'estat, nivell i temporada de l'atleta.

11.3 Aprenantatge final

A part dels grans resultats obtinguts, el desenvolupament d'aquest projecte també ha significat un gran aprentantatge. En primer lloc, recordar que la primera de les competències tècniques que es va establir quan es va presentar el treball va ser la competència **CCO1.1**, aquesta diu: *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.* Com s'ha vist en el capítol d'implementació, hi havia un problema realment molt costós de resoldre, però després d'estudiar-lo i analitzar-lo, s'ha sapigut trobar una molt bona estratègia per la seva implementació.

En segon lloc, també s'han adquirit molt conneixents alhora d'escriure codi net, reusable i eficient en temps d'execució. Donada l'embargadura d'aquest projecte, ha estat molt important que totes les parts del codi estiguessin molt ben optimitzades.

En tercer lloc, s'ha posat en pràctica el desenvolupament de sistemes d'optimització amb restriccions per resoldre problemes complexos. Per l'altra banda, i també com s'havia vist en alguna de les assignatures de l'especialitat de computació del grau, s'ha utilitzat solvers entrenats per resoldre els models d'optimització.

Per últim, s'han aplicat coneixements que potser no sent especialitats de la branca de computació, també són necessàries en el desenvolupament d'un producte software i que estan presents la feina d'un Enginyer Informàtic. Aquests coneixements són la gestió de connexions asíncrones entre diferents parts del projecte, disseny i arquitectura del sistema, disseny i arquitectura de la base de dades, i arquitectura en l'àmbit de programació utilitzant patrons de disseny.

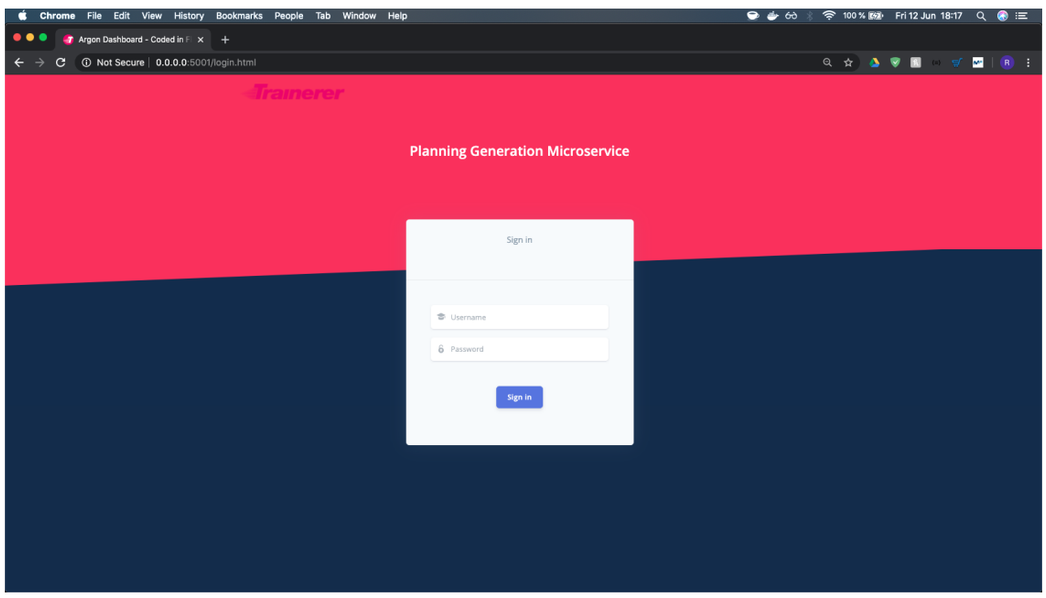
Apèndix

12.1 Codi del projecte

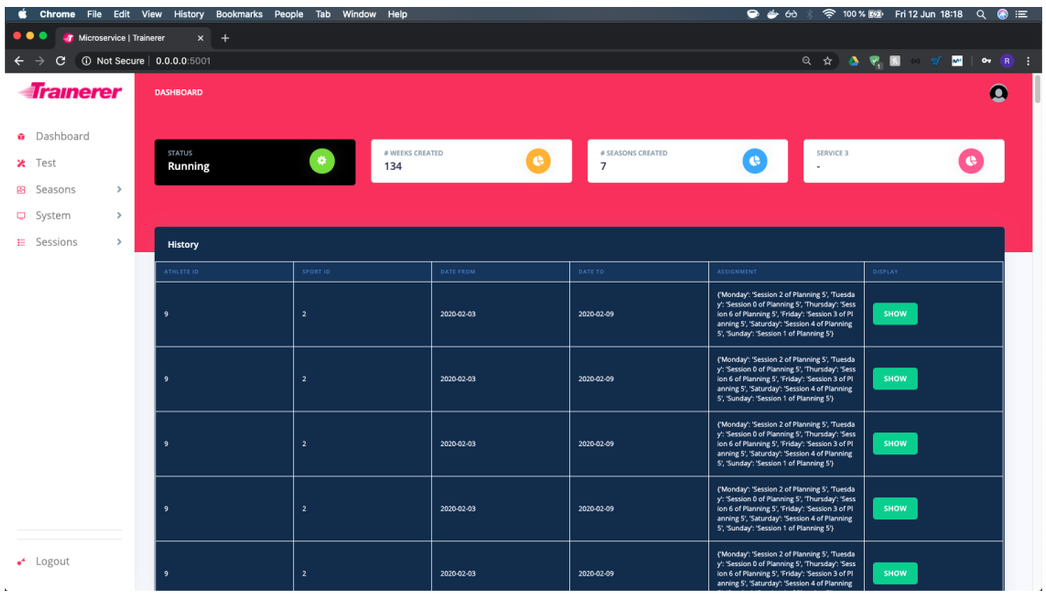
Junt amb la memòria està inclòs el codi del projecte. El fitxer *README.md* detalla la instal·lació de totes les parts. És imprescindible tenir tot el programari, dependències i dades a la base de dades per funcionar correctament.

12.2 Interfície

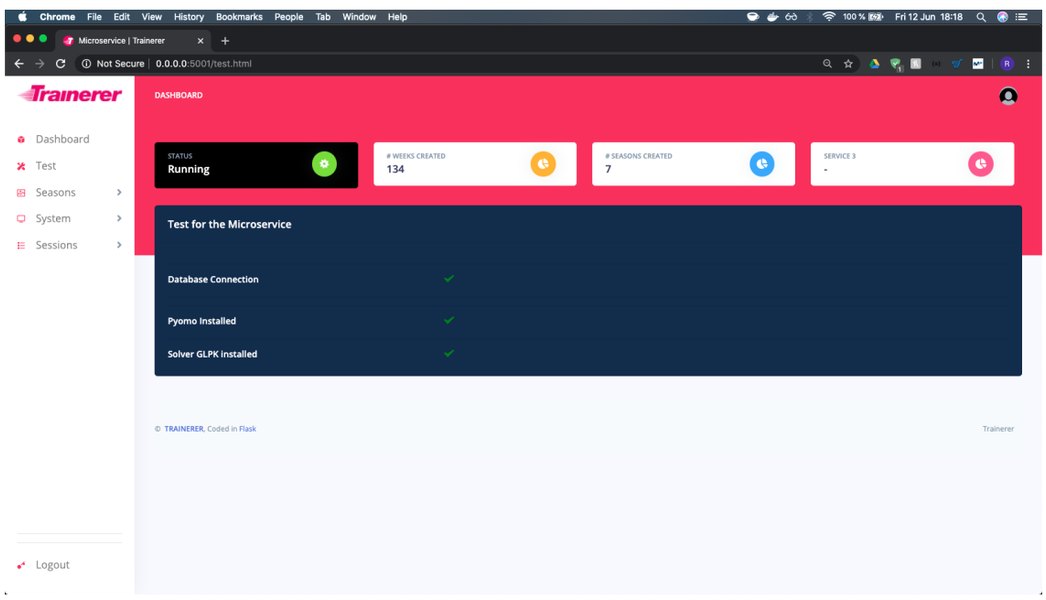
Login



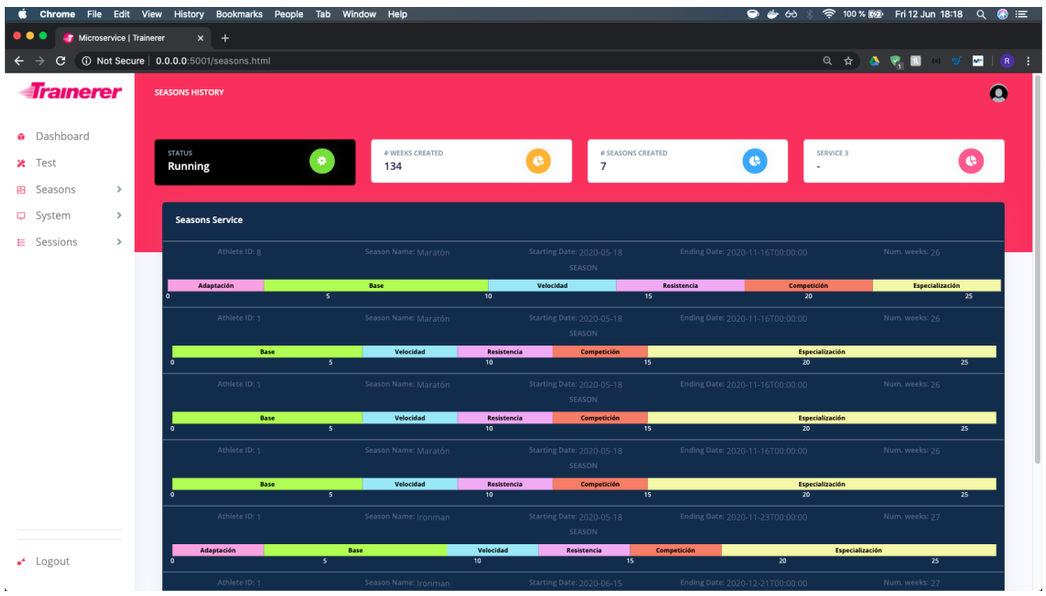
Planificacions setmanals creades



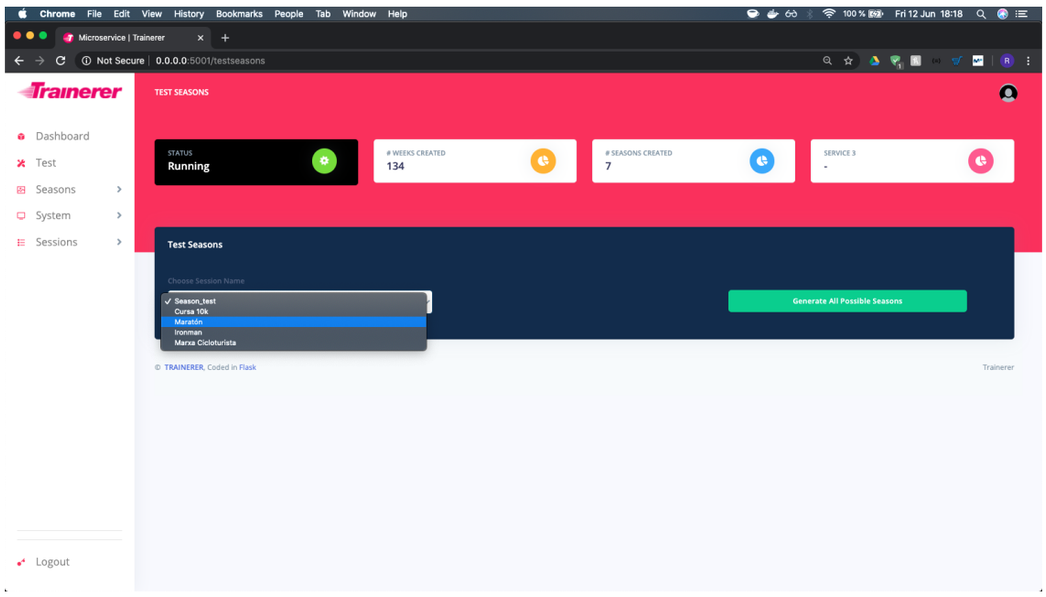
Comprovació dependències projecte



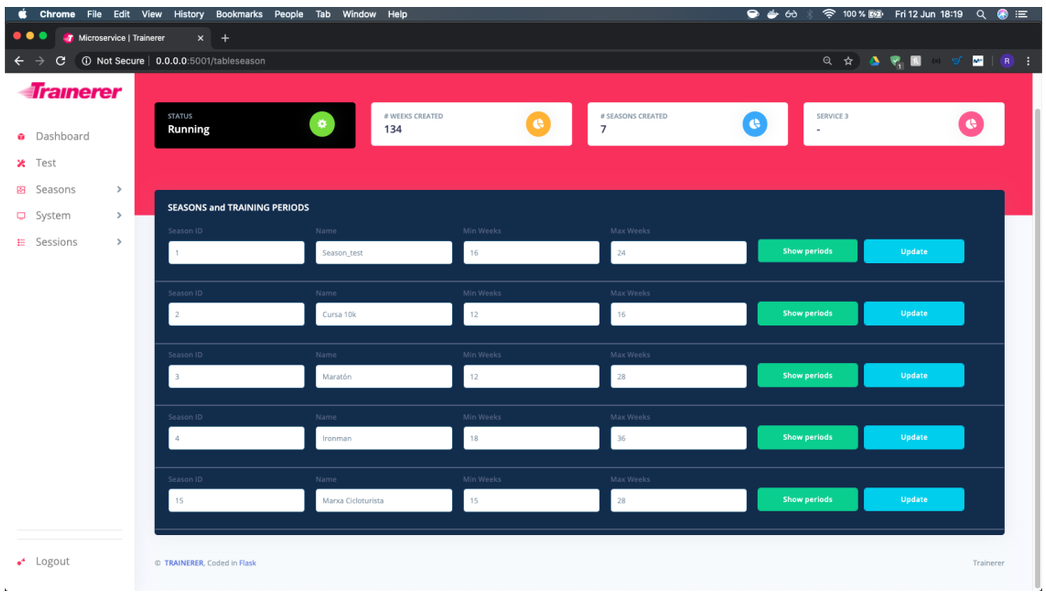
Temporades creades



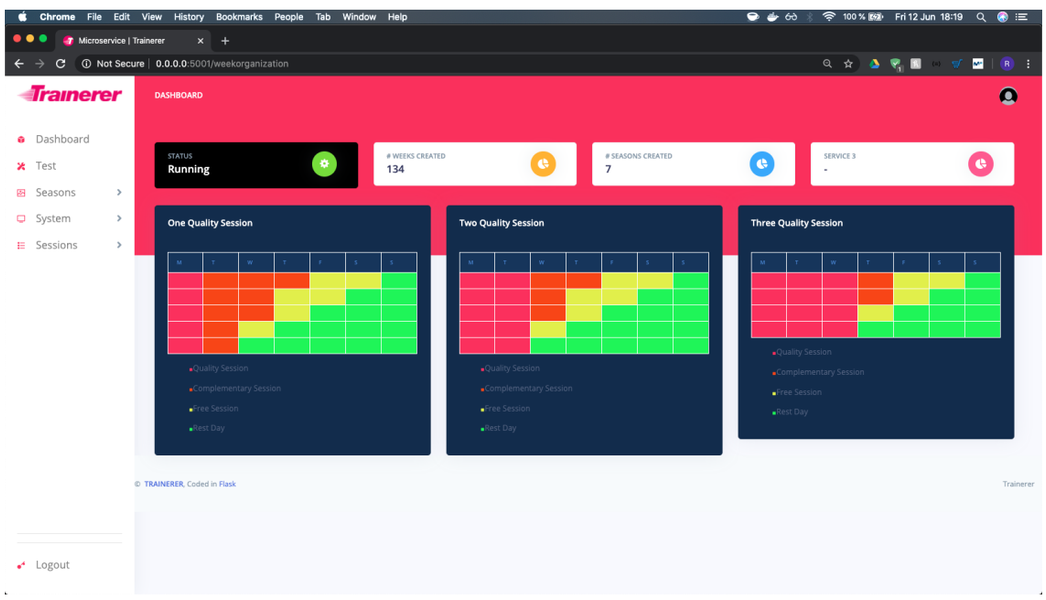
Generar permutacions (entre min. i max. de setmanes) temporades



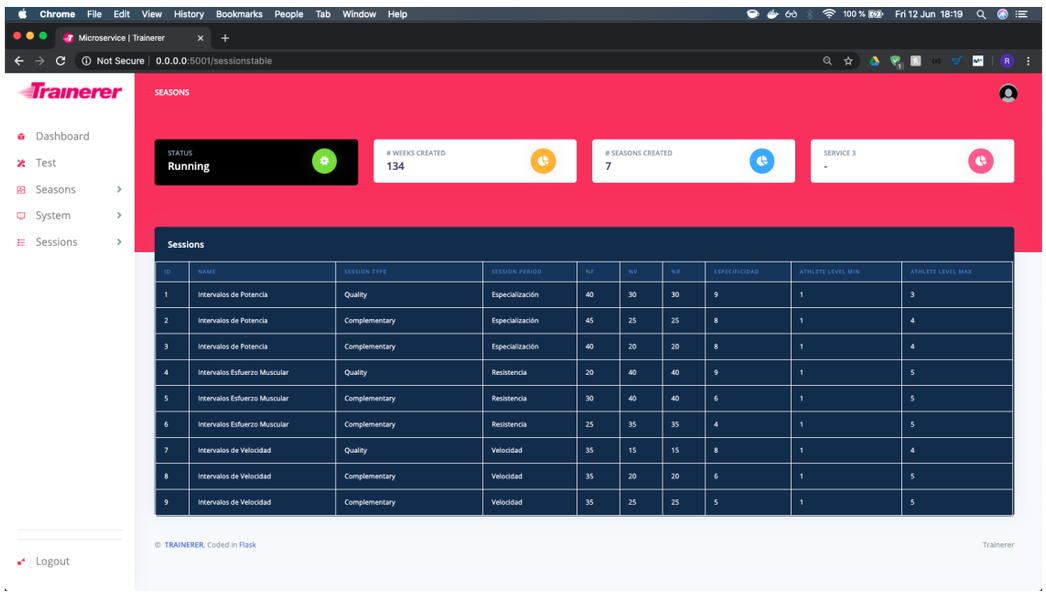
Modificació de temporades predefinides



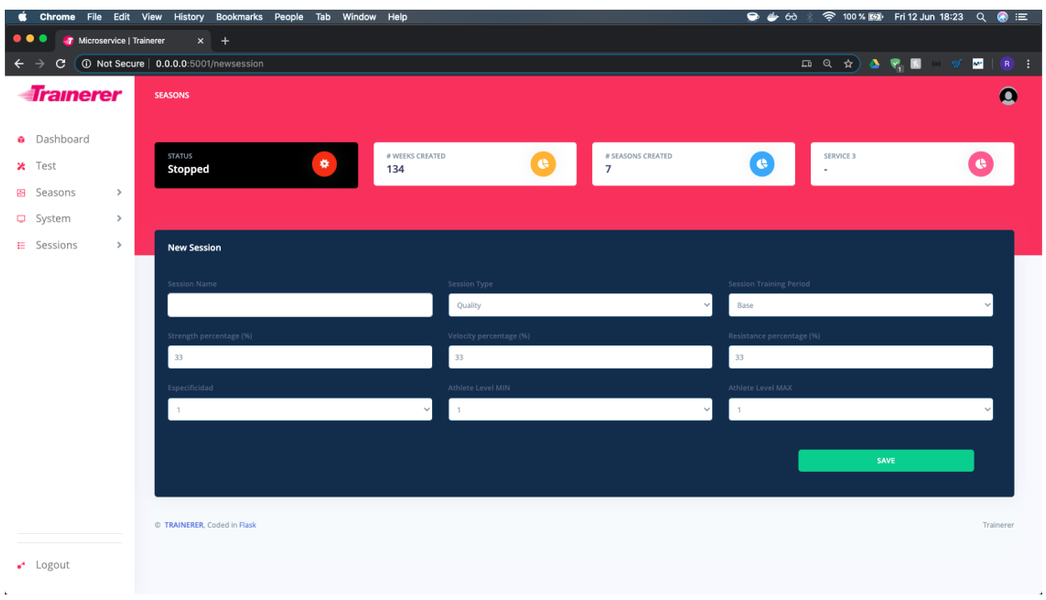
Organització setmana en funció del nombre sessions qualitat (extensió)



Sessions d'entrenament del sistema

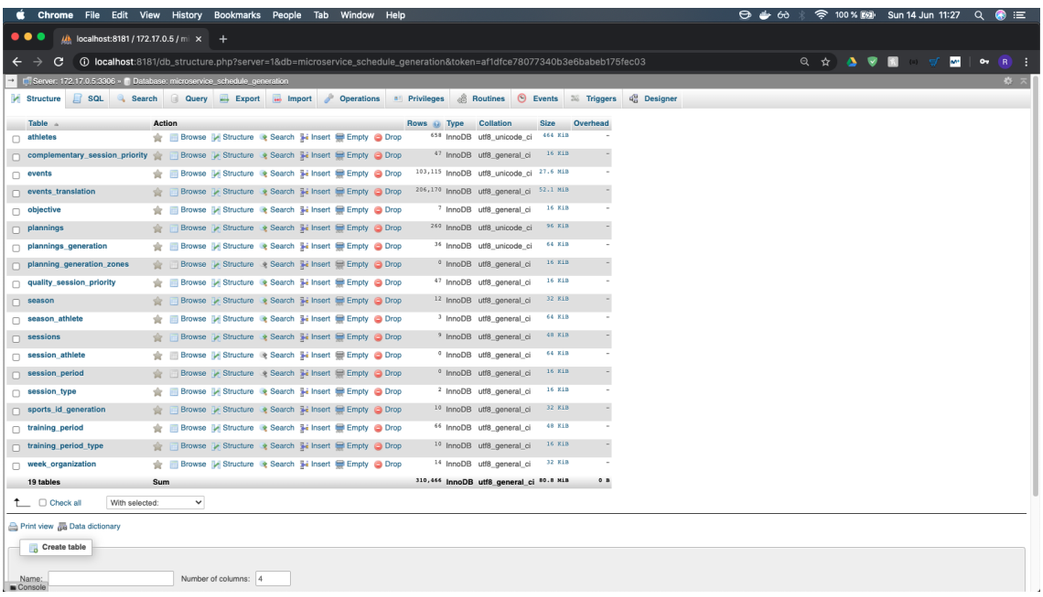


Creació sessió d'entrenament

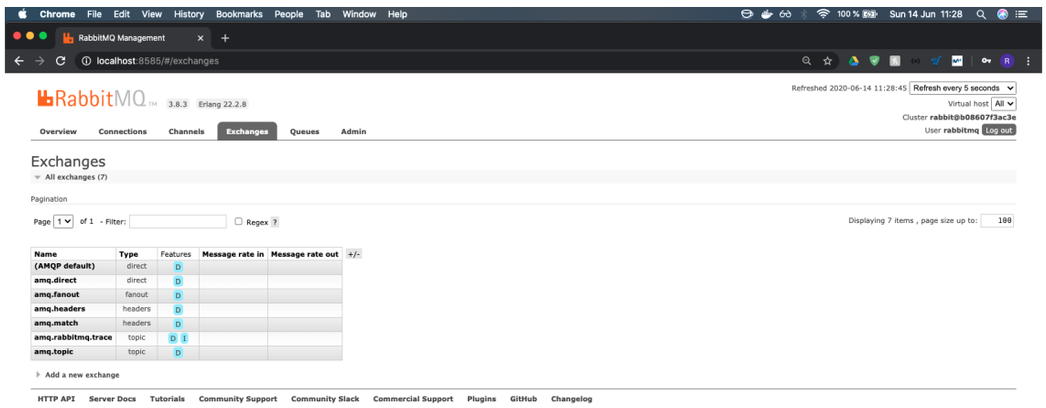


12.3 Docker

Contenidor base de dades - Mariadb



Contenidor gestor connexió asíncrona - RabbitMQ



12.4 Tests - temps d'execució en funció de la disponibilitat diària

Implementació recursiva

30 minuts diaris

```
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash# python3 testRecursive.py
temps crear permutacions -> 0.2285017967224121
Number of permutations -> 1024

FINAL
{'2020-02-03': [[20, 7, 0, 0, 'ESTIRAMENTS']],
 '2020-02-04': [],
 '2020-02-05': [],
 '2020-02-06': [],
 '2020-02-07': [[30, 6, 4, 0, 'ABDOMINALS + FLEXIONS']],
 '2020-02-08': [],
 '2020-02-09': []}
obj value -> 90
temps total -> 4.55883526802063
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash#
```

60 minuts diaris

```
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash# python3 testRecursive.py
temps crear permutacions -> 0.21808934211730957
Number of permutations -> 1024

FINAL
{'2020-02-03': [[25, 7, 0, 0, 'ESTIRAMENTS']],
 '2020-02-04': [],
 '2020-02-05': [],
 '2020-02-06': [],
 '2020-02-07': [[60, 3, 4, 0, 'EXCURSIÃ“  PER  LA  MUNTANYA']],
 '2020-02-08': [],
 '2020-02-09': []}
obj value -> 110
temps total -> 6.540103912353516
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash#
```

90 minuts diaris

```
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash# python3 testRecursive.py
temps crear permutacions -> 0.2196028232574463
Number of permutations -> 1024

FINAL
{'2020-02-03': [[25, 7, 0, 0, 'ESTIRAMENTS']],
 '2020-02-04': [],
 '2020-02-05': [],
 '2020-02-06': [],
 '2020-02-07': [[60, 3, 4, 0, 'EXCURSIÃ“  PER  LA  MUNTANYA']],
 '2020-02-08': [],
 '2020-02-09': []}
obj value -> 110
temps total -> 6.304154872894287
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash#
```

120 minuts diaris

```
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash# python3 testRecursive.py
temps crear permutacions -> 0.22113728523254395
Number of permutations -> 1024

FINAL
{'2020-02-03': [[20, 7, 0, 0, 'ESTIRAMENTS']],
 '2020-02-04': [],
 '2020-02-05': [[120, 8, 2, 1, 'INTERVALS  VELOCITAT']],
 '2020-02-06': [[30, 6, 4, 0, 'ABDOMINALS + FLEXIONS']],
 '2020-02-07': [[120, 8, 3, 0, 'RECUPERACIÃ“  INTERVALS']],
 '2020-02-08': [[105, 8, 1, 0, 'TEMPO']],
 '2020-02-09': []}
obj value -> 320
temps total -> 105.5096206665039
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash#
```

150 minuts diaris

```
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash# python3 testRecursive.py
temps crear permutacions -> 0.23834943771362305
Number of permutations -> 1024

FINAL
{'2020-02-03': [[25, 7, 0, 0, 'ESTIRAMENTS']],
 '2020-02-04': [[60, 3, 4, 0, 'EXCURSIÃ“  PER  LA  MUNTANYA']],
 '2020-02-05': [[120, 8, 2, 1, 'SORTIDES  LLANÃ‡ADES']],
 '2020-02-06': [],
 '2020-02-07': [[120, 8, 2, 0, 'SORTIDES  LLANÃ‡ADES']],
 '2020-02-08': [[135, 8, 1, 0, 'SÃˆRIES PUJADA + PROGRESSIONS']],
 '2020-02-09': []}
obj value -> 340
temps total -> 196.28121733665466
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash#
```

180 minuts diaris

```
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash# python3 testRecursive.py
temps crear permutacions -> 0.28722548484802246
Number of permutations -> 1024

FINAL
{'2020-02-03': [[20, 7, 0, 0, 'ESTIRAMENTS']],
 '2020-02-04': [[180, 8, 6, 0, 'CARTA  BLANCA']],
 '2020-02-05': [[120, 8, 2, 1, 'INTERVALS  VELOCITAT']],
 '2020-02-06': [[30, 6, 4, 0, 'ABDOMINALS + FLEXIONS']],
 '2020-02-07': [[120, 8, 3, 0, 'RECUPERACIÃ“  INTERVALS']],
 '2020-02-08': [[105, 8, 1, 0, 'TEMPO']],
 '2020-02-09': []}
obj value -> 380
temps total -> 377.770446062088
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash#
```

210 minuts diaris

```
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash# python3 testRecursive.py
temps crear permutacions -> 0.22025084495544434
Number of permutations -> 1024

FINAL
{'2020-02-03': [[20, 7, 0, 0, 'ESTIRAMENTS']],
 '2020-02-04': [[180, 8, 6, 0, 'CARTA  BLANCA']],
 '2020-02-05': [[120, 8, 2, 1, 'INTERVALS  VELOCITAT']],
 '2020-02-06': [[30, 6, 4, 0, 'ABDOMINALS + FLEXIONS']],
 '2020-02-07': [[120, 8, 3, 0, 'RECUPERACIÃ“  INTERVALS']],
 '2020-02-08': [[105, 8, 1, 0, 'TEMPO']],
 '2020-02-09': []}
obj value -> 380
temps total -> 715.8507051467896
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash#
```

240 minuts diaris

```
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash# python3 testRecursive.py
temps crear permutacions -> 0.2193434238433838
Number of permutations -> 1024

FINAL
{'2020-02-03': [[20, 7, 0, 0, 'ESTIRAMENTS']],
 '2020-02-04': [[225, 8, 5, 0, 'SIMULACIÃ“  DE  MARXA']],
 '2020-02-05': [[120, 8, 3, 0, 'RECUPERACIÃ“  INTERVALS'],
                [120, 8, 2, 1, 'INTERVALS  VELOCITAT']],
 '2020-02-06': [[30, 6, 4, 0, 'ABDOMINALS + FLEXIONS']],
 '2020-02-07': [[180, 8, 6, 0, 'CARTA  BLANCA']],
 '2020-02-08': [[105, 8, 1, 0, 'TEMPO']],
 '2020-02-09': []}
obj value -> 390
temps total -> 1347.317682504654
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash#
```

Implementació model d'optimització

30 minuts diaris

```
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash# python3 testNewSolver.py
Processing availability time -> 0.0001800060272216797
RESULTS:

objective value -> 90.0
Planning -> 0
sessions 0 in day 0 of planning 0  ------ [20, 7, 0, 0, True, 'ESTIRAMENTS']
sessions 4 in day 1 of planning 0  ------ [30, 6, 4, 0, True, 'ABDOMINALS + FLEXIONS']
Execution time: 1.6024277210235596
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash#
```

60 minuts diaris

```
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash# python3 testNewSolver.py
Processing availability time -> 0.00017833709716796875
RESULTS:

objective value -> 110.0
Planning -> 1
sessions 2 in day 0 of planning 1  ------ [25, 7, 0, 0, True, 'ESTIRAMENTS']
sessions 0 in day 2 of planning 1  ------ [60, 3, 4, 0, False, 'EXCURSIÃ“  PER  LA  MUNTANYA']
Execution time: 2.012683629989624
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash#
```

90 minuts diaris

```
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash# python3 testNewSolver.py
Processing availability time -> 0.0001933574676513672
RESULTS:

objective value -> 110.0
Planning -> 1
sessions 2 in day 0 of planning 1  ------ [25, 7, 0, 0, True, 'ESTIRAMENTS']
sessions 0 in day 2 of planning 1  ------ [60, 3, 4, 0, False, 'EXCURSIÃ“  PER  LA  MUNTANYA']
Execution time: 1.918635368347168
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash#
```

120 minuts diaris

```
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash# python3 testNewSolver.py
Processing availability time -> 0.0002009868621826172
RESULTS:

objective value -> 320.0
Planning -> 0
sessions 4 in day 0 of planning 0  ------ [30, 6, 4, 0, True, 'ABDOMINALS + FLEXIONS']
sessions 2 in day 2 of planning 0  ------ [120, 8, 2, 1, False, 'INTERVALS  VELOCITAT']
sessions 0 in day 3 of planning 0  ------ [20, 7, 0, 0, True, 'ESTIRAMENTS']
sessions 1 in day 4 of planning 0  ------ [105, 8, 1, 0, False, 'TEMPO']
sessions 3 in day 5 of planning 0  ------ [120, 8, 3, 0, False, 'RECUPERACIÃ“  INTERVALS']
Execution time: 1.8944292068481445
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash#
```

150 minuts diaris

```
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash# python3 testNewSolver.py
Processing availability time -> 0.00020360946655273438
RESULTS:

objective value -> 340.0
Planning -> 1
sessions 2 in day 0 of planning 1  ------ [25, 7, 0, 0, True, 'ESTIRAMENTS']
sessions 3 in day 1 of planning 1  ------ [135, 8, 1, 0, False, 'SÃˆRIES PUJADA + PROGRESSIONS']
sessions 5 in day 2 of planning 1  ------ [120, 8, 2, 1, False, 'SORTIDES  LLANÃ‡ADES']
sessions 0 in day 4 of planning 1  ------ [60, 3, 4, 0, False, 'EXCURSIÃ“  PER  LA  MUNTANYA']
sessions 6 in day 5 of planning 1  ------ [120, 8, 2, 0, False, 'SORTIDES  LLANÃ‡ADES']
Execution time: 1.9150919914245605
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash#
```

180 minuts diaris

```
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash# python3 testNewSolver.py
Processing availability time -> 0.00019025802612304688
RESULTS:

objective value -> 380.0
Planning -> 0
sessions 0 in day 0 of planning 0  ------ [20, 7, 0, 0, True, 'ESTIRAMENTS']
sessions 1 in day 1 of planning 0  ------ [105, 8, 1, 0, False, 'TEMPO']
sessions 2 in day 2 of planning 0  ------ [120, 8, 2, 1, False, 'INTERVALS  VELOCITAT']
sessions 4 in day 3 of planning 0  ------ [30, 6, 4, 0, True, 'ABDOMINALS + FLEXIONS']
sessions 5 in day 4 of planning 0  ------ [180, 8, 6, 0, False, 'CARTA  BLANCA']
sessions 3 in day 5 of planning 0  ------ [120, 8, 3, 0, False, 'RECUPERACIÃ“  INTERVALS']
Execution time: 1.8879997730255127
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash#
```

210 minuts diaris

```
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash# python3 testNewSolver.py
Processing availability time -> 0.00017690658569335938
RESULTS:

objective value -> 380.0
Planning -> 2
sessions 0 in day 0 of planning 2  ------ [40, 7, 0, 0, True, 'PROPIOCEPCIÃ“']
sessions 3 in day 1 of planning 2  ------ [135, 8, 1, 0, False, 'SÃˆRIES PUJADA + PROGRESSIONS']
sessions 1 in day 2 of planning 2  ------ [120, 8, 2, 0, False, 'SESSIÃ“  ANÃ€RQUICA']
sessions 5 in day 3 of planning 2  ------ [120, 8, 3, 1, False, 'SORTIDES  LLANÃ‡ADES']
sessions 6 in day 4 of planning 2  ------ [210, 8, 6, 0, False, 'RODATGE  LLIURE']
sessions 2 in day 5 of planning 2  ------ [180, 8, 5, 0, False, 'CARTA  BLANCA']
Execution time: 2.1039202213287354
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash#
```

240 minuts diaris

```
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash# python3 testNewSolver.py
Processing availability time -> 0.0002589225769042969
RESULTS:

objective value -> 380.0
Planning -> 2
sessions 4 in day 0 of planning 2  ------ [50, 6, 4, 0, True, 'GIMNÃ€S A CASA']
sessions 2 in day 1 of planning 2  ------ [180, 8, 5, 0, False, 'CARTA  BLANCA']
sessions 5 in day 2 of planning 2  ------ [120, 8, 3, 1, False, 'SORTIDES  LLANÃ‡ADES']
sessions 3 in day 3 of planning 2  ------ [135, 8, 1, 0, False, 'SÃˆRIES PUJADA + PROGRESSIONS']
sessions 6 in day 4 of planning 2  ------ [210, 8, 6, 0, False, 'RODATGE  LLIURE']
sessions 1 in day 5 of planning 2  ------ [120, 8, 2, 0, False, 'SESSIÃ“  ANÃ€RQUICA']
Execution time: 2.031836986541748
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash#
```

12.5 Tests - temps d'execució en funció del nombre de planificacions predefinides

Implementació recursiva

1 planificació predefinides

```
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash# python3 testRecursiveMultiple.py
Number plannings predefined: 1
temps crear permutacions -> 0.2285158634185791
Number of permutations -> 1024

FINAL
{'2020-02-03': [[40, 7, 0, 0, 'PROPIOCEPCIÃ“'']],
 '2020-02-04': [[120, 8, 3, 0, 'SESSIÃ“  ANÃ€RQUICA']],
 '2020-02-05': [[120, 8, 2, 1, 'RODATGE  PROGRESSIU']],
 '2020-02-06': [[60, 3, 4, 0, 'EXCURSIÃ“  PER  LA  MUNTANYA']],
 '2020-02-07': [[120, 8, 1, 0, 'MANTENIMENT  AERÃ'BIC']],
 '2020-02-08': [[210, 8, 5, 0, 'SORTIDA  EN  GRUPETTA']],
 '2020-02-09': []}
obj value -> 380
temps total -> 311.09212493896484
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash#
```

2 planificacions predefinides

```
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash# python3 testRecursiveMultiple.py
Number plannings predefined: 2
temps crear permutacions -> 0.2395937442779541
Number of permutations -> 1024

FINAL
{'2020-02-03': [[40, 7, 0, 0, 'PROPIOCEPCIÃ“'']],
 '2020-02-04': [[120, 8, 3, 0, 'SESSIÃ“  ANÃ€RQUICA']],
 '2020-02-05': [[120, 8, 2, 1, 'RODATGE  PROGRESSIU']],
 '2020-02-06': [[60, 3, 4, 0, 'EXCURSIÃ“  PER  LA  MUNTANYA']],
 '2020-02-07': [[120, 8, 1, 0, 'MANTENIMENT  AERÃ'BIC']],
 '2020-02-08': [[210, 8, 5, 0, 'SORTIDA  EN  GRUPETTA']],
 '2020-02-09': []}
obj value -> 380
temps total -> 565.6994073390961
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash#
```

3 planificacions predefinides

```
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash# python3 testRecursiveMultiple.py
Number plannings predefined: 3
temps crear permutacions -> 0.23842310905456543
Number of permutations -> 1024

FINAL
{'2020-02-03': [[40, 7, 0, 0, 'PROPIOCEPCIÃ“'']],
 '2020-02-04': [[120, 8, 3, 0, 'SESSIÃ“  ANÃ€RQUICA']],
 '2020-02-05': [[120, 8, 2, 1, 'RODATGE  PROGRESSIU']],
 '2020-02-06': [[60, 3, 4, 0, 'EXCURSIÃ“  PER  LA  MUNTANYA']],
 '2020-02-07': [[120, 8, 1, 0, 'MANTENIMENT  AERÃ'BIC']],
 '2020-02-08': [[210, 8, 5, 0, 'SORTIDA  EN  GRUPETTA']],
 '2020-02-09': []}
obj value -> 380
temps total -> 578.6686291694641
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash#
```

4 planificacions predefinides

```
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash# python3 testRecursiveMultiple.py
Number plannings predefined: 4
temps crear permutacions -> 0.2251884937286377
Number of permutations -> 1024

FINAL
{'2020-02-03': [[40, 7, 0, 0, 'PROPIOCEPCIÃ“'']],
 '2020-02-04': [[120, 8, 3, 0, 'SESSIÃ“  ANÃ€RQUICA']],
 '2020-02-05': [[120, 8, 2, 1, 'RODATGE  PROGRESSIU']],
 '2020-02-06': [[60, 3, 4, 0, 'EXCURSIÃ“  PER  LA  MUNTANYA']],
 '2020-02-07': [[120, 8, 1, 0, 'MANTENIMENT  AERÃ'BIC']],
 '2020-02-08': [[210, 8, 5, 0, 'SORTIDA  EN  GRUPETTA']],
 '2020-02-09': []}
obj value -> 380
temps total -> 678.0229954719543
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash#
```

5 planificacions predefinides

```
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash# python3 testRecursiveMultiple.py
Number plannings predefined: 5
temps crear permutacions -> 0.236433744430542
Number of permutations -> 1024

FINAL
{'2020-02-03': [[40, 7, 0, 0, 'PROPIOCEPCIÃ“'']],
 '2020-02-04': [[120, 8, 3, 0, 'SESSIÃ“  ANÃ€RQUICA']],
 '2020-02-05': [[120, 8, 2, 1, 'RODATGE  PROGRESSIU']],
 '2020-02-06': [[60, 3, 4, 0, 'EXCURSIÃ“  PER  LA  MUNTANYA']],
 '2020-02-07': [[120, 8, 1, 0, 'MANTENIMENT  AERÃ'BIC']],
 '2020-02-08': [[210, 8, 5, 0, 'SORTIDA  EN  GRUPETTA']],
 '2020-02-09': []}
obj value -> 380
temps total -> 836.4802956581116
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash#
```

6 planificacions predefinides

```
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash# python3 testRecursiveMultiple.py
Number plannings predefined: 6
temps crear permutacions -> 0.225785493850708
Number of permutations -> 1024

FINAL
{'2020-02-03': [[40, 7, 0, 0, 'PROPIOCEPCIÃ“'']],
 '2020-02-04': [[120, 8, 3, 0, 'SESSIÃ“  ANÃ€RQUICA']],
 '2020-02-05': [[120, 8, 2, 1, 'RODATGE  PROGRESSIU']],
 '2020-02-06': [[60, 3, 4, 0, 'EXCURSIÃ“  PER  LA  MUNTANYA']],
 '2020-02-07': [[120, 8, 1, 0, 'MANTENIMENT  AERÃ'BIC']],
 '2020-02-08': [[210, 8, 5, 0, 'SORTIDA  EN  GRUPETTA']],
 '2020-02-09': []}
obj value -> 380
temps total -> 930.1142101287842
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash#
```

7 planificacions predefinides

```
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash# python3 testRecursiveMultiple.py
Number plannings predefined: 7
temps crear permutacions -> 0.22039318084716797
Number of permutations -> 1024

FINAL
{'2020-02-03': [[40, 7, 0, 0, 'PROPIOCEPCIÃ“']],
 '2020-02-04': [[120, 8, 3, 0, 'SESSIÃ“  ANÃ€RQUICA']],
 '2020-02-05': [[120, 8, 2, 1, 'RODATGE  PROGRESSIU']],
 '2020-02-06': [[60, 3, 4, 0, 'EXCURSIÃ“  PER  LA  MUNTANYA']],
 '2020-02-07': [[120, 8, 1, 0, 'MANTENIMENT  AERÃ’BIC']],
 '2020-02-08': [[210, 8, 5, 0, 'SORTIDA  EN  GRUPETTA']],
 '2020-02-09': []}
obj value -> 380
temps total -> 1046.8166780471802
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash#
```

8 planificacions predefinides

```
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash# python3 testRecursiveMultiple.py
Number plannings predefined: 8
temps crear permutacions -> 0.22760605812072754
Number of permutations -> 1024

FINAL
{'2020-02-03': [[40, 7, 0, 0, 'PROPIOCEPCIÃ“']],
 '2020-02-04': [[120, 8, 3, 0, 'SESSIÃ“  ANÃ€RQUICA']],
 '2020-02-05': [[120, 8, 2, 1, 'RODATGE  PROGRESSIU']],
 '2020-02-06': [[60, 3, 4, 0, 'EXCURSIÃ“  PER  LA  MUNTANYA']],
 '2020-02-07': [[120, 8, 1, 0, 'MANTENIMENT  AERÃ’BIC']],
 '2020-02-08': [[210, 8, 5, 0, 'SORTIDA  EN  GRUPETTA']],
 '2020-02-09': []}
obj value -> 380
temps total -> 1144.5462653636932
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash#
```

9 planificacions predefinides

```
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash# python3 testRecursiveMultiple.py
Number plannings predefined: 9
temps crear permutacions -> 0.22887015342712402
Number of permutations -> 1024

FINAL
{'2020-02-03': [[40, 7, 0, 0, 'PROPIOCEPCIÃ“']],
 '2020-02-04': [[120, 8, 3, 0, 'SESSIÃ“  ANÃ€RQUICA']],
 '2020-02-05': [[120, 8, 2, 1, 'RODATGE  PROGRESSIU']],
 '2020-02-06': [[60, 3, 4, 0, 'EXCURSIÃ“  PER  LA  MUNTANYA']],
 '2020-02-07': [[120, 8, 1, 0, 'MANTENIMENT  AERÃ’BIC']],
 '2020-02-08': [[210, 8, 5, 0, 'SORTIDA  EN  GRUPETTA']],
 '2020-02-09': []}
obj value -> 380
temps total -> 1214.0394747257233
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash#
```

10 planificacions predefinides

```
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash# python3 testRecursiveMultiple.py
Number plannings predefined: 10
temps crear permutacions -> 0.22455811500549316
Number of permutations -> 1024

FINAL
{'2020-02-03': [[40, 7, 0, 0, 'PROPIOCEPCIÃ“']],
 '2020-02-04': [[120, 8, 3, 0, 'SESSIÃ“  ANÃ€RQUICA']],
 '2020-02-05': [[120, 8, 2, 1, 'RODATGE  PROGRESSIU']],
 '2020-02-06': [[60, 3, 4, 0, 'EXCURSIÃ“  PER  LA  MUNTANYA']],
 '2020-02-07': [[120, 8, 1, 0, 'MANTENIMENT  AERÃ’BIC']],
 '2020-02-08': [[210, 8, 5, 0, 'SORTIDA  EN  GRUPETTA']],
 '2020-02-09': []}
obj value -> 380
temps total -> 1579.4103796482086
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash#
```

11 planificacions predefinides

```
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash# python3 testRecursiveMultiple.py
Number plannings predefined: 11
temps crear permutacions -> 0.22638392448425293
Number of permutations -> 1024

FINAL
{'2020-02-03': [[40, 7, 0, 0, 'PROPIOCEPCIÃ“']],
 '2020-02-04': [[120, 8, 3, 0, 'SESSIÃ“  ANÃ€RQUICA']],
 '2020-02-05': [[120, 8, 2, 1, 'RODATGE  PROGRESSIU']],
 '2020-02-06': [[60, 3, 4, 0, 'EXCURSIÃ“  PER  LA  MUNTANYA']],
 '2020-02-07': [[120, 8, 1, 0, 'MANTENIMENT  AERÃ’BIC']],
 '2020-02-08': [[210, 8, 5, 0, 'SORTIDA  EN  GRUPETTA']],
 '2020-02-09': []}
obj value -> 380
temps total -> 1634.0669090747833
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash#
```

12 planificacions predefinides

```
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash# python3 testRecursiveMultiple.py
Number plannings predefined: 12
temps crear permutacions -> 0.22066593170166016
Number of permutations -> 1024

FINAL
{'2020-02-03': [[40, 7, 0, 0, 'PROPIOCEPCIÃ“']],
 '2020-02-04': [[120, 8, 3, 0, 'SESSIÃ“  ANÃ€RQUICA']],
 '2020-02-05': [[120, 8, 2, 1, 'RODATGE  PROGRESSIU']],
 '2020-02-06': [[60, 3, 4, 0, 'EXCURSIÃ“  PER  LA  MUNTANYA']],
 '2020-02-07': [[120, 8, 1, 0, 'MANTENIMENT  AERÃ’BIC']],
 '2020-02-08': [[210, 8, 5, 0, 'SORTIDA  EN  GRUPETTA']],
 '2020-02-09': []}
obj value -> 380
temps total -> 1641.4639601707458
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash#
```

Implementació model d'optimització

1 planificació predefinides

```
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash# python3 testRecursiveMultiple.py
Number plannings predefined: 1
temps crear permutacions -> 0.2285158634185791
Number of permutations -> 1024

FINAL
{'2020-02-03': [[40, 7, 0, 0, 'PROPIOCEPCIÃ“']],
 '2020-02-04': [[120, 8, 3, 0, 'SESSIÃ“  ANÃ€RQUICA']],
 '2020-02-05': [[120, 8, 2, 1, 'RODATGE  PROGRESSIU']],
 '2020-02-06': [[60, 3, 4, 0, 'EXCURSIÃ“  PER  LA  MUNTANYA']],
 '2020-02-07': [[120, 8, 1, 0, 'MANTENIMENT  AERÃ'BIC']],
 '2020-02-08': [[210, 8, 5, 0, 'SORTIDA  EN  GRUPETTA']],
 '2020-02-09': []}
obj value -> 380
temps total -> 311.09212493896484
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash#
```

2 planificacions predefinides

```
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash# python3 testNewSolverMultiple.py
Processing availability time -> 0.0002567768096923828
RESULTS:

objective value -> 380.0
Planning -> 0
sessions 0 in day 0 of planning 0  ------ [40, 7, 0, 0, True, 'PROPIOCEPCIÃ“']
sessions 4 in day 1 of planning 0  ------ [120, 8, 3, 0, False, 'SESSIÃ“  ANÃ€RQUICA']
sessions 3 in day 2 of planning 0  ------ [120, 8, 2, 1, False, 'RODATGE  PROGRESSIU']
sessions 2 in day 3 of planning 0  ------ [60, 3, 4, 0, False, 'EXCURSIÃ“  PER  LA  MUNTANYA']
sessions 6 in day 4 of planning 0  ------ [150, 8, 6, 0, False, 'SORTIDA  LLIURE']
sessions 5 in day 5 of planning 0  ------ [210, 8, 5, 0, False, 'SORTIDA  EN  GRUPETTA']
number of sample plannings: 2
Execution time: 1.6960971355438232
```

3 planificacions predefinides

```
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash# python3 testNewSolverMultiple.py
Processing availability time -> 0.0002183914184570312 5
RESULTS:

objective value -> 380.0
Planning -> 1
sessions 0 in day 0 of planning 1  ------ [20, 7, 0, 0, True, 'ESTIRAMENTS']
sessions 1 in day 1 of planning 1  ------ [105, 8, 1, 0, False, 'TEMPO']
sessions 3 in day 2 of planning 1  ------ [120, 8, 3, 0, False, 'RECUPERACIÃ“  INTERVALS']
sessions 2 in day 3 of planning 1  ------ [120, 8, 2, 1, False, 'INTERVALS  VELOCITAT']
sessions 5 in day 4 of planning 1  ------ [180, 8, 6, 0, False, 'CARTA  BLANCA']
sessions 6 in day 5 of planning 1  ------ [225, 8, 5, 0, False, 'SIMULACIÃ“  DE  MARXA']
number of sample plannings: 3
Execution time: 1.6783747673034668
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash#
```

4 planificacions predefinides

```
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash# python3 testNewSolverMultiple.py
Processing availability time -> 0.00026416778564453125
RESULTS:

objective value -> 380.0
Planning -> 1
sessions 0 in day 0 of planning 1  ------ [20, 7, 0, 0, True, 'ESTIRAMENTS']
sessions 3 in day 1 of planning 1  ------ [120, 8, 3, 0, False, 'RECUPERACIÃ“  INTERVALS']
sessions 2 in day 2 of planning 1  ------ [120, 8, 2, 1, False, 'INTERVALS  VELOCITAT']
sessions 4 in day 3 of planning 1  ------ [30, 6, 4, 0, True, 'ABDOMINALS + FLEXIONS']
sessions 5 in day 4 of planning 1  ------ [180, 8, 6, 0, False, 'CARTA  BLANCA']
sessions 6 in day 5 of planning 1  ------ [225, 8, 5, 0, False, 'SIMULACIÃ“  DE  MARXA']
number of sample plannings: 4
Execution time: 1.797853708267212
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash#
```

5 planificacions predefinides

```
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash# python3 testNewSolverMultiple.py
Processing availability time -> 0.00018143653869628906
RESULTS:

objective value -> 380.0
Planning -> 0
sessions 0 in day 0 of planning 0  ------ [40, 7, 0, 0, True, 'PROPIOCEPCIÃ“']
sessions 4 in day 1 of planning 0  ------ [120, 8, 3, 0, False, 'SESSIÃ“  ANÃ€RQUICA']
sessions 1 in day 2 of planning 0  ------ [120, 8, 1, 0, False, 'MANTENIMENT  AERÃ'BIC']
sessions 3 in day 3 of planning 0  ------ [120, 8, 2, 1, False, 'RODATGE  PROGRESSIU']
sessions 6 in day 4 of planning 0  ------ [150, 8, 6, 0, False, 'SORTIDA  LLIURE']
sessions 5 in day 5 of planning 0  ------ [210, 8, 5, 0, False, 'SORTIDA  EN  GRUPETTA']
number of sample plannings: 5
Execution time: 1.890709400177002
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash#
```

6 planificacions predefinides

```
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash# python3 testNewSolverMultiple.py
Processing availability time -> 0.00018024444580078125
RESULTS:

objective value -> 380.0
Planning -> 0
sessions 0 in day 0 of planning 0  ------ [40, 7, 0, 0, True, 'PROPIOCEPCIÃ“']
sessions 1 in day 1 of planning 0  ------ [120, 8, 1, 0, False, 'MANTENIMENT  AERÃ'BIC']
sessions 3 in day 2 of planning 0  ------ [120, 8, 2, 1, False, 'RODATGE  PROGRESSIU']
sessions 2 in day 3 of planning 0  ------ [60, 3, 4, 0, False, 'EXCURSIÃ“  PER  LA  MUNTANYA']
sessions 4 in day 4 of planning 0  ------ [120, 8, 3, 0, False, 'SESSIÃ“  ANÃ€RQUICA']
sessions 6 in day 5 of planning 0  ------ [150, 8, 6, 0, False, 'SORTIDA  LLIURE']
number of sample plannings: 6
Execution time: 2.0540966987609863
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash#
```

7 planificacions predefinides

```
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash# python3 testNewSolverMultiple.py
Processing availability time -> 0.0001785755157470703
RESULTS:

objective value -> 380.0
Planning -> 1
sessions 4 in day 0 of planning 1  ------ [30, 6, 4, 0, True, 'ABDOMINALS + FLEXIONS']
sessions 1 in day 1 of planning 1  ------ [105, 8, 1, 0, False, 'TEMPO']
sessions 2 in day 2 of planning 1  ------ [120, 8, 2, 1, False, 'INTERVALS  VELOCITAT']
sessions 0 in day 3 of planning 1  ------ [20, 7, 0, 0, True, 'ESTIRAMENTS']
sessions 3 in day 4 of planning 1  ------ [120, 8, 3, 0, False, 'RECUPERACIÃ“  INTERVALS']
sessions 5 in day 5 of planning 1  ------ [180, 8, 6, 0, False, 'CARTA  BLANCA']
number of sample plannings: 7
Execution time: 2.3360326290130615
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash#
```

8 planificacions predefinides

```
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash# python3 testNewSolverMultiple.py
Processing availability time -> 0.00018262863159179688
RESULTS:

objective value -> 380.0
Planning -> 0
sessions 0 in day 0 of planning 0  ------ [40, 7, 0, 0, True, 'PROPIOCEPCIÃ“']
sessions 4 in day 1 of planning 0  ------ [120, 8, 3, 0, False, 'SESSIÃ“  ANÃ€RQUICA']
sessions 1 in day 2 of planning 0  ------ [120, 8, 1, 0, False, 'MANTENIMENT  AERÃ'BIC']
sessions 3 in day 3 of planning 0  ------ [120, 8, 2, 1, False, 'RODATGE  PROGRESSIU']
sessions 6 in day 4 of planning 0  ------ [150, 8, 6, 0, False, 'SORTIDA  LLIURE']
sessions 5 in day 5 of planning 0  ------ [210, 8, 5, 0, False, 'SORTIDA  EN  GRUPETTA']
number of sample plannings: 8
Execution time: 2.662358283996582
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash#
```

9 planificacions predefinides

```
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash# python3 testNewSolverMultiple.py
Processing availability time -> 0.0001823902130126953
RESULTS:

objective value -> 380.0
Planning -> 4
sessions 3 in day 0 of planning 4  ------ [20, 7, 0, 0, True, 'ESTIRAMENTS']
sessions 1 in day 1 of planning 4  ------ [120, 8, 1, 0, False, 'SESSIÃ“  ANÃ€RQUICA']
sessions 0 in day 2 of planning 4  ------ [120, 8, 2, 1, False, 'MANTENIMENT  AERÃ'BIC']
sessions 5 in day 3 of planning 4  ------ [45, 7, 4, 0, True, 'PROPIOCEPCIÃ“']
sessions 2 in day 4 of planning 4  ------ [150, 8, 6, 0, False, 'SORTIDA  LLIURE']
sessions 4 in day 5 of planning 4  ------ [150, 8, 3, 0, False, 'SÃˆRIES  PROGRESSIVES']
number of sample plannings: 9
Execution time: 2.7656757831573486
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash#
```

10 planificacions predefinides

```
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash# python3 testNewSolverMultiple.py
Processing availability time -> 0.0001957416534423828
RESULTS:

objective value -> 380.0
Planning -> 0
sessions 0 in day 0 of planning 0  ------ [40, 7, 0, 0, True, 'PROPIOCEPCIÃ“']
sessions 4 in day 1 of planning 0  ------ [120, 8, 3, 0, False, 'SESSIÃ“  ANÃ€RQUICA']
sessions 3 in day 2 of planning 0  ------ [120, 8, 2, 1, False, 'RODATGE  PROGRESSIU']
sessions 2 in day 3 of planning 0  ------ [60, 3, 4, 0, False, 'EXCURSIÃ“  PER  LA  MUNTANYA']
sessions 6 in day 4 of planning 0  ------ [150, 8, 6, 0, False, 'SORTIDA  LLIURE']
sessions 5 in day 5 of planning 0  ------ [210, 8, 5, 0, False, 'SORTIDA  EN  GRUPETTA']
number of sample plannings: 10
Execution time: 2.775928497314453
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash#
```

11 planificacions predefinides

```
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash# python3 testNewSolverMultiple.py
Processing availability time -> 0.0003104209899902344
RESULTS:

objective value -> 380.0
Planning -> 4
sessions 3 in day 0 of planning 4  ------ [20, 7, 0, 0, True, 'ESTIRAMENTS']
sessions 1 in day 1 of planning 4  ------ [120, 8, 1, 0, False, 'SESSIÃ“  ANÃ€RQUICA']
sessions 0 in day 2 of planning 4  ------ [120, 8, 2, 1, False, 'MANTENIMENT  AERÃ'BIC']
sessions 5 in day 3 of planning 4  ------ [45, 7, 4, 0, True, 'PROPIOCEPCIÃ“']
sessions 4 in day 4 of planning 4  ------ [150, 8, 3, 0, False, 'SÃˆRIES  PROGRESSIVES']
sessions 6 in day 5 of planning 4  ------ [210, 8, 5, 0, False, 'RODATGE  LLIURE']
number of sample plannings: 11
Execution time: 3.015747308731079
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash#
```

12 planificacions predefinides

```
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash# python3 testNewSolverMultiple.py
Processing availability time -> 0.00018286705017089844
RESULTS:

objective value -> 380.0
Planning -> 1
sessions 4 in day 0 of planning 1  ------ [30, 6, 4, 0, True, 'ABDOMINALS + FLEXIONS']
sessions 1 in day 1 of planning 1  ------ [105, 8, 1, 0, False, 'TEMPO']
sessions 2 in day 2 of planning 1  ------ [120, 8, 2, 1, False, 'INTERVALS  VELOCITAT']
sessions 0 in day 3 of planning 1  ------ [20, 7, 0, 0, True, 'ESTIRAMENTS']
sessions 5 in day 4 of planning 1  ------ [180, 8, 6, 0, False, 'CARTA  BLANCA']
sessions 3 in day 5 of planning 1  ------ [120, 8, 3, 0, False, 'RECUPERACIÃ“  INTERVALS']
number of sample plannings: 12
Execution time: 3.483834981918335
(venv) root@07df548eade2:/usr/src/IAplannings/planning_generation/bash# python3 testNewSolverMultiple.py
```

Bibliografia

[1] Facultat d'Informàtica de Barcelona. *GESTIÓ DE PROJECTES (GEP)*. URL: https://www.fib.upc.edu/sites/fib/files/documents/estudis/guia-gep-raco-fib.pdf. (visitat: 02.03.2020).

[2] Eduardo Barrios. *Patrón Repositorio*. URL: https://dev.to/ebarrioscode/patron-repositorio-repository-pattern-y-unidad-de-trabajo-unit-of-work-en-asp-net-core-webapi-3-0-5goj. (visitat: 20.05.2020).

[3] Dr. Peter Brucker. *Scheduling Algorithms - Universität Osnabrück*. URL: http://math.nsc.ru/LBRT/k5/Scheduling/BruckerSchedulingAlgorithms_Full.pdf. (visitat: 01.06.2020).

[4] CEOE. *Cómo fijar los días efectivos de trabajo y el horario laboral de mis trabajadores*. URL: http://www.ceoecuenca.es/portal/lang__es/rowid__1539412,60133/dTabID__1/tabid__25091/Default.aspx. (visitat: 02.07.2020).

[5] COIN|OR. *CBC*. URL: https://github.com/coin-or/Cbc. (visitat: 01.06.2020).

[6] Coworking. *Serveis canòdrom*. URL: https://canodrom.com/serveis/. (visitat: 02.08.2020).

[7] Django. *Django*. URL: https://www.djangoproject.com/. (visitat: 20.05.2020).

[8] Docker. *Overview of Docker Compose*. URL: https://docs.docker.com/compose/. (visitat: 27.04.2020).

[9] Python Documentation. *TimeComplexity*. URL: https://wiki.python.org/moin/TimeComplexity. (visitat: 29.05.2020).

[10] Dropbox. *Dropbox*. URL: https://www.dropbox.com/es_ES/. (visitat: 20.05.2020).

[11] *Facultat d'Informàtica de Barcelona*. URL: https://www.fib.upc.edu/. (visitat: 22.02.2020).

[12] Flask. *Flask*. URL: https://flask.palletsprojects.com/en/1.1.x/. (visitat: 20.05.2020).

[13] Mariadb Foundation. *Documentation*. URL: https://mariadb.org/documentation/. (visitat: 20.05.2020).

[14] Gigabyte. *Gigabyte GB-BRI5H-8250-BW*. URL: https://www.amazon.es/Gigabyte-GB-BRI5H-8250-BW-Ultra-Compacto-barebones/dp/B07CZ5FGLY/ref=sr_1_6?__mk_es_ES. (visitat: 02.08.2020).

[15] GitLab. *GitLab*. URL: https://about.gitlab.com/. (visitat: 06.03.2020).

[16] Glassdor. *Sueldos para Full Stack Developer*. URL: https://www.glassdoor.es/Sueldos/full-stack-developer-sueldo-SRCH_KO0,20.htm. (visitat: 02.07.2020).

[17] Glassdor. *Sueldos para Project Manager*. URL: https://www.glassdoor.es/Sueldos/project-manager-sueldo-SRCH_KO0,15.htm. (visitat: 02.07.2020).

[18] Glassdor. *Sueldos para Tester*. URL: https://www.glassdoor.es/Sueldos/tester-sueldo-SRCH_KO0,6.htm. (visitat: 02.07.2020).

[19] GNU. *GNU Linear Programming Kit*. URL: https://www.gnu.org/software/glpk/. (visitat: 01.06.2020).

[20] Google. *Advancing AI for everyone*. URL: https://ai.google/. (visitat: 03.06.2020).

[21] Google. *CP-SAT Solver*. URL: https://developers.google.com/optimization/cp/cp_solver. (visitat: 01.06.2020).

[22] Google. *Google OR-Tools*. URL: https://developers.google.com/optimization. (visitat: 03.06.2020).

[23] Google. *The Glop Linear Solver*. URL: https://developers.google.com/optimization/lp/glop. (visitat: 01.06.2020).

[24] Gurobi. *The Fastest Solver*. URL: https://www.gurobi.com/. (visitat: 01.06.2020).

[25] William E Hart, Jean-Paul Watson i David L Woodruff. "Pyomo: modeling and solving mathematical programs in Python". A: *Mathematical Programming Computation* 3.3 (2011), pàg. 219 - 260.

[26] Red Hat. *¿Qué es Linux?* URL: https://www.redhat.com/es/topics/linux. (visitat: 27.04.2020).

[27] Can Ho. *Understanding a middleware pattern*. URL: https://dev.to/ebarrioscode/patron-repositorio-repository-pattern-y-unidad-de-trabajo-unit-of-work-en-asp-net-core-webapi-3-0-5goj. (visitat: 20.05.2020).

[28] IBM. *CPLEX Optimizer*. URL: https://www.ibm.com/es-es/analytics/cplex-optimizer. (visitat: 01.06.2020).

[29] Enric Rodríguez-Carbonelly (UPC) Javier Larrosa Albert Oliveras. *The Simplex Method*. URL: https://www.cs.upc.edu/~erodri/webpage/cps/theory/lp/simplex/slides.pdf. (visitat: 01.06.2020).

[30] S.L. KANANAH CIENCIA DEPORTIVA. *Trainerer*. URL: https://www.trainerer.com/. (visitat: 22.02.2020).

[31] Leonela Lanz. *Training Stress Score (TSS)*. URL: https://www.adnciclista.com/np-if-tss/. (visitat: 23.02.2020).

[32] Object-relational mapping. *Wikipedia*. URL: https://en.wikipedia.org/wiki/Object-relational_mapping. (visitat: 20.05.2020).

[33] Enrique Romero Merino. *Enrique Romero Merino*. URL: https://www.cs.upc.edu/~eromero/. (visitat: 24.02.2020).

[34] Microsoft. *Middleware de ASP.NET Core*. URL: hhttps://docs.microsoft.com/es-es/aspnet/core/fundamentals/middleware/?view=aspnetcore-3.1. (visitat: 20.05.2020).

[35] MiniZinc. *MiniZinc Challenge 2019 Results*. URL: https://www.minizinc.org/challenge2019/results2019.html. (visitat: 03.06.2020).

[36] Overleaf. *Overleaf*. URL: https://es.overleaf.com/about. (visitat: 02.03.2020).

[37] Pivotal. *RabbitMq*. URL: https://www.rabbitmq.com/. (visitat: 20.05.2020).

[38] Python. *Classes*. URL: https://docs.python.org/3/tutorial/classes.html. (visitat: 20.10.2020).

[39] Python. *PEP 8 – Style Guide for Python Code*. URL: https://www.python.org/dev/peps/pep-0008/. (visitat: 06.03.2020).

[40] Real Python. *Object-Oriented Programming (OOP) in Python 3*. URL: https://realpython.com/python3-object-oriented-programming/. (visitat: 20.05.2020).

[41] Reddit. *Reddit - the front page of internet*. URL: https://www.reddit.com/. (visitat: 20.05.2020).

[42] Adidas Runtastic. *Runtastic*. URL: https://www.runtastic.com/es/. (visitat: 27.04.2020).

[43] Aarti Singh. *Primal-Dual Interior-Point Methods*. URL: http://www.cs.cmu.edu/~pradeepr/convexopt/Lecture_Slides/primal-dual.pdf. (visitat: 01.06.2020).

[44] Slack. *Slack*. URL: https://slack.com/intl/es-es/. (visitat: 06.03.2020).

[45] SQLAlchemy. *The Python SQL Toolkit and Object Relational Mapper*. URL: https://www.sqlalchemy.org/. (visitat: 20.05.2020).

[46] ECMA-404 The JSON Data Interchange Standard. *Introducción a JSON*. URL: https://www.json.org/json-es.html. (visitat: 20.05.2020).

[47] Strava. *Strava*. URL: https://www.strava.com/?hl=es. (visitat: 27.04.2020).

[48] Orix Systems. *¿Qué es un framework y para qué se utiliza?* URL: https://www.orix.es/que-es-un-framework-y-para-que-se-utiliza. (visitat: 01.06.2020).

[49] New York University. *Computational Complexity, Approximation Algorithms*. URL: http://web-static.stern.nyu.edu/om/faculty/pinedo/scheduling/shakhlevich/handout05.pdf. (visitat: 01.06.2020).

[50] Wikipedia. *Docker (software)*. URL: https://es.wikipedia.org/wiki/Docker_(software). (visitat: 02.03.2020).

[51] Wikipedia. *Mapeo objeto-relacional*. URL: https://es.wikipedia.org/wiki/Mapeo_objeto-relacional. (visitat: 02.03.2020).

[52] Wikipedia. *Patrón de diseño*. URL: https://es.wikipedia.org/wiki/Patron_de_dise%C3%B1o. (visitat: 02.03.2020).

[53] Wikipedia. *Pruebas de rendimiento del software*. URL: https://es.wikipedia.org/wiki/Pruebas_de_rendimiento_del_software#Prueba_de_estr%C3%A9s. (visitat: 20.05.2020).

[54] Wikipedia. *Solver*. URL: https://en.wikipedia.org/wiki/Solver. (visitat: 01.06.2020).

[55] Fengqi You YenChieh Chou (ChE 345 Spring 2014) Steward: Dajun Yue. *Branch and cut*. URL: https://optimization.mccormick.northwestern.edu/index.php/Branch_and_cut. (visitat: 01.06.2020).