UNIVERSITAT DE BARCELONA

Treball Final de Grau

GRAU D'ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

Sistema intel·ligent de distribució de comandes en bars basat en càrrega i temps d'execució

Autor: Pol Ferrer Fernández

Director: Albert Clapés Sintes
Realitzat a: Departament de Matemàtiques i Informàtica

Barcelona, 15 de gener de 2026

Declaració d'autoria

Per la present declaro que algunes parts d'aquest manuscrit s'han redactat mitjançant l'ús de **chatgpt.com** en la seva versió pro. Aquestes eines s'han fet servir per millorar la claredat del llenguatge i suggerir redactats alternatius.

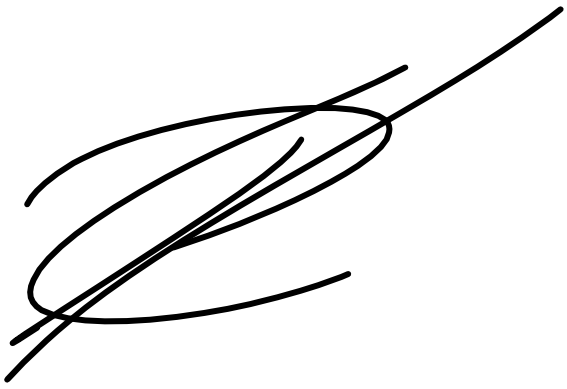
Respecte el programari, s'ha utilitzat **GitHub copilot** per autocomplció de codi i, pel que fa a la documentació, s'ha utilitzat Overleaf amb l'extensió de Chrome **corrector ortográfico y gramatical - LanguageTool** per ortografia. Tot el contingut ha estat revisat críticament i validat per l'autor, que assumeix tota la responsabilitat de la versió final del Treball Fi de Grau lliurat (tant del manuscrit com del codi).

A més, val a dir que no s'ha utilitzat cap sistema d'IA generativa per generar les idees, els anàlisis o els resultats relatius al present treball.

Lloc i data: [Barcelona], [15/01/2026]

Signatura: _____

Nom i cognoms: Pol Ferrer Fernández



Resum

Aquest Treball de Fi de Grau se centra en el **disseny, implementació i avaluació d'un algoritme de repartiment de càrrega de treball** aplicat a la distribució de comandes en entorns de bar amb alta concurrència. La motivació del treball sorgeix de l'experiència directa de l'autor treballant en un bar de còctels, on es van identificar situacions recurrents de saturació a la barra i una distribució poc equilibrada de les tasques entre cambrers, especialment en moments de gran volum de comandes.

Tot i que existeixen des de fa anys sistemes informàtics per a la gestió de comandes en bars i restaurants, molts d'aquests es basen en una separació rígida de rols o en fluxos de treball estàtics. En contraposició, la proposta d'aquest treball parteix d'un escenari en què **tots els cambrers de barra poden preparar qualsevol producte**, i el repte principal és decidir **com repartir les tasques de manera eficient i equitativa en temps real**.

El treball aborda aquest problema mitjançant un enfocament algorísmic basat en criteris de càrrega acumulada, cost estimat de cada producte i ordre temporal de les comandes. S'han **dissenyat, implementat i analitzat diferents variacions de l'algoritme de repartiment**, amb l'objectiu d'avaluar com afecten aquests criteris al balanç de càrrega, al temps de preparació i al comportament global del sistema.

Per validar les diferents propostes, s'ha desenvolupat un sistema software de suport que permet simular escenaris de servei amb múltiples comandes i cambrers, recollint mètriques quantitatives sobre càrrega, temps i inactivitat. Els resultats mostren que l'anàlisi comparativa de les variants de l'algoritme permet identificar estratègies de repartiment més equilibrades i eficients, especialment en situacions d'alta demanda, demostrant la viabilitat de l'enfocament proposat com a eina de suport operatiu en entorns de restauració.

Abstract (English)

This Bachelor's Thesis explores the **design and evaluation of a workload distribution algorithm** for managing drink orders in busy bar environments. The work is motivated by the author's personal experience working as a bartender, where uneven task allocation and congestion at the bar were common issues during peak hours.

While digital order management systems are already widely used in the hospitality sector, many of them rely on predefined roles or static task assignments. This project takes a different approach by considering a setting in which **all bartenders are equally capable of preparing any item**, shifting the focus towards deciding **how tasks should be dynamically assigned in real time**.

The proposed solution is based on an algorithm that considers factors such as the current workload of each bartender, the estimated preparation effort of each product, and the arrival order of requests. Several **alternative algorithmic strategies are implemented and compared** in order to study their impact on workload balance, service time, and overall system efficiency.

A lightweight software system is developed to simulate realistic service scenarios and collect quantitative metrics related to task distribution and bartender activity. The experimental results indicate that comparing different algorithmic variants makes it possible to achieve a more balanced and efficient distribution of work, particularly under high-demand conditions, supporting the validity of the proposed approach as a decision-support mechanism for bar operations.

Resumen (Castellano)

Este Trabajo de Fin de Grado analiza el **diseño y la evaluación de un algoritmo de distribución de carga de trabajo** aplicado a la gestión de comandas en bares con alta afluencia. El proyecto nace a partir de la experiencia personal del autor trabajando en un bar de cócteles, donde se observaban con frecuencia problemas de saturación en la barra y una asignación desigual de tareas durante los momentos de mayor actividad.

Aunque los sistemas digitales de gestión de comandas están ampliamente extendidos en el sector de la restauración, muchos de ellos utilizan flujos de trabajo rígidos o una separación fija de roles. Frente a este enfoque, el presente trabajo considera un escenario en el que **todos los camareros de barra pueden preparar cualquier producto**, trasladando el reto principal a la **asignación dinámica y eficiente de las tareas**.

La solución propuesta se basa en un algoritmo que tiene en cuenta la carga acumulada de cada camarero, el coste estimado de preparación de los productos y el orden de llegada de las comandas. Se implementan y comparan **distintas variantes del algoritmo**, con el objetivo de analizar su efecto sobre el equilibrio de carga, los tiempos de servicio y el comportamiento global del sistema.

Para llevar a cabo la validación, se desarrolla un sistema software sencillo que permite simular escenarios realistas y obtener métricas cuantitativas sobre la distribución del trabajo y la actividad de los camareros. Los resultados muestran que el análisis comparativo de las distintas estrategias permite mejorar el reparto de tareas, especialmente en situaciones de alta demanda, confirmando la utilidad del enfoque propuesto como herramienta de apoyo operativo.

Agraïments

Vull expressar el meu agraïment, en primer lloc, al meu tutor, Albert Clapés, per l'acompanyament durant tot el projecte i, especialment, per ajudar-me a enfocar bé la idea i a prendre decisions clau quan calia redefinir l'abast del treball.

També vull donar les gràcies als meus companys, tant per les aportacions i comentaris com pel suport en el dia a dia, que han fet més fàcil mantenir el ritme i la motivació.

Així mateix, agraeixo als propietaris i responsables del bar la seva disponibilitat per respondre l'enquesta i aportar el context real necessari per orientar el projecte cap a necessitats pràctiques.

Finalment, vull agrair a la meva família la paciència i el suport, especialment en els moments d'estrès, per aguantar-me i ajudar-me a tirar endavant fins al final.

Índex

1 Introducció

El projecte presentat en aquest Treball de Fi de Grau consisteix en el desenvolupament d'un algoritme d'assignació de comandes en el context d'un bar de còctels, on tots els cambrers han de ser capaços de preparar qualsevol producte de la carta. L'objectiu principal de l'algoritme és trobar un equilibri entre un repartiment just de la càrrega de treball durant cada torn i l'optimització del temps necessari per completar les comandes.

Es tracta d'un TFG clarament aplicat, en el qual s'ha dissenyat un algoritme propi i s'ha avaluat mitjançant un entorn de simulació creat específicament per comparar diferents estratègies de repartiment en un escenari realista. En cap cas es pretén presentar un algoritme pioner, sinó analitzar una solució pensada per l'autor, inspirada en algorismes clàssics de repartiment de càrrega utilitzats en sistemes informàtics.

1.1 Context

El context d'un bar de restauració és complex, ja que el flux de comandes depèn de molts factors difícils de controlar. És habitual que durant diverses hores la càrrega de treball sigui baixa, però que de manera sobtada s'entri en una hora punta amb un volum molt elevat de comandes. En aquests moments de màxima afluència, la gestió tradicional de les comandes mitjançant comunicació verbal pot generar confusions i desorganització, especialment quan hi ha diversos cambrers cantant comandes alhora.

Per aquest motiu, disposar d'un sistema que mostri de manera clara i immediata les comandes en una pantalla, així com quin cambrer s'encarrega de cada producte, pot resultar molt útil per a aquest tipus d'establiments. Una solució d'aquest tipus pot aportar un benefici econòmic important per diferents raons: en primer lloc, la satisfacció del client augmenta, ja que habitualment agraeix rebre la comanda ràpidament després de demanar-la; en segon lloc, una preparació més eficient accelera la rotació de clients i, en moments d'alta demanda, permet facturar més.

Finalment, cal tenir en compte els beneficis d'equilibrar la càrrega entre cambrers. Sovint passa que, per la manera de treballar d'un cambrer, aquest assumeixi sistemàticament més o menys càrrega que un altre, per exemple perquè tendeix a escollir els productes més fàcils. Aquest desequilibri pot derivar en problemes tant logístics com interpersonals entre els empleats. Repartir les tasques de manera equitativa mitjançant un sistema ajuda a evitar aquestes situacions i a millorar el funcionament general del servei.

El repte principal d'aquest treball és definir com realitzar aquesta distribució dels productes de cada comanda de la manera més eficient i justa possible, especialment en situacions d'alta demanda.

1.2 Motivació

La motivació d'aquest treball neix de la necessitat d'abordar problemes reals detectats en el funcionament quotidià d'un bar amb alta concurrència. Durant les hores punta, una distribució poc equilibrada de les tasques pot provocar saturacions puntuals, augment dels temps d'espera i una càrrega de treball desigual entre cambrers, factors que afecten tant l'eficiència del servei com la qualitat percebuda pels clients.

Tot i la presència de sistemes digitals de gestió de comandes en el sector de la restauració, molts d'aquests se centren principalment en la presa i visualització de comandes, però no resolen de manera explícita el problema de com repartir internament les tasques entre els treballadors quan tots poden assumir qualsevol funció. Aquesta manca de mecanismes de repartiment dinàmic fa que, en situacions de gran volum de feina, la càrrega recaigui de manera desigual en alguns cambrers.

En aquest context, resulta rellevant explorar un enfocament algorísmic que permeti donar suport a la presa de decisions en temps real, utilitzant criteris mesurables com la càrrega acumulada o el cost estimat de preparació. La motivació del treball rau, per tant, en analitzar fins a quin punt l'aplicació d'estratègies de repartiment de càrrega, inspirades en sistemes informàtics, pot contribuir a una gestió més equilibrada i eficient del servei en entorns de restauració.

1.3 Problema a resoldre

El problema que es vol abordar en aquest treball és la millora de l'eficiència i del repartiment equitatiu de les tasques associades a les comandes en bars on tots els cambrers poden preparar qualsevol producte de la carta. Aquest repte esdevé especialment rellevant durant les hores punta, quan una assignació poc adequada de les tasques pot generar saturacions, desequilibris en la càrrega de treball i un augment dels temps de servei.

En aquest context, definir un mecanisme eficaç de repartiment de les comandes és clau per garantir un flux de treball més equilibrat, eficient i sostenible per a l'equip de cambrers.

Figura 1: Bar desorganitzat

1.4 Objectius del treball

L'objectiu principal d'aquest Treball de Fi de Grau és dissenyar i analitzar un algoritme de repartiment de comandes que permeti millorar l'eficiència en quant a temps i l'equilibri de la càrrega de treball entre cambrers en un entorn de bar amb alta concurrència.

De manera més específica, els objectius del treball són els següents:

- Analitzar el funcionament real d'un bar i identificar els factors que influeixen en la distribució de les tasques.
- Implementar un entorn de simulació que permeti reproduir escenaris realistes de servei.
- Dissenyar diferents estratègies algorísmiques per a l'assignació de comandes basades en criteris definits explícitament (càrrega, cost i heurístiques de context) i parametritzats mitjançant constants de control (llindars i pesos).
- Comparar el comportament de les diferents variants de l'algoritme principalment mitjançant mètriques quantitatives obtingudes en simulació controlada.
- Avaluar l'impacte de cada estratègia en el temps de preparació i en la dispersió de la càrrega acumulada entre cambrers, intentant trobar un equilibri entre aquests dos.

1.5 Abast i limitacions

Inicialment, l'abast d'aquest Treball de Fi de Grau contemplava el desenvolupament d'una aplicació completa per a la gestió de comandes en un bar, utilitzant tecnologies com React Native i Expo, amb l'objectiu que pogués ser utilitzada en un entorn real. Aquesta aplicació havia d'incloure funcionalitats com la creació i gestió de comandes, la selecció de taules, una pantalla de visualització de comandes dissenyada per minimitzar el nombre d'interaccions necessàries, sistemes de cobrament i facturació, així com l'obtenció d'estadístiques relacionades amb el funcionament del servei.

No obstant això, a mesura que el projecte va avançar, es va constatar que el desenvolupament complet d'aquest sistema implicava una complexitat elevada, tant a nivell tècnic com de disseny i validació, que podia comprometre la profunditat de l'anàlisi dins del marc temporal d'un Treball de Fi de Grau. En aquest context, es va considerar més adequat redefinir l'abast del projecte i centrar els esforços en un dels aspectes clau identificats: el repartiment eficient i equitatiu de la càrrega de treball entre cambrers.

En conseqüència, aquest TFG se centra principalment en el disseny, implementació i avaluació d'algoritmes de repartiment de comandes, amb especial atenció a la comparació de diferents estratègies i a l'anàlisi del seu comportament en situacions

d'alta concurrència. La validació del sistema es realitza mitjançant un entorn de simulació, que permet estudiar el comportament dels algoritmes de manera controlada i obtenir mètriques quantitatives sobre càrrega, temps de preparació i inactivitat.

Queden fora de l'abast d'aquest treball la implementació d'una aplicació completa orientada a l'usuari final, la seva integració amb sistemes comercials existents, la gestió real de pagaments i facturació, així com una avaluació exhaustiva de l'experiència d'usuari. Aquests aspectes es consideren línies de treball futur que podrien complementar i ampliar la proposta presentada en aquest TFG

1.6 Estructura del document

Aquest document s'organitza en diversos capítols. En el Capítol 2 es presenta l'estat de l'art i es revisen les solucions existents relacionades amb la digitalització de comandes i el repartiment de càrrega. El Capítol 3 analitza el problema i descriu el context real de funcionament d'un bar, així com els requisits del sistema.

En el Capítol 4 es presenta la validació contextual mitjançant qüestionariS, utilitzada per reforçar les decisions de disseny. El Capítol 5 descriu el disseny del sistema i les estratègies algorísmiques implementades, mentre que el Capítol 6 detalla la implementació del sistema.

Finalment, el Capítol 7 exposa la metodologia experimental i els resultats obtinguts, i el Capítol 8 recull les conclusions del treball i les possibles línies de treball futur.

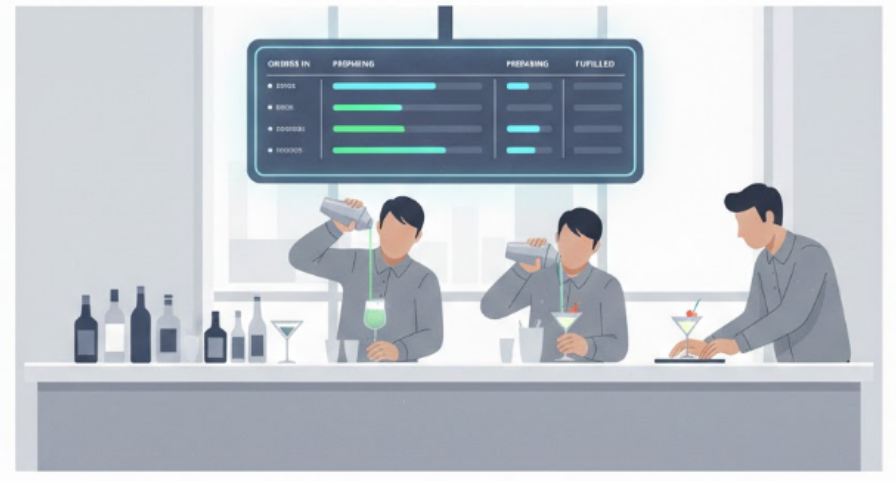


Figura 2: Bar organitzat amb tecnologia

2 Estat de l'art

2.1 Digitalització en entorns de restauració

Durant els darrers anys, el sector de la restauració ha adoptat de manera creixent solucions digitals per millorar l'eficiència operativa i reduir els errors en la gestió del servei. La digitalització sol començar per la presa estructurada de comandes i la seva transmissió immediata a la barra o cuina, substituint processos tradicionals basats en paper o verbals [9, 14].

En aquest context, els sistemes de punt de venda i els dispositius mòbils de presa de comandes permeten introduir l'ordre directament al sistema en el mateix moment en què es pren, reduint confusions i agilitzant el flux de treball. Diverses fonts del sector reporten millores en velocitat de servei i precisió quan es digitalitza la presa de comandes, especialment en hores punta [9, 14].

Paral·lelament, s'ha estès l'ús de sistemes de visualització i gestió d'ordres a la cuina (Kitchen Display Systems, KDS), que substitueixen els tiquets impresos i centralitzen les comandes en pantalles en temps real. Aquests sistemes faciliten la priorització, el seguiment d'estats i la coordinació entre sala i barra o cuina, minimitzant la dependència d'interrupcions i comunicació verbal [16, 15, 5].

Finalment, arran de la pandèmia, s'ha popularitzat l'ús de menús digitals i sistemes basats en codis QR, tant per consultar la carta com per facilitar comandes sense contacte. La literatura recent estudia la percepció i satisfacció dels clients amb aquests sistemes, així com els seus efectes sobre l'experiència d'usuari [11, 12]. Tot i això, també s'ha observat rebuig en determinats contextos i una tendència cap a solucions híbrides (paper + digital) per no degradar l'experiència d'hospitalitat en restaurants de servei complet [13].

2.2 Sistemes de gestió de comandes

Els sistemes de gestió de comandes moderns acostumen a integrar tres peces principals: (1) presa de comandes, (2) enviament i visualització a barra/cuina (KDS o impressió automàtica), i (3) tancament del servei (pagament, facturació i dades). El seu principal impacte operatiu és reduir errors de transcripció, accelerar el temps de comunicació entre sala i barra/cuina, i millorar el seguiment d'estats [16, 15].

Malgrat aquests avantatges, moltes solucions comercials se centren sobretot en el *registre* i la *visualització* de comandes, però no resolen de manera explícita el problema de la distribució interna de tasques quan tots els treballadors poden executar qualsevol producte. En entorns de bar amb alta concurrència, aquesta decisió (qui prepara què i quan) pot continuar fent-se de manera informal o basada en hàbits, i pot generar desequilibris de càrrega i saturacions puntuals fins i tot amb comandes digitalitzades [16].

Això justifica l'interès d'estudiar estratègies d'assignació dinàmica que, mantenint la digitalització de la comanda, incorporin un criteri algorísmic per repartir el

treball de forma més equitativa i eficient.

2.3 Algorismes de repartiment de càrrega

El problema de repartir tasques entre múltiples recursos apareix en molts àmbits (sistemes distribuïts, cues o planificació). En computació, és habitual modelar l'assignació de feines a servidors mitjançant polítiques de *load balancing* i *routing*. Aquestes polítiques proporcionen un marc útil per inspirar solucions en entorns de restauració on cal decidir com distribuir productes entre cambrers en temps real.

Una política simple és el *round-robin*, que assigna tasques de manera cíclica. Tot i aportar certa equitat, no té en compte que les tasques poden tenir durades molt diferents. Una alternativa és *join-the-shortest-queue* (JSQ) o *Least-Load*, que envia cada nova tasca al recurs amb la cua més curta o menor càrrega observada, i que està àmpliament estudiada en termes de rendiment i temps de resposta [2, 4].

També existeixen estratègies probabilístiques com el *power of two choices*, on per cada tasca es consideren dos candidats i s'escull el menys carregat. Aquesta idea és rellevant perquè aconsegueix millores importants en distribució de càrrega amb un cost de decisió baix, i ha estat analitzada extensament a la literatura [8, 7].

2.3.1 Round-robin (assignació cíclica)

L'algoritme *round-robin* (RR) és una estratègia d'assignació molt simple basada en un patró cíclic. En l'àmbit de la planificació de processos, RR executa cada procés durant un *time slice* (quantum) i després passa a la següent de la cua, repetint el cicle fins que tots els processos finalitzen. En essència, es tracta d'un mecanisme de torns que garanteix que cada procés rep oportunitats d'execució de manera regular.

Traslladat al context d'aquest TFG, RR es pot utilitzar com a *base* per repartir la preparació de productes d'una comanda entre cambrers: es manté un índex que apunta al "següent cambrer" i, per cada producte a assignar, s'assigna al cambrer apuntat i s'avança l'índex de forma circular. Així, si hi ha N cambrers, l'assignació del producte k segueix:

$$i = k \bmod N$$

on i és l'índex del cambrer assignat.

Pseudocodi

```
i = 0
for producte in comanda:
    assigna(producte, cambrer[i])
    i = (i + 1) % N
```

Avantatges

- **Simplicitat i cost baix:** decisió en temps constant ($O(1)$) i fàcil d'implementar.
- **Equitat en el nombre d'assignacions:** tendeix a repartir un nombre similar de productes a cada cambrer.
- **Bona línia base:** és útil com a punt de comparació contra estratègies més "intel·ligents".

Limitacions

- **No té en compte el cost dels productes:** si els temps de preparació són heterogenis, RR pot generar càrregues molt desequilibrades (pocs productes "cars" poden saturar un cambrer).
- **No considera l'estat actual:** no incorpora informació de càrrega acumulada, cues o inactivitat.

Nota sobre *round-robin* com a planificació temporal Cal distingir el *round-robin* utilitzat en aquest treball (assignació cíclica de productes a cambrers) del *round-robin scheduling* clàssic en sistemes operatius, basat en llesques de temps (*time slices*). Tot i que es podria plantejar una analogia de planificació temporal, en el context d'un bar seria poc realista: implicaria interrompre tasques i potencialment repartir la preparació d'un mateix producte entre diversos cambrers, generant canvis constants, desplaçaments i sobrecost operatiu. Per aquest motiu, el treball es limita al *round-robin* com a política d'assignació discreta de tasques, i no com a planificació amb preempció temporal.

2.3.2 Least-load

L'estratègia *least-load* (o *least-loaded*) assigna cada nova tasca al recurs que, en aquell moment, presenta la menor càrrega. En entorns de cues, una forma clàssica d'aquesta idea és *Join-the-Shortest-Queue* (JSQ), on una petició entrant es dirigeix al servidor amb menys feines pendents. L'objectiu és reduir la probabilitat que un recurs quedi saturat mentre d'altres romanen infrautilitzats, millorant el temps de resposta i l'equilibri global del sistema [2].

En el context d'aquest TFG, on els cambrers poden preparar qualsevol producte, *least-load* es pot definir mitjançant una funció de càrrega L_i per a cada cambrer i, i assignar cada producte al cambrer amb L_i mínima. Aquesta càrrega pot representar, per exemple, (a) el nombre de productes pendents, (b) la suma del cost estimat dels productes assignats (treball pendent), o (c) una combinació ponderada de càrrega acumulada i treball pendent.

Definició (forma genèrica) Siguin N cambrers i una càrrega acumulada L_i per a cada cambrer i. Per a cada producte p amb cost estimat $c(p)$, l'assignació és:

$$i^* = \arg\min_{i \in \{1,\dots,N\}} L_i, \qquad \text{i s'actualitza } L_{i^*} \leftarrow L_{i^*} + c(p).$$

Pseudocodi

```
for producte p in comanda:
    i = argmin_i(load[i])
    assigna(p, cambrer[i])
    load[i] += cost(p)
```

Avantatges

- **Adaptatiu:** reacciona a situacions on els costos de les tasques són desiguals (p. ex. còctels ràpids vs elaborats).
- **Reducció de saturacions:** tendeix a evitar que un cambrer acumuli massa feina mentre altres estan relativament lliures [2, 3].
- **Interpretabilitat:** la regla de decisió és clara i justificable (assignar al menys carregat).

Limitacions

- **Necessita informació de càrrega en temps real:** si L_i no es mesura bé (o el cost és una estimació dolenta), l'assignació pot ser subòptima.
- **Cost de decisió:** requereix comparar recursos (típicament $O(N)$ per producte), tot i que en un bar N acostuma a ser petit.
- **No s'optimitza el temps**: És un algoritme que prioritza balancejar la càrrega i no sacrifica res de càrrega per intentar augmentar temps

En resum, *least-load* és una política interessant per tenir com a base, tendeix a equilibrar la càrrega acumulada entre cambrers, però no està dissenyada específicament per minimitzar el temps de finalització de cada comanda.

2.3.3 Power of two choices

L'estratègia coneguda com *power of two choices* és una variant probabilística del repartiment de càrrega que ofereix una millora notable respecte a l'assignació completament aleatòria amb un cost de decisió molt baix. La idea és simple: en lloc d'avaluar tots els recursos disponibles, per a cada tasca es seleccionen aleatòriament dos candidats i s'assigna la tasca al candidat menys carregat [8].

Aquesta tècnica va ser popularitzada en el model de *balanced allocations* (boles i cubs) i posteriorment analitzada en profunditat en models de *load balancing* i cues. Els resultats mostren que passar d'una sola elecció aleatòria a dues eleccions i triar el millor dels dos redueix dràsticament els desequilibris de càrrega, sovint amb millores qualitatives (no només quantitatives) [1, 8].

Regla d'assignació Sigui L_i la càrrega del recurs i en el torn. Per a cada nova tasca:

1. es trien dos candidats a i b de forma aleatòria,
2. s'assigna la tasca al que tingui menor càrrega: $\arg\min\{L_a, L_b\}$,
3. s'actualitza la càrrega del recurs escollit.

Pseudocodi

```
for producte p in comanda:
    a = random_waiter()
    b = random_waiter()
    i = a if load[a] <= load[b] else b
    assigna(p, cambrer[i])
    load[i] += cost(p)
```

Avantatges

- **Cost de decisió baix:** només compara dos candidats (cost constant), en lloc d'avaluar tots els cambrers com en un *least-load* pur.
- **Bona distribució de càrrega:** redueix la probabilitat d'assignar tasques a recursos ja saturats i tendeix a equilibrar el sistema de forma robusta [8, 1].

Limitacions

- **Depèn de com es defineix la càrrega:** si L_i (o $cost(p)$) és una estimació dolenta, la regla pot prendre decisions subòptimes.
- **No garanteix l'òptim global:** triar el millor de dos és una aproximació; amb $d > 2$ es pot millorar, però amb rendiments decreixents.
- **No s'optimitza el temps**: És un algoritme que prioritza balancejar la càrrega i no sacrifica res de càrrega per intentar augmentar temps

En resum, *power of two choices* és una alternativa intermèdia entre *round-robin* (massa simple) i *least-load* (més costós), oferint un bon compromís entre eficiència de decisió i qualitat del repartiment. Tot i que en haver-hi normalment pocs cambrers en un bar el cost de least-load no és un problema.

2.4 Limitacions de les solucions existents

La digitalització de comandes (POS, dispositius de presa de comandes i pantalles de visualització) resol amb eficàcia problemes clàssics com els errors de transcripció, la pèrdua d'informació i la manca de traçabilitat del servei. A més, moltes solucions comercials incorporen un model de gestió de personal basat en *usuaris, grups i permisos*, amb l'objectiu que cada treballador només pugui accedir a les funcions necessàries segons el seu rol (per exemple, cambrer, encarregat o administrador) [6, 10].

Tot i això, en entorns de bar on **tots els cambrers de barra poden preparar qualsevol producte**, el problema principal no és tant "qui té permís per fer què", sinó **com repartir dinàmicament el treball** quan arriben moltes comandes i els productes tenen costos de preparació diferents. En aquests casos, els sistemes existents acostumen a donar suport a la *captura* i *visualització* de la comanda, però la decisió fina de **quin cambrer prepara cada producte i en quin moment** continua recaient en criteris informals (hàbits, intuïció, pressa, "qui està més lliure") i pot derivar en desequilibris de càrrega o saturacions puntuals.

En conseqüència, tot i que els POS/KDS aporten una base operativa sòlida, es detecta la manca d'un mecanisme explícit orientat a **assignar tasques de manera eficient i equitativa en temps real** quan els recursos són intercanviables. Aquesta limitació justifica l'interès d'estudiar estratègies algorísmiques de repartiment de càrrega adaptades al context del bar.

Com a exemple, solucions comercials àmpliament utilitzades com *Lightspeed Restaurant* o *NCR Aloha* ofereixen eines robustes per a la presa de comandes, el control d'accés al sistema i la definició de permisos segons el rol del treballador. Aquests sistemes permeten determinar qui pot accedir al POS, modificar comandes o gestionar el cobrament, aportant control i seguretat operativa. Tanmateix, aquests enfocaments no estan pensats per resoldre de manera automàtica la micro-assignació de tasques quan els cambrers són intercanviables. En entorns de barra on tots poden preparar qualsevol producte, la decisió de quin cambrer s'encarrega de cada beguda continua depenent principalment de criteris manuals o informals. Això fa que, malgrat disposar d'un sistema digital avançat, el repartiment de la càrrega de treball segueixi sent un punt crític no resolt per les solucions existents. [6, 10].

2.5 Justificació de la proposta

La justificació de la proposta presentada en aquest Treball de Fi de Grau es fonamenta en la manca d'una solució específica que abordi de manera directa el problema plantejat. Tot i l'existència de sistemes digitals de gestió de comandes i de models teòrics d'assignació de tasques en altres dominis, no s'ha identificat cap enfocament que resolgui explícitament la distribució dinàmica de productes entre cambrers intercanviables en un entorn de bar amb alta concurrència.

En aquest context, el repte principal no és la digitalització de la comanda, sinó la decisió de com repartir les tasques de manera justa i eficient quan tots els cam-

brers poden assumir qualsevol producte i els temps de preparació són heterogenis. Aquesta decisió continua recaient habitualment en criteris informals o manuals, fet que pot derivar en desequilibris de càrrega i saturacions puntuals.

Per aquest motiu, resulta pertinent estudiar i analitzar un algoritme de repartiment que, inspirant-se en estratègies clàssiques de balanceig de càrrega, permeti organitzar el treball dels cambrers de forma equitativa, alhora que s'optimitzen els temps de preparació de les comandes. La proposta d'aquest TFG pretén, doncs, aportar una solució algorísmica aplicada a un problema real, avaluant diferents estratègies i analitzant el seu comportament en un entorn controlat mitjançant simulació.

3 Anàlisi del problema

3.1 Funcionament real d'un bar en el context plantejat

En el context considerat en aquest treball, el funcionament d'un bar segueix un flux de treball relativament estructurat, però amb un grau elevat de variabilitat segons el volum de clients i el moment del servei. De manera simplificada, el procés es pot descriure mitjançant els següents passos:

1. El client s'asseu a la taula i consulta la carta.
2. El cambrer de terrassa s'acosta a la taula i pren la comanda.
3. La comanda s'introdueix en un sistema digital, seleccionant els productes sol·licitats pels clients.
4. Un cop completada, la comanda es processa i s'envia a la barra de manera immediata.
5. A la barra, la comanda es visualitza en una pantalla compartida, on es mostra l'assignació de cada producte als diferents cambrers.
6. Cada cambrer de barra prepara els productes que li han estat assignats i els col·loca a la safata corresponent a la comanda.
7. Quan tots els productes estan finalitzats, el cambrer de terrassa recull la safata.
8. Finalment, els productes són entregats als clients.

Aquest flux reflecteix el funcionament habitual d'un bar on els cambrers de barra són intercanviables i poden preparar qualsevol producte, i posa de manifest la importància d'un sistema d'assignació eficient per evitar desajustos de càrrega, especialment en situacions d'alta concurrència.

3.2 Actors del sistema

En el sistema plantejat intervenen diversos actors amb rols clarament diferenciats, cadascun dels quals participa en una part concreta del flux de treball. Identificar aquests actors permet delimitar responsabilitats i entendre millor sobre quins elements actua l'algoritme de repartiment de càrrega.

Clients Els clients són els actors externs que originen les comandes. La seva interacció amb el sistema és indirecta, ja que no accedeixen al sistema digital intern, però determinen el volum i la composició de les comandes, i per tant influeixen directament en la càrrega de treball generada.

Cambrer de terrassa El cambrer de terrassa és responsable de la presa de comandes a taula i de la seva introducció al sistema digital. A més, s'encarrega de recollir les comandes un cop estan preparades a la barra i de lliurar-les als clients. Aquest actor no participa en el procés d'assignació interna dels productes, sinó que actua com a intermediari entre clients i barra.

Cambrers de barra Els cambrers de barra són els actors encarregats de preparar els productes de les comandes. En el context d'aquest treball, tots els cambrers de barra es consideren intercanviables, és a dir, qualsevol d'ells pot preparar qualsevol producte de la carta. Aquesta característica és clau, ja que elimina una assignació basada en rols fixos i fa necessari un mecanisme que reparteixi les tasques de manera eficient i equilibrada.

Sistema El sistema digital actua com a actor central de coordinació. És responsable de rebre les comandes, visualitzar-les a la barra i, especialment, decidir l'assignació de cada producte als cambrers de barra segons l'estratègia de repartiment utilitzada. En aquest sentit, el sistema no és només un suport passiu, sinó que pren decisions que afecten directament la distribució de la càrrega de treball.

3.3 Flux de comandes i productes

En el context d'aquest treball, una **comanda** es defineix com el conjunt de productes que són sol·licitats simultàniament per una mateixa taula. Tots els productes associats a una comanda es preparen conjuntament i es lliuren de manera unificada, habitualment mitjançant una única safata.

Des del punt de vista del sistema, una comanda pot trobar-se en un dels següents estats:

- **Pendent**: la comanda ha estat creada però encara no s'ha iniciat la preparació de cap dels seus productes.

- **En preparació**: almenys un dels productes associats a la comanda s'està preparant.
- **Preparada**: tots els productes de la comanda han estat finalitzats i estan llestos per ser entregats. (Estat de tots els productes = preparat)
- **Entregada**: la comanda ha estat recollida i servida a la taula corresponent.

D'altra banda, un **producte** es defineix com un element individual de la carta que, conjuntament amb altres productes, pot formar part d'una comanda. Una mateixa comanda pot contenir un o més productes, i aquests es consideren unitats de treball independents a efectes d'assignació i preparació.

Cada producte pot trobar-se en un dels següents estats:

- **Pendent**: el producte ha estat sol·licitat però encara no s'ha assignat ni iniciat la seva preparació.
- **En preparació**: el producte està sent elaborat per un cambrer de barra.
- **Preparat**: el producte ha estat completat i es troba llest per ser inclòs a la safata de la comanda.

A més, cada producte té associat un **cost de preparació**, que representa una estimació de l'esforç o temps necessari per a la seva elaboració. La definició i utilització d'aquest cost es detallarà en apartats posteriors, ja que constitueix un dels elements clau en el disseny dels algorismes de repartiment de càrrega proposats en aquest treball.

En el model proposat en aquest treball, s'assumeix que una comanda només pot iniciar el seu estat de **preparació** quan tots els cambrers als quals s'han assignat productes dins d'aquesta comanda es troben disponibles.

3.4 Factors que influeixen en la càrrega de treball

La càrrega de treball associada a la preparació de comandes en un bar no depèn únicament del nombre de productes assignats a cada cambrer, sinó també de diversos factors qualitatius i temporals que influeixen directament en l'esforç necessari per completar cada tasca. En el model considerat en aquest treball, s'han tingut en compte els següents factors principals:

- **Temps base de preparació:** cada producte de la carta disposa d'un temps de preparació estimat, que representa el temps necessari per elaborar-lo en condicions normals.
- **Complexitat del producte:** el cost de preparació d'un producte pot variar en funció de la seva dificultat, del nombre de passos necessaris i de la quantitat d'ingredients implicats. Productes amb més passos o més ingredients tendeixen a tenir un cost base més elevat.

- **Reposició d'ingredients:** en determinats casos, la preparació d'un producte pot requerir la reposició prèvia d'algun ingredient (per exemple, gel, fruita tallada o begudes esgotades). Aquest fet pot incrementar el temps real de preparació. En el model, aquest increment es contempla mitjançant una probabilitat que el temps de preparació augmenti respecte al valor base.
- **Preparació simultània de productes:** quan un mateix cambrer prepara diversos productes de manera consecutiva o simultània, es considera que existeix una certa economia d'escala. En particular, el cost de preparació del segon producte i dels següents es redueix respecte al cost del primer, ja que part del treball (preparació d'estris, accés a ingredients, organització de l'espai) ja s'ha realitzat. En el model, aquest efecte es reflecteix assumint que el cost del segon producte i dels posteriors és aproximadament la meitat del cost individual inicial.

Aquests factors permeten definir una estimació més realista de la càrrega de treball associada a cada producte i, en conseqüència, millorar la representació del comportament del sistema en situacions de concurrència elevada.

3.5 Requeriments funcionals

El sistema desenvolupat en aquest Treball de Fi de Grau té com a objectiu principal donar suport al repartiment de comandes en un entorn de bar mitjançant un enfocament algorísmic. A continuació es detallen els requeriments funcionals principals del sistema:

- **Gestió de cambrers:** el sistema ha de permetre definir un conjunt de cambrers disponibles, tots ells amb la capacitat de preparar qualsevol producte de la carta.
- **Gestió de comandes:** el sistema ha de poder crear comandes associades a una taula, formades per un conjunt de productes demanats simultàniament.
- **Gestió de productes:** cada producte ha de disposar d'un cost de preparació estimat i d'un estat (pendent, en preparació o preparat).
- **Assignació de productes a cambrers:** el sistema ha d'assignar automàticament cada producte d'una comanda a un cambrer segons l'algoritme de repartiment seleccionat.
- **Control d'estats de les comandes:** una comanda ha de poder evolucionar entre els estats pendent, en preparació, preparada i entregada, en funció de l'estat dels seus productes.
- **Execució d'algoritmes alternatius:** el sistema ha de permetre executar diferents estratègies d'assignació (round-robin, least-load, power of two choices i l'algoritme proposat) sobre el mateix conjunt de dades.

- **Simulació del servei:** el sistema ha de permetre simular escenaris de servei amb múltiples comandes i cambrers, sense necessitat d'interacció humana.
- **Recollida de mètriques:** durant la simulació, el sistema ha de registrar mètriques com el temps total de preparació, la càrrega acumulada per cambrer i el temps d'inactivitat.

3.6 Requeriments no funcionals

A més de les funcionalitats descrites, el sistema ha de complir una sèrie de requisits no funcionals que garanteixin la seva utilitat com a eina d'anàlisi i validació algorísmica:

- **Rendiment:** el sistema ha de ser capaç de processar múltiples comandes i assignacions de productes amb un cost computacional baix, de manera que la simulació sigui eficient fins i tot amb un volum elevat de comandes.
- **Escalabilitat:** el disseny del sistema ha de permetre augmentar el nombre de cambrers i comandes sense necessitat de modificar l'arquitectura bàsica.
- **Determinisme i repetibilitat:** donat un mateix conjunt de paràmetres inicials, el sistema ha de permetre repetir simulacions per comparar el comportament dels diferents algoritmes en condicions equivalents.
- **Simplicitat:** el sistema no ha de dependre de components externs complexos ni d'infraestructures reals, ja que l'objectiu principal és l'anàlisi algorísmica i no el desplegament en producció.
- **Claredat de resultats:** les dades i mètriques generades han de ser fàcilment interpretables, permetent una comparació clara entre estratègies de repartiment.
- **Independència de la interfície d'usuari:** el funcionament del sistema i dels algoritmes és independent de qualsevol interfície operativa. Tot i disposar d'una visualització del simulador per facilitar l'anàlisi i la comprensió del comportament del sistema, aquesta no influeix en la presa de decisions ni en l'execució dels algoritmes, que poden funcionar de manera completament autònoma.

Aquests requisits defineixen el marc en què s'ha dissenyat i validat el sistema, i serveixen de base per a les decisions algorísmiques que es presenten en els capítols posteriors.

4 Validació contextual mitjançant qüestionari

4.1 Objectiu del qüestionari

L'objectiu del qüestionari és validar de manera qualitativa i contextual les hipòtesis de partida d'aquest treball, especialment pel que fa als problemes operatius detectats en la gestió de comandes i en el repartiment de tasques dins d'un bar amb alta concurrència.

El qüestionari no té com a finalitat realitzar una anàlisi estadística exhaustiva, sinó recollir percepcions reals de persones amb experiència directa o indirecta en entorns de restauració, amb l'objectiu de reforçar les decisions de disseny del sistema i dels algorismes de repartiment de càrrega proposats.

4.2 Metodologia

El qüestionari es va dissenyar com una eina breu i directa, formada principalment per preguntes tancades i una pregunta oberta final. Les preguntes se centren en identificar problemes habituals en la presa de comandes, la comunicació amb la barra i la percepció de possibles solucions digitals.

Els participants van respondre el qüestionari de manera anònima. El perfil dels enquestats inclou persones que han treballat o treballen en bars o restaurants, així com clients habituals d'establiments de restauració, fet que permet obtenir una visió combinada des del punt de vista operatiu i d'experiència d'usuari.

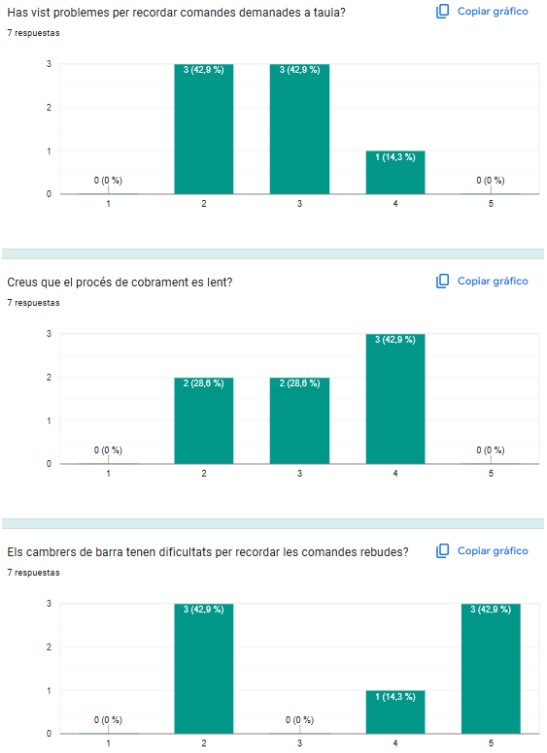
Els resultats es van recollir mitjançant un formulari digital i posteriorment es van exportar en forma de gràfics, que es mostren en els apartats següents com a suport visual de l'anàlisi realitzada.

4.3 Resultats principals

Els resultats del qüestionari mostren una tendència clara cap a la percepció de problemes en la gestió de comandes i la coordinació entre terrassa i barra, especialment en situacions d'alta concurrència.

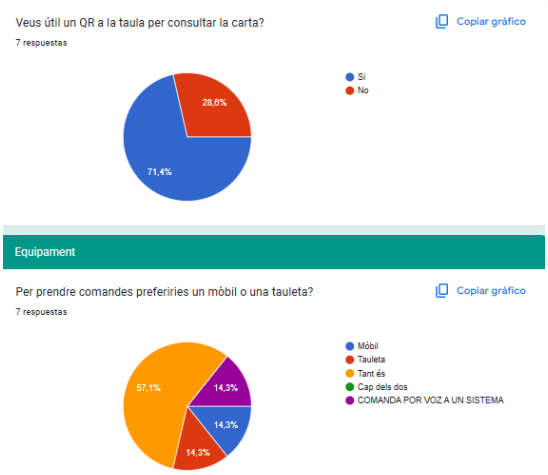
Una part significativa dels participants indica haver observat dificultats per recordar comandes, errors de comunicació i moments de desorganització durant hores punta. Així mateix, una majoria considera que un sistema digital podria ajudar a reduir aquests problemes i millorar l'eficiència del servei.

A la Figura 3 es mostren alguns dels resultats obtinguts, on es pot observar una valoració generalment positiva de solucions com la presa digital de comandes, la visualització centralitzada a la barra i l'ús de pantalles per indicar l'estat dels productes.



(a) Valoració de problemes

(b) Percepció de solucions digitals



(c) Digitalització de la presa de comandes

(d) Valoració sobre solucions pels cambrers de barra

Figura 3: Resultats principals del qüestionari de validació contextual

4.4 Conclusions i implicacions

Els resultats del qüestionari reforcen les hipòtesis plantejades en aquest treball i permeten identificar amb claredat quins aspectes del funcionament d'un bar generen més fricció en situacions d'alta concurrència. En particular, es constata que els problemes principals no es troben únicament en la presa de comandes, sinó en la coordinació interna i el repartiment de tasques a la barra quan el volum de servei augmenta.

És rellevant destacar que les preguntes relacionades amb la necessitat de millorar la visualització de les comandes a la barra i amb l'ús de sistemes que ajudin els cambrers de barra a organitzar-se obtenen una acceptació unànime per part dels participants. Aquest consens posa de manifest que, més enllà de digitalitzar la presa de comandes, existeix una demanda clara de solucions que donin suport explícit a la distribució i assignació de productes entre cambrers.

Aquest fet ha estat determinant en la definició de l'enfocament del treball. En lloc de centrar-se exclusivament en la interfície o en la captura de comandes, ja que és un camp molt explorat, aquest TFG proposa abordar el problema des d'un punt de vista algorísmic, estudiant com assignar els productes d'una comanda de manera eficient i equitativa quan els cambrers són intercanviables i poden preparar qualsevol producte.

Així, el qüestionari no només valida la rellevància del problema abordat, sinó que justifica de manera directa la decisió de focalitzar el treball en el disseny i avaluació d'algoritmes de repartiment de càrrega, orientats a millorar l'organització del treball a la barra en temps real.

5 Disseny del sistema

5.1 Arquitectura general

El sistema desenvolupat en aquest treball té un objectiu principalment **experimental i algorísmic**: permetre aplicar estratègies d'assignació de productes a cambrers i comparar-ne el comportament mitjançant simulació i mètriques quantitatives. Per aquest motiu, l'arquitectura es construeix de manera modular, separant (i) la gestió de dades i persistència, (ii) el motor de decisió d'assignació i (iii) l'entorn de simulació i visualització.

Visió global A nivell global, el sistema es pot entendre com la combinació de dos fluxos complementaris:

- **Flux operatiu (backend + base de dades)**: registra comandes i línies de producte, i aplica l'assignació automàtica en el moment de crear la comanda.
- **Flux experimental (simulador)**: executa escenaris controlats amb múltiples comandes i cambrers, aplicant les mateixes polítiques d'assignació per

recollir mètriques i comparar algoritmes.

Components principals L'arquitectura es divideix en els mòduls següents:

- **Persistència i accés a dades**: una base de dades SQLite emmagatzema entitats com cambrers, productes, taules i comandes. L'accés a dades es centralitza mitjançant operacions *CRUD* que encapsulen consultes habituals (p. ex. obtenir cambrers disponibles, productes i informació contextual com els cambrers assignats a la darrera comanda).

- **Servei d'assignació de comandes**: el servei `assign_waiters_to_order` actua com a cas d'ús principal del backend. A partir de les línies d'una comanda, construeix una representació agregada (producte → quantitat) i invoca el motor d'assignació per obtenir una distribució cambrer–producte que posteriorment es persisteix.

- **Motor d'assignació (capa algorísmica)**: el motor encapsula la política de decisió que, donat l'estat del sistema i una comanda, determina a quin cambrer s'assigna cada producte (o unitat de producte). La decisió es basa, principalment, en una variable de càrrega `load` associada a cada cambrer i emmagatzemada a la base de dades. En aquest treball, `load` es defineix com la **càrrega acumulada assignada**: cada vegada que s'assigna una unitat de producte a un cambrer, s'incrementa el seu valor sumant-hi el cost estimat del producte (ponderat per la quantitat). D'aquesta manera, criteris com *least-load* es poden implementar seleccionant el cambrer amb menor càrrega acumulada, amb l'objectiu de repartir el volum total de treball de manera més equitativa.

  A més de la càrrega acumulada, el motor pot utilitzar informació contextual per ajustar el repartiment. En particular, es considera el conjunt de cambrers que han intervingut a la darrera comanda (`last_order_waiters`) com a heurística per **evitar repetir sistemàticament els mateixos cambrers** en comandes consecutives i afavorir una distribució més homogènia en períodes de molta concurrència.

  La política concreta d'elecció (p. ex. *round-robin*, *least-load* o variants heurístiques orientades a reduir duplicacions dins la mateixa comanda) s'implementa com a funcions al mòdul `load.py`, que reben l'estat de càrrega i el context de la comanda (p. ex. assignacions ja realitzades dins la mateixa comanda) i retornen el cambrer escollit per a la següent assignació.

  Cal remarcar que la variable `load` és acumulativa i s'utilitza com a criteri d'equitat en el repartiment; la modelització del temps i la disponibilitat dels cambrers es tracta en l'entorn de simulació i en la definició de mètriques.

- **Aplicació de l'assignació i consistència**: un cop obtinguda la distribució, el servei desa a la base de dades la correspondència de cada grup de producte amb cada cambrer o cambrers.

- **Simulador i recollida de mètriques**: de manera independent del backend, el simulador executa escenaris de servei, actualitza l'estat temporal (disponibilitat i evolució de càrrega) i registra mètriques com el temps de preparació, la càrrega acumulada per cambrer i el temps d'inactivitat. La visualització és un component auxiliar que facilita la interpretació del comportament, però no participa en la presa de decisions.

Flux de dades en el backend Quan una comanda es registra al sistema, el servei d'assignació (1) carrega dades de domini via CRUD, (2) construeixi l'agregació de la comanda, (3) executa la política d'assignació seleccionada i (4) persisteix el resultat associant cada producte als cambrers. Aquest flux garanteix traçabilitat: cada producte queda vinculat explícitament a un responsable.

Flux experimental En els experiments, les mateixes idees d'assignació es traslladen a un entorn de simulació que permet controlar el nombre de comandes, els instants d'arribada i els temps de preparació. D'aquesta manera, es poden comparar estratègies en condicions equivalents i obtenir resultats quantitatius reproduïbles. Per garantir comparabilitat entre execucions, la càrrega `load` de cada cambrer es reinicia a 0 l'inici de cada experiment.

5.2 Model de dades

Propòsit El model de dades d'aquest projecte té com a objectiu emmagatzemar la informació necessària per **registrar comandes** formades per múltiples productes i donar suport al procés de **distribució de tasques** entre cambrers. Cada producte disposa d'un **cost** associat, utilitzat com a estimació de càrrega en el motor d'assignació.

A nivell de detall, el model permet que **cada unitat o línia de producte dins d'una comanda** tingui assignat un cambrer responsable i un **estat de preparació** (pendent, en preparació o preparat). De manera anàloga, les comandes mantenen un **estat global** (pendent, en preparació, preparada o entregada) que resumeix el progrés del conjunt.

Els productes poden pertànyer a un **grup**, que s'utilitza per representar restriccions operatives rellevants (per exemple, el punt físic de preparació o límits de preparació simultània segons el tipus de producte). Finalment, cada cambrer disposa d'una variable `load` que representa la **càrrega acumulada assignada durant el torn actual**, i que s'incrementa a mesura que se li assignen productes, d'acord amb el seu cost. A la simulació, cada execució representa un torn.

Entitats El model es compon d'un conjunt d'entitats operatives, que intervenen directament en el flux de comandes i l'assignació de tasques, i d'entitats de context, que representen informació auxiliar de l'entorn.

Entitats operatives (nucli)

- `Order` **(Comanda)**: representa una comanda associada a una taula i conté l'estat global del servei.
- `OrderProduct` **(Línia de comanda)**: unitat assignable que enllaça una comanda amb un producte, incorporant quantitat, cambrer responsable i estat de preparació.
- `Product` **(Producte)**: element de la carta amb un cost estimat (i, si escau, un preu) i una possible pertinença a un grup.
- `Waiter` **(Cambrer)**: recurs executor de tasques, amb una càrrega acumulada (`load`) utilitzada com a criteri d'equitat en l'assignació.

Entitats de context

- `Table` **(Taula)**: identifica la taula a la qual s'associa cada comanda i pot incloure informació auxiliar per a la visualització de les comandes a la simulació
- `Group` **(Grup)**: categoritza productes per representar restriccions operatives. Concretament, diferents productes del mateix grup es poden fer alhora, sempre que no es superi el limit de productes simultanis del grup. Això serà important per a l'algorisme prioritzar posar dos productes del mateix grup al mateix cambrer.

Relacions A la Figura 4 es mostra el diagrama entitat–relació (E/R) del model de dades i les cardinalitats principals entre entitats.

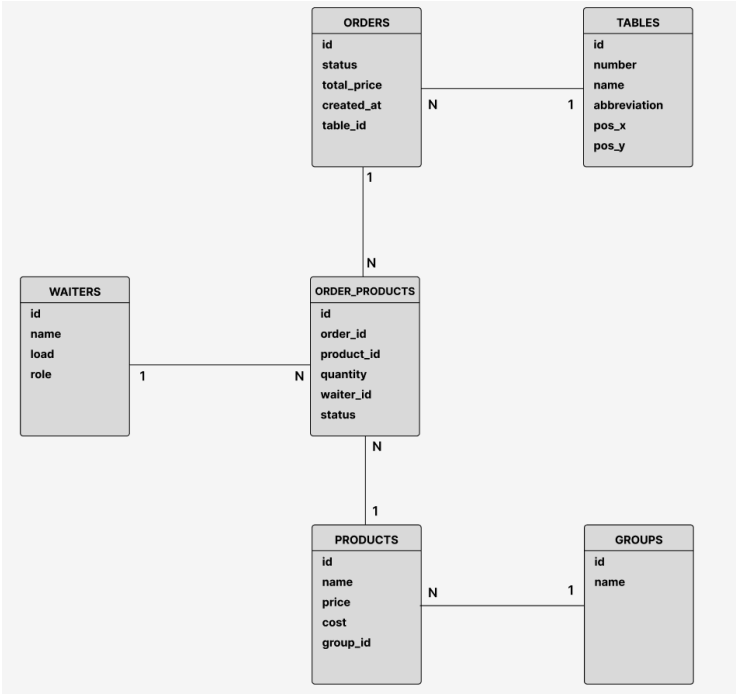


Figura 4: Diagrama entitat–relació (E/R) del model de dades

El nucli del model es basa en la representació explícita de les línies de comanda mitjançant l'entitat `OrderProduct`, que actua com a enllaç entre comandes, productes i cambrers. Les relacions principals són les següents:

- `Table` → `Order` **(1:N)**: una taula pot generar múltiples comandes al llarg del servei, mentre que cada comanda està associada a una única taula.
- `Order` → `OrderProduct` **(1:N)**: una comanda conté una o més línies de producte, cadascuna representant un producte concret i la seva quantitat.
- `Product` → `OrderProduct` **(1:N)**: un producte pot aparèixer en moltes línies de comanda, però cada línia referencia exactament un producte.
- `Waiter` → `OrderProduct` **(1:N)**: un cambrer pot tenir múltiples línies assignades al llarg del servei. L'atribut `waiter_id` pot ser nul abans de l'assignació, i es fixa quan s'aplica la política de repartiment.
- `Group` → `Product` **(1:N)**: cada grup pot contenir diversos productes, mentre que cada producte pertany, com a màxim, a un únic grup.

Aquesta estructura garanteix traçabilitat a nivell de producte, ja que permet registrar de manera explícita quin cambrer és responsable de cada línia de comanda i quin estat de preparació tenen els productes i les comandes en cada moment.

5.3 Representació de la càrrega de treball

Objectiu La representació de la càrrega és necessària per disposar d'una mesura quantitativa de la feina assignada a cada cambrer durant el torn (o execució de la simulació). Aquesta informació permet que els diferents algorismes d'assignació prenguin decisions basades en criteris mesurables, com ara seleccionar el cambrer amb menor càrrega acumulada per assignar-li el següent producte i així afavorir un repartiment més equilibrat.

Unitat de treball En el context d'aquest sistema, la unitat de treball que s'assigna és una **quantitat q d'un mateix producte** que el motor decideix assignar a un cambrer en una mateixa fase d'assignació. Operativament, aquesta unitat queda reflectida en una o més línies `OrderProduct`, la quantitat d'una línia de comanda es pot repartir entre cambrers diferents.

Cost de producte El cost de cada producte es defineix i s'emmagatzema a la taula `Product` de la base de dades. Aquest cost s'ha establert com una estimació de l'esforç de preparació i es determina a partir de factors com la dificultat del producte, el nombre de passos necessaris, la probabilitat d'haver de reposar ingredients i la quantitat d'ingredients implicats en la seva elaboració.

Càrrega acumulada per cambrer (`load`) La variable `load` representa la **càrrega acumulada assignada** a cada cambrer durant el torn (o durant l'execució d'un experiment). Aquesta càrrega s'actualitza cada vegada que el motor assigna a un cambrer un conjunt d'unitats d'un mateix producte, i s'incrementa a partir del cost base del producte, aplicant un descompte a les unitats addicionals per modelar una economia d'escala en la preparació consecutiva.

Sigui L_i la càrrega actual del cambrer i i $c(p)$ el cost base del producte p. Quan s'assignen $q \geq 1$ unitats del producte p al cambrer i, l'increment de càrrega es defineix com:

$$\Delta L_i(p, q) = c(p) + \frac{c(p)}{2}\,(q - 1),$$

i la nova càrrega passa a ser:

$$L_i \leftarrow L_i + \Delta L_i(p, q).$$

En el simulador, aquest increment de càrrega $\Delta L_i(p, q)$ s'interpreta també com una aproximació del **temps de servei** associat a la preparació: quan un cambrer rep una assignació, queda ocupat durant un interval proporcional a $\Delta L_i(p, q)$, i no pot rebre noves tasques fins que aquest temps finalitza.

Aquesta definició reflecteix que la primera unitat d'un producte incorpora el cost complet, mentre que les unitats addicionals tenen un cost marginal reduït (aproximadament la meitat), ja que part del treball i la preparació de l'espai ja s'han realitzat.

Exemple Si $q = 3$ i $c(p) = 6$, aleshores:

$$\Delta L_i = 6 + \frac{6}{2} \cdot 2 = 12,$$

i la nova càrrega és $L_i \leftarrow L_i + 12$.

A l'inici de cada simulació o torn, la variable `load` es reinicia a zero a cada cambrer per garantir comparacions reproduïbles entre estratègies.

Variables internes de càrrega dins del motor d'assignació:

- **Càrrega dins d'una comanda**: A més de la càrrega acumulada del cambrer (`load`), el motor utilitza una variable auxiliar `waiter_order_load`, que representa la càrrega assignada a cada cambrer **dins la comanda actual**. Aquesta variable s'empra en algunes estratègies com a criteri de desempat i de control de coherència: quan, dins d'una mateixa comanda, ja s'han utilitzat tots els cambrers disponibles (o no és possible evitar repeticions), el sistema selecciona el següent cambrer prioritzant aquell amb menor `waiter_order_load`, per evitar concentrar excessivament el pes de la comanda en un sol cambrer.
- **Historial recent**: la variable `last_order_waiters` recull els cambrers que han estat assignats a la comanda anterior. Aquesta informació s'utilitza com

a heurística per evitar repetir sistemàticament els mateixos cambrers en comandes consecutives i afavorir que, en la següent comanda, hi hagi el màxim nombre de cambrers disponibles tan aviat com sigui possible.

- **Productes ja assignats al cambrer en la comanda**: la variable `waiter_products` manté, per a la comanda actual, el conjunt de productes (i quantitats) assignats a cada cambrer. S'utilitza principalment per detectar quins cambrers encara no han estat utilitzats en la comanda i, quan és possible, prioritzar-ne l'assignació per reduir duplicacions i repartir les tasques de manera més homogènia. També s'utilitza al finalitzar per saber quin producte s'ha assignat a cada cambrer.

Relació amb les mètriques experimentals La representació de la càrrega descrita en aquest apartat es vincula directament amb les mètriques utilitzades en la fase experimental. En primer lloc, la distribució de `load` entre cambrers permet quantificar el **balanç de càrrega**, analitzant fins a quin punt una estratègia reparteix el treball de manera homogènia (per exemple, mitjançant mesures de dispersió o diferència entre cambrers).

En segon lloc, la definició de cost i la seva acumulació en forma de càrrega s'utilitzen per estimar l'impacte de l'assignació sobre el **temps de preparació de les comandes**, ja que una concentració excessiva de càrrega en un subconjunt de cambrers pot incrementar el temps fins que una comanda queda completada.

Finalment, la comparació entre càrrega assignada i disponibilitat dels cambrers permet mesurar el **temps d'inactivitat** (*idle time*), és a dir, el temps en què un cambrer roman sense tasques assignades mentre encara hi ha comandes pendents. Aquestes tres dimensions (equilibri, temps i inactivitat) serveixen com a base per comparar de manera consistent el comportament de les diferents estratègies d'assignació.

5.4 Estratègia de repartiment de comandes

Entrada i sortida del motor d'assignació

Entrada:

- **Ordre a assignar** (O): representació de la comanda com un mapping producte–quantitat,
$$O : P \to \mathbb{N}, \quad O(p) = q$$
on P és el conjunt de productes i q la quantitat demanada del producte p.

- **Cambrers disponibles** (C): conjunt de cambrers candidats a rebre assignacions.

- **Càrrega prèvia dels cambrers** (L):
$$L : C \to \mathbb{R}_{\geq 0}$$

on $L(c)$ és la càrrega acumulada del cambrer c abans d'assignar la comanda.

- **Costos dels productes** ($c(p)$): Diccionari per poder accedir al cost de cada producte. Per exemple, $c(p) = 4$.
- **Context recent** (A): conjunt de cambrers assignats en l'última comanda, utilitzat per aplicar heurístiques d'evitació o rotació (p. ex. evitar repetir els mateixos cambrers en comandes consecutives),

$$A \subseteq C$$

Sortida:

- **Distribució d'assignació** (D): mapping de producte a (cambrer a quantitat),

$$D : \ P \to (C \to \mathbb{N})$$

on $D(p)(c)$ indica la quantitat del producte p assignada al cambrer c.

- **Càrrega actualitzada** ($\widehat{L}$): el motor retorna directament la càrrega resultant després d'aplicar l'assignació de la comanda,

$$\widehat{L} : \ C \to \mathbb{R}_{\geq 0}$$

on $\widehat{L}(c)$ és la càrrega final del cambrer c un cop assignada la comanda (és a dir, la càrrega que s'escriu la base de dades).

Criteris principals

- **Balanceig global (càrrega)**: el motor té en compte l'assignació al cambrer amb menor *càrrega* en el moment de decidir, actualitzant-la a mesura que s'assignen productes. Això busca reduir desequilibris acumulats entre cambrers, i depèn del criteri concret implementat a la funció `select_least_loaded_waiter` (o variants), que determina com es compara i es prioritza la càrrega entre candidats.
- **Coherència dins comanda**: es tendeix a limitar duplicacions innecessàries dins d'una mateixa comanda, intentant (quan és possible) que un cambrer pugui assumir un producte sencer o un bloc coherent de productes, i evitant repartir petites quantitats entre molts cambrers si això no aporta un benefici operatiu clar. En particular, la partició d'un producte entre cambrers només s'aplica quan és necessària per complir límits de quantitat o regles específiques del tipus de producte.
- **Restriccions per grup o tipus de producte**: el motor aplica regles simples per evitar saturacions i decisions poc realistes. En concret, defineix un límit de quantitat que es pot assignar a un mateix cambrer abans de forçar un repartiment:

- **Límit general** (`MAX_PRODUCT_QUANTITY`): per defecte, no s'acumula més de 3 unitats del mateix producte en un sol cambrer dins la mateixa comanda, per evitar que un cambrer quedi "enganxat" a un únic producte mentre altres queden infrautilitzats.
- **Productes de dificultat alta** (`MAKE_ALONE_PRODUCTS` i `MAX_DIFFICULT_QUANTITY`): per un conjunt de còctels considerats més costosos o lents de preparar (p. ex. *caipirinha*, *espresso martini*, *long island*), el límit baixa a 2 unitats, afavorint repartir-los entre cambrers quan la quantitat creix.
- **Productes de nevera** (`MAX_FRIDGE_QUANTITY`): per productes de servei més ràpid associats a nevera, es permet un límit superior (6 unitats) ja que acostumen a tenir menor cost i menor risc de crear un coll d'ampolla.

A més, quan un producte pertany al conjunt `MAX_DIFFICULT_PRODUCTS` (subconjunt de productes especialment exigents), el motor tendeix a canviar de cambrer més aviat per evitar acumular-ne múltiples unitats seguides en una sola persona.

- **Context recent (depenent de l'algoritme)**: quan es disposa d'informació de l'ordre anterior, es penalitza (o s'evita) seleccionar cambrers que hagin estat assignats recentment, afavorint la rotació i reduint la probabilitat de repetir el mateix cambrer en comandes consecutives.

Nivell de decisió El motor pren decisions a nivell de **producte agregat** dins d'una comanda. En una primera fase, l'ordre d'entrada O es representa com un mapping producte $\rightarrow$ quantitat (sumant totes les línies d'un mateix producte). A partir d'aquesta representació, el motor calcula una distribució D que pot:

- assignar **la totalitat** d'un producte a un sol cambrer, o bé
- **partir (split)** la quantitat quan es superen certs llindars (p. ex. límits per grup o producte) i repartir-la entre dos (o més) cambrers.

Això implica que, tot i que l'entrada es tracta de forma agregada, l'execució final permet particions de quantitat per mantenir restriccions operatives i equilibrar la càrrega.

Traçabilitat i consistència La distribució calculada pel motor es reflecteix de forma traçable a la base de dades mitjançant l'atribut `waiter_id` de les entitats `OrderProduct`. Quan una mateixa línia original (`OrderProduct`) ha de ser assignada parcialment a cambrers diferents, s'aplica una operació de **partició**:

- la línia original es redueix a la quantitat restant, i
- es crea una nova línia amb la quantitat assignada i el seu `waiter_id`.

D'aquesta manera, cada unitat de producte queda vinculada a un cambrer concret, preservant la coherència de quantitats. Addicionalment, el procés d'assignació no altera l'estat lògic de la comanda sinó que n'afegeix la responsabilitat d'execució per línia, mantenint consistents els estats existents del flux de comanda. Finalment, cada línia `OrderProduct` queda assignada a un únic `waiter_id`, i la suma de quantitats de totes les línies (originals i generades per *split*) preserva la quantitat total demanada per a cada producte.

5.5 Algoritmes implementats

Aquesta secció descriu les polítiques de selecció de cambrer implementades al motor d'assignació. Totes comparteixen la mateixa interfície de decisió (selecció d'un cambrer candidat) i operen sobre l'estat de la comanda i del torn, encapsulat a `AssignmentInput` (`waiter_loads`, `waiter_order_loads`, `waiter_products`, i opcionalment `last_order_waiters` i `exclude`). El resultat de cada política és el nom del cambrer escollit per rebre la següent assignació.

5.5.1 Round-robin

Implementació. L'algorisme `round-robin` implementa una selecció cíclica global: manté un punter intern (`_rr_idx`) que recorre sempre el conjunt de cambrers en un ordre fix (ordenats pel seu nom). A cada decisió, retorna el següent cambrer segons el punter i l'avança. Si el cambrer està dins la llista d'exclusió (`exclude`), s'avança fins trobar un candidat elegible.

Comportament. És una política que no té en compte la càrrega (L) ni el context recent. Assegura un repartiment uniforme en nombre de seleccions, però pot generar desequilibris si els costos dels productes són heterogenis, ja que tracta totes les assignacions com a equivalents.

5.5.2 Least-load

Implementació. L'algorisme `least-loaded` selecciona el cambrer amb menor càrrega acumulada actual (`waiter_loads`), ignorant la composició de la comanda i el context recent. En cas d'exclusió, només es consideren els cambrers elegibles.

Comportament. És una política orientada al balanceig global: tendeix a compensar diferències de càrrega al llarg del torn. Tot i així, no té en compte l'optimització del temps de preparació de les comandes

5.5.3 Power of two choices

Implementació. L'algorisme `power-of-two-choices` aplica l'estratègia clàssica de triar dos candidats aleatoris (elegibles) i seleccionar el que té menor càrrega acumulada. Per tal de fer experiments reproduïbles, l'atzar és determinista mitjançant una llavor fixa (`seed`) al generador `_rng`.

Comportament. Aquesta política acostuma a aproximar el comportament de *least-load* amb menys cost de decisió (no cal comparar contra tots els cambrers). En entorns amb molts cambrers, ofereix un bon compromís entre simplicitat i balanceig, tot mantenint una certa diversitat d'assignacions.

5.5.4 Algoritme proposat

En aquest treball s'han desenvolupat dues polítiques personalitzades, `custom-1` i `custom-2`, que operen sobre el mateix estat i introdueixen objectius més específics que el simple *least-load*.

Custom-1 (puntuació ponderada) Idea. Combina tres factors en una única puntuació: (i) càrrega global acumulada (equitat al llarg del torn), (ii) càrrega assignada dins la comanda actual (evitar concentrar una comanda en una sola persona), i (iii) una penalització per reutilitzar un cambrer ja utilitzat en aquella comanda.

Implementació. Per evitar dominància d'escales, es normalitzen la càrrega global i la càrrega de comanda (dividint per màxims). La funció objectiu té la forma:

$$\text{score}(c) = 0.7 \cdot \widetilde{L}(c) + 0.3 \cdot \widetilde{L}_O(c) + \pi(c)$$

on $\widetilde{L}$ és la càrrega global normalitzada, $\widetilde{L}_O$ la càrrega de la comanda normalitzada, i $\pi(c)$ una penalització fixa si el cambrer ja ha estat utilitzat dins la comanda. Es tria el cambrer amb puntuació mínima.

Comportament. Manté el focus en equitat , però introdueix control sobre la concentració dins la comanda, penalitzant que un mateix cambrer acumuli molts productes de la mateixa ordre quan hi ha alternatives similars en càrrega.

Custom-2 (Targeted Spread) Idea. Maximitzar el nombre de cambrers diferents que entren en joc a l'inici de cada comanda, sense perdre el criteri d'equitat global.

Implementació. L'algorisme segueix una jerarquia de preferències:

1. triar un cambrer **no utilitzat en la comanda actual**,
2. si és possible, que també **no hagi aparegut en l'última comanda**,
3. usar una puntuació combinada (global + comanda) com a criteri de desempat o *fallback*.

La puntuació de desempat es defineix com:

$$\text{blend}(c) = 0.6 \cdot \widetilde{L}(c) + 0.4 \cdot \widetilde{L}_O(c)$$

i, si tots els cambrers ja han estat utilitzats en la comanda, es prioritza minimitzar primer la càrrega de comanda i després la global.

Comportament. Aquesta política força un repartiment inicial més "ampli" dins la comanda, que pot millorar la percepció de rapidesa i reduir colls d'ampolla puntuals. El *fallback* final evita que el mecanisme de dispersió generi desequilibris acumulats massa grans.

5.6 Decisions de disseny

- **Interfície unificada per algoritmes.** Totes les polítiques es resolen a través d'una mateixa funció de selecció (`assign_order`) basada en un paràmetre `algorithm`. Això permet comparar estratègies amb el mateix motor de distribució i només variar el criteri de selecció.
- **Separació entre estat global i estat de comanda.** Es diferencia explícitament la càrrega acumulada del torn (`waiter_loads`) de la càrrega ja assignada dins la comanda actual (`waiter_order_loads`), fet que permet controlar tant l'equitat a llarg termini com la concentració dins una ordre.
- **Reproductibilitat experimental.** Les estratègies amb component aleatori (p. ex. *power of two choices*) utilitzen un generador pseudoaleatori determinista amb llavor fixa, facilitant comparacions repetibles entre execucions.
- **Mecanismes d'exclusió.** El paràmetre `exclude` permet evitar assignacions a un subconjunt de cambrers (p. ex. per restriccions temporals o per forçar alternança), simplificant la integració de regles operatives sense modificar els algoritmes.
- **Heurístiques de context recent.** Les variants orientades a velocitat/rotació fan ús de `last_order_waiters` per evitar repeticions consecutives i incrementar la diversitat d'assignació, especialment en escenaris amb flux intens de comandes.

6 Implementació

Aquest capítol descriu com s'ha implementat el sistema de suport a l'experimentació: la base de dades, el motor d'assignació (polítiques), el simulador i el registre/-visualització de mètriques. L'objectiu de la implementació no és un producte final de producció, sinó una base modular que permeti executar algoritmes en condicions controlades i comparar-ne el comportament de manera repetible.

6.1 Tecnologies utilitzades

La implementació s'ha construït amb una selecció d'eines orientades a la simplicitat i la reproductibilitat experimental:

- **Python**: llenguatge principal per al motor d'assignació, el simulador i la generació de mètriques. Es prioritza la rapidesa de prototipatge i la facilitat per iterar sobre heurístiques.
- **SQLite**: base de dades lleugera per persistir entitats (cambrers, productes, comandes i línies de comanda) i facilitar execucions repetibles reconstruint l'estat inicial.
- **Pygame**: biblioteca per implementar la simulació amb interfície visual i bucle temporal controlat (FPS fixos), permetent observar el flux d'assignació i execució mentre es registren mètriques.
- **Entorn de desenvolupament**: control de versions amb Git i edició amb un IDE (p. ex. VSCode), orientat a iteració ràpida sobre el codi i el disseny d'experiments.

Aquesta combinació és suficient per modelar l'assignació, executar escenaris i obtenir resultats quantitatius sense dependències de desplegament ni infraestructura externa.

6.2 Estructura del projecte

El projecte s'organitza en mòduls que separen responsabilitats: persistència, lògica d'assignació, simulació i mètriques. Aquesta separació és clau per poder substituir polítiques d'assignació sense afectar el simulador, i per iterar sobre l'entorn de simulació sense modificar el nucli algorísmic.

L'estructura del repositori és la següent: Veure Figura 5 (pàg. 31).

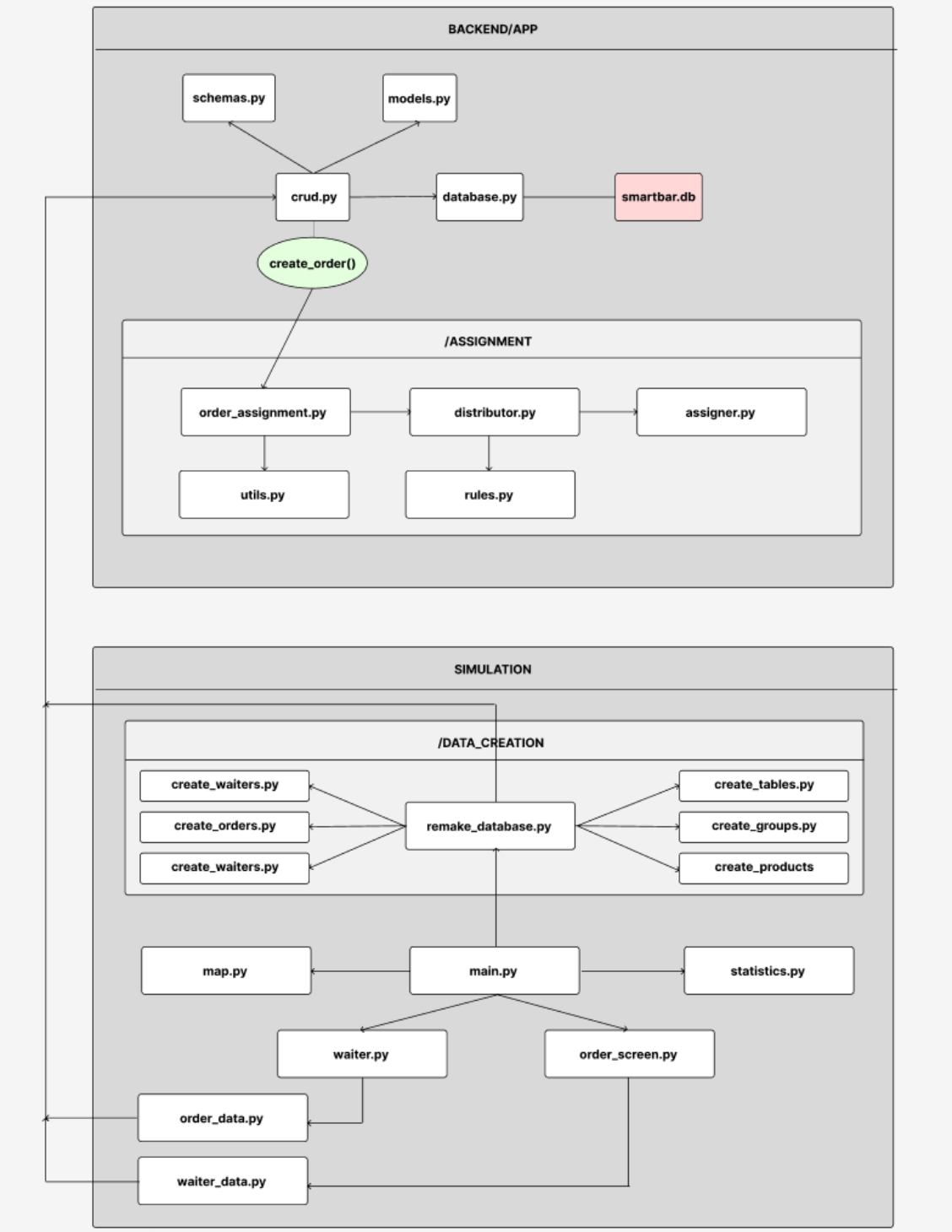


Figura 5: Esquema d'arxius del projecte

Explicació de diferents components importants pel projecte:

- `backend/app/`: inicialització de la base de dades SQLite, definició d'esquemes/models i operacions *CRUD*.
- `backend/app/assignment/`: capa d'assignació dins el backend (cas d'ús principal). Punt d'entrada: `order_assignment_service.py`, es crida des de `create_order()` del mòdul *CRUD*. Quan s'acaba de distribuir la comanda, s'encarrega de guardar les assignacions i les noves càrregues a la base de dades.
- `backend/app/assignment/distributor.py`: arxiu amb la funció principal per repartir les comandes entre els cambrers. Aplica la fórmula d'increment de càrrega i divideix (split) les quantitats quan cal. Quan necessita seleccionar un nou cambrer, invoca la política escollida definida a `assigner.py`.
- `backend/app/assignment/assigner.py`: implementació de les polítiques (*round-robin*, *least-load*, *power-of-two*, *custom-1* i *custom-2*).
- `simulation/`: simulador amb Pygame (bucle temporal, gestió d'estats i execució), integració amb el motor d'assignació i definició de l'escenari (p. ex. `map.json`).
- `simulation/data_creation/`: directori amb scripts per regenerar una base de dades de simulació per testejar els algoritmes.
- `simulation/main.py`: arxiu executable per iniciar la simulació.

A nivell d'execució, el simulador i el backend comparteixen el mateix model de dades per evitar duplicació de lògica i garantir coherència entre l'estat que es persisteix i les mètriques que es mesuren. En particular, l'assignació es calcula mitjançant el servei del backend i el simulador consumeix aquestes assignacions i actualitza l'evolució temporal (ocupació, finalització de productes i estat de les comandes) mentre registra els resultats.

6.3 Motor de repartiment de càrrega

El motor d'assignació implementa les polítiques descrites al Capítol 5 i s'integra al backend com un servei invocable tant des del flux de creació de comandes com des del simulador. La peça d'entrada és el servei `assign_waiters_to_order`, que (i) carrega l'estat de domini des de la base de dades, (ii) construeix una representació agregada de la comanda i (iii) delega la decisió al component `OrderDistributor`, que aplica regles operatives i invoca la política de selecció (`OrderAssigner`).

Entrada del motor El motor treballa sobre una representació agregada de la comanda (producte → quantitat) i sobre l'estat necessari per prendre decisions coherents amb els objectius de balanç i temps:

- **Comanda agregada** (`order_dict`): diccionari `{product_name: quantity}`.
- **Càrrega global acumulada** (`waiter_loads`): valors $L(c)$ per a tots els cambrers, inicialitzats a partir del camp persistent `load` de cada cambrer.
- **Càrrega dins la comanda** (`waiter_order_loads`): increment de càrrega ja assignat dins la comanda actual, útil com a criteri de desempat i per evitar concentració. Inicialment és igual a 0.
- **Productes assignats dins la comanda** (`waiter_products`): estructura que permet detectar cambrers encara no utilitzats a la comanda i evitar repeticions quan convé. Inicialment és un diccionari buit.
- **Context recent** (`last_order_waiters`): llista de cambrers assignats a la comanda anterior, utilitzada per variants orientades a rotació/velocitat.

6.4 Pseudocodi del procés d'assignació

Aquest apartat resumeix el flux general del motor d'assignació: agregació de la comanda, selecció de cambrers segons una política, aplicació de regles operatives (*split*) i actualització de càrrega.

order_assignment.py:

Algorithm 1: Assignar cambrers a una comanda

Entrada: *order_products*, *algoritme*
Sortida: *distribucio*

```
cambrers, productes, last ← CRUD.CARREGAR_DADES()
product_costs ← BUILD_COSTS(productes) ▷ grup → (producte → cost)

order_dict ← AGREGAR_COMANDA(order_products, productes) ▷ producte →
  unitats

distribucio ←
  DISTRIBUIR_COMANDA(algoritme, order_dict, product_costs, cambrers, last)
CRUD.GUARDAR_DISTRIBUCIO(distribucio)
return distribucio
```

Algorithm 2: Distribuir una comanda (`distribute_single_order`)

Entrada: *algoritme*, *order* (producte → quantitat), *last* (llista de cambrers)
Sortida: D (producte → (cambrer → quantitat)), *estimated_load*

$waiter\_products \leftarrow \{w \mapsto \emptyset \mid w \in waiters\}$
$waiter\_order\_load \leftarrow \{w \mapsto 0 \mid w \in waiters\}$
$estimated\_load \leftarrow$ `COPIA`$(waiter\_load)$
foreach $group \in groups$ **do**
    $current \leftarrow$ `ASSIGN_ORDER`$(algoritme, waiter\_products, estimated\_load, waiter\_order\_load, last)$
    $units\_assigned \leftarrow \{w \mapsto 0 \mid w \in waiters\}$ ▷ `reset per grup`
    foreach $(product, quantity) \in order$ **do**
        if $product \notin product\_costs[group]$ **then**
        $limit \leftarrow$ `GROUP_LIMIT`$(group, product)$
        if $units\_assigned[current] + quantity \geq limit$ **then**
            $split \leftarrow \lfloor quantity/2 \rfloor$
            `assign`(current, product, split)
            $units\_assigned[current] \leftarrow units\_assigned[current] + split$
            $remaining \leftarrow quantity - split$
            if $remaining > 0$ **then**
                $next \leftarrow$ `ASSIGN_ORDER`$(algoritme, waiter\_products, estimated\_load, waiter\_order\_load, last, \{current\})$
                `assign`(next, product, remaining)
                $units\_assigned[next] \leftarrow units\_assigned[next] + remaining$
        else
            `assign`(current, product, quantity)
            $units\_assigned[current] \leftarrow units\_assigned[current] + quantity$
        $q\_cur \leftarrow waiter\_products[current].get(product, 0)$
        if *SHOULD_SWITCH_WAITER*$(group, product, q\_cur)$ **then**
            $current \leftarrow$ `ASSIGN_ORDER`$(algoritme, waiter\_products, estimated\_load, waiter\_order\_load, last)$
$D \leftarrow$ `BUILD_DISTRIBUTION`$(order, waiter\_products)$
return $(D,\ estimated\_load)$

Algorithm 3: Funció `assign(waiter, product, qty)`

Entrada: *waiter* (string), *product* (string), *qty* (int), *product_costs*, *carrega*, *carrega_comanda*, *waiter_products*
Sortida: Actualitza càrregues i *waiter_products*

if $qty \leq 0$ **then**
  return
$cost \leftarrow$ `COST`$(product, product\_costs)$
$load \leftarrow cost + \frac{cost}{2} \cdot (qty - 1)$ ▷ $\Delta L_i(p, q)$

$carrega[waiter] \leftarrow carrega[waiter] + load$ ▷ $L_i \leftarrow L_i + \Delta L_i$

$carrega\_comanda[waiter] \leftarrow carrega\_comanda[waiter] + load$
$waiter\_products[waiter][product] \leftarrow$
  $waiter\_products[waiter].get(product, 0) + qty$

6.5 Pseudocodi dels diferents algorismes

6.5.1 round-robin

Algorithm 4: `ROUND_ROBIN(waiter_loads, exclude)`

Entrada: *waiter_loads* (cambrer → càrrega), *exclude* (llista)
Sortida: w

$E \leftarrow$ `SET`$(exclude)$
$W \leftarrow$ `SORT`(`CLAUS`$(waiter\_loads)$)
$n \leftarrow |W|$
if $n = 0$ **then**
  error ▷ `no hi ha cambrers`

for $k \leftarrow 1$ **to** n **do**
  $w \leftarrow W[rr\_idx \bmod n]$
  $rr\_idx \leftarrow (rr\_idx + 1) \bmod n$
  if $w \notin E$ **then**
    return w
error ▷ `tots exclosos`

6.5.2 least-loaded

Algorithm 5: `LEAST_LOADED(waiter_loads, exclude)`

Entrada: *waiter_loads* (cambrer → càrrega), *exclude* (llista)
Sortida: w

$E \leftarrow$ `SET`$(exclude)$
$C \leftarrow \{\, w \in$ `CLAUS`$(waiter\_loads) \mid w \notin E \,\}$
if $|C| = 0$ **then**
    error ▷ `tots exclosos`
return $\arg\min_{w \in C}\ waiter\_loads[w]$

6.5.3 power-of-two-choices

Algorithm 6: `POWER_OF_TWO(waiter_loads, exclude)`

Entrada: *waiter_loads* (cambrer → càrrega), *exclude* (llista)
Sortida: w

$E \leftarrow$ `SET`$(exclude)$
$C \leftarrow [\, w \in$ `CLAUS`$(waiter\_loads) \mid w \notin E \,]$
if $|C| = 0$ **then**
    error
if $|C| = 1$ **then**
    return $C[0]$
$(a, b) \leftarrow$ `RANDOM_SAMPLE_2`(C) ▷ `RNG determinista amb llavor`

if $waiter\_loads[a] \leq waiter\_loads[b]$ **then**
    return a
else
    return b

6.5.4 custom-1

Algorithm 7: `CUSTOM_1(waiter_products, waiter_loads, waiter_order_loads, exclude)`

Entrada: $waiter\_products$, $waiter\_loads$, $waiter\_order\_loads$, $exclude$
Sortida: w

$E \leftarrow \texttt{SET}(exclude)$
$C \leftarrow \{\, w \in \texttt{CLAUS}(waiter\_loads) \mid w \notin E \,\}$
if $|C| = 0$ **then**
    error
▷ `Normalització per prevenir que un valor domini l'escala`
$maxG \leftarrow \max_{w \in C}\ waiter\_loads[w]$
$maxO \leftarrow \max_{w \in C}\ waiter\_order\_loads[w]$
if $maxG = 0$ **then**
    $maxG \leftarrow 1$
if $maxO = 0$ **then**
    $maxO \leftarrow 1$
foreach $w \in C$ **do**
    $g \leftarrow waiter\_loads[w]/maxG$
    $o \leftarrow waiter\_order\_loads[w]/maxO$
    $\pi \leftarrow 0$
    if $waiter\_products[w] \neq \emptyset$ **then**
        $\pi \leftarrow 0.25$
    $score[w] \leftarrow 0.7 \cdot g + 0.3 \cdot o + \pi$
return $\arg\min_{w \in C}\ score[w]$

6.5.5 custom-2

Algorithm 8: `CUSTOM_2(waiter_products, waiter_loads, waiter_order_loads, last, exclude)`

Entrada: $waiter\_products$, $waiter\_loads$, $waiter\_order\_loads$, $last$, $exclude$
Sortida: w

$E \leftarrow \texttt{SET}(exclude)$
$L \leftarrow \texttt{SET}(last)$
$Eligible \leftarrow [\, w \in \texttt{CLAUS}(waiter\_loads) \mid w \notin E \,]$
if $|Eligible| = 0$ **then**
    error
$Unused \leftarrow [\, w \in Eligible \mid waiter\_products[w] = \emptyset \,]$
$UnusedNotLast \leftarrow [\, w \in Unused \mid w \notin L \,]$
$maxG \leftarrow \max_{w \in Eligible}\ waiter\_loads[w]$
$maxO \leftarrow \max_{w \in Eligible}\ waiter\_order\_loads[w]$
if $maxG = 0$ **then**
    $maxG \leftarrow 1$
if $maxO = 0$ **then**
    $maxO \leftarrow 1$
`BLEND`(w) $g \leftarrow waiter\_loads[w]/maxG$
$o \leftarrow waiter\_order\_loads[w]/maxO$
return $0.6 \cdot g + 0.4 \cdot o$
▷ `1) Spread: unused i evitar last si es pot`
if $|UnusedNotLast| > 0$ **then**
    return $\arg\min_{w \in UnusedNotLast}$ `BLEND`(w)
▷ `2) Si no es pot evitar last, igualment spread dins la comanda`
if $|Unused| > 0$ **then**
    return $\arg\min_{w \in Unused}$ `BLEND`(w)
▷ `3) Tots ja s'han usat: minimitzar concentració dins la comanda`
return $\arg\min_{w \in Eligible} (waiter\_order\_loads[w],\ waiter\_loads[w])$

6.6 Simulador de comandes i cambrers

El simulador implementa un entorn controlat que executa comandes i cambrers en el temps, permetent:

- Injectar un conjunt de comandes aleatòries
- Aplicar una política d'assignació concreta,
- Quantificar mètriques (temps de preparació, càrrega final i inactivitat).

Model temporal i bucle de simulació La simulació s'executa amb un bucle a FPS fixos. El temps del model avança en passos discrets, i el factor `speed` permet accelerar l'experiment mantenint proporcions temporals.

Estats dels cambrers Cada cambrer es modela com un agent amb estat, com a mínim:

- **Estat ocupat**: està preparant un producte (o un conjunt assignat) i no pot acceptar noves tasques fins que finalitza el temps associat,
- **Estat esperant** (o *idling*): no té tasca assignada i roman disponible.

El temps d'inactivitat (*idle time*) s'acumula exactament quan el cambrer es troba en estat d'espera mentre el sistema encara té comandes pendents.

Execució de comandes Una comanda es considera completada quan totes les seves línies han finalitzat. El simulador registra:

- Temps d'inici de preparació.
- Temps de completament.
- Calcula T_{prep} per comanda.

Mapa i visualització El mapa (`map.json`) defineix posicions inicials i una disposició bàsica de l'entorn per fer la simulació interpretable visualment. Tot i que l'objectiu principal és experimental, la visualització facilita l'avaluació qualitativa del funcionament dels algorismes i determinats patrons que puguin donar-se (saturacions, cambrers infrautilitzats i efectes de fragmentació de comandes).

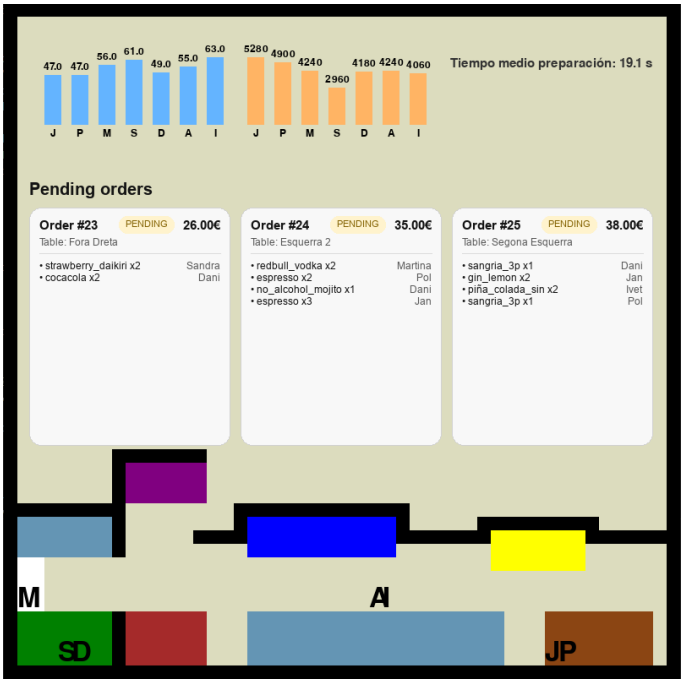


Figura 6: Visualització simulació

7 Anàlisi estadística comparativa dels algoritmes

En aquest apartat es presenta una anàlisi estadística comparativa de les polítiques d'assignació implementades a partir de les execucions registrades a la fulla de càlcul de resultats. Per a cada política s'han executat $n = 4$ simulacions sota les mateixes condicions estructurals (mateix nombre de cambrers i mateixa configuració del simulador), registrant les mètriques d'avaluació següents:

- El **temps mitjà per comanda** ($\overline{T}_{prep}$).
- La **dispersió de càrrega entre cambrers** com a mesura d'equitat.

7.1 Disseny experimental i mètriques

Conjunts de comandes i determinisme Cada execució consta de **100 comandes** generades de manera determinista amb una llavor (seed) fixa. S'han utilitzat quatre llavors diferents (`43, 44, 45, 46`), de manera que:

- dins d'una mateixa execució, **tots els algoritmes processen exactament el mateix conjunt de 100 comandes** (comparació justa),
- entre execucions, el conjunt de comandes canvia (robustesa davant diferents mostres d'entrada).

A més, l'estat de càrrega dels cambrers es reinicia a l'inici de cada execució per garantir comparabilitat.

Visualització La pantalla de la simulació mostra:

- la distribució de càrrega final per cambrer $L(c)$ (gràfic de barres),
- el temps mitjà de preparació de les comandes $\overline{T}_{prep}$.

Estimadors Per al temps per comanda es reporta la mitjana mostral $\overline{T}_{prep}$ i la desviació estàndard mostral s_T sobre $n = 4$ repeticions, amb un interval de confiança del 95% sota t-Student:

$$IC_{95\%}\big(\overline{T}_{prep}\big) = \overline{T}_{prep} \pm t_{0.975,\,n-1} \cdot \frac{s_T}{\sqrt{n}}.$$

Com que n és petit, l'interval s'ha d'interpretar com a orientatiu (els intervals se solapen en els millors algoritmes).

Per a l'equitat, a partir del vector de càrregues finals per cambrer $L(c)$ (agregat com a mitjana entre execucions), es reporta:

$$\sigma_L = \mathrm{sd}\big(L(c)\big), \qquad \Delta L = \max_c L(c) - \min_c L(c).$$

Valors baixos de σ_L i ΔL indiquen millor repartiment de la càrrega.

7.2 Resum estadístic global

Algorisme	n	$\overline{T}_{prep}$ (s)	s_T	$IC_{95\%}$ (s)	σ_L	ΔL
`custom-1`	4	21,05	0,99	[19,48; 22,62]	4,70	12,75
`least-loaded`	4	21,48	0,71	[20,34; 22,61]	4,15	10,50
`custom-2`	4	21,50	0,84	[20,16; 22,84]	4,73	14,25
`power-of-2`	4	26,15	2,41	[22,32; 29,98]	45,14	118,50
`round-robin`	4	28,38	1,38	[26,17; 30,58]	93,86	233,00

Taula 1: Resum estadístic global per algorisme (100 comandes per execució, 4 execucions amb seeds diferents). σ_L i ΔL mesuren la dispersió de càrrega final entre cambrers (equitat).

Lectura global dels resultats Els resultats mostren un patró clar:

- **Eficiència temporal.** Els tres millors algoritmes en temps mitjà són `custom-1`, `least-loaded` i `custom-2`, amb valors molt propers (al voltant de 21–22 s). En canvi, `power-of-2` i `round-robin` empitjoren el temps de manera notable.
- **Equitat de càrrega.** Les polítiques `least-loaded`, `custom-1` i `custom-2` mantenen una dispersió molt baixa ($\sigma_L \approx$ 4–5), mentre que `power-of-2` i, sobretot, `round-robin` generen desequilibris marcats.
- **Interpretació dels intervals.** Tot i que `custom-1` obté la millor mitjana, els intervals de confiança entre els tres millors es solapen, de manera que la diferència és petita i el resultat s'ha d'entendre com una evidència empírica a favor d'aquestes estratègies, no com una separació absoluta.

7.2.1 Distribució de càrrega per cambrer (exemples representatius)

Per visualitzar el tipus de desequilibri que generen alguns algoritmes, la Taula 2 mostra la càrrega *mitjana per execució* assignada a cada cambrer en quatre polítiques representatives. Es veu que `round-robin` tendeix a concentrar molta càrrega en alguns cambrers (p. ex. IVET i DANI) i deixar-ne d'altres sistemàticament per sota (p. ex. SANDRA), mentre que `least-loaded` i `custom-1` mantenen els valors molt propers.

7.2.2 Anàlisi dels algorismes

`round-robin` És el *baseline* més simple, però en aquest escenari és clarament el pitjor en eficiència temporal i equitat. Obté $\overline{T}_{prep} = 28,38$ s i un desequilibri molt alt ($\sigma_L = 93,86$, $\Delta L = 233,00$). La causa és estructural: l'assignació cíclica ignora el cost dels productes i pot acumular càrrega "cara" en cambrers concrets, generant colls d'ampolla quan arriben comandes grans o productes lents.

Algorisme	JAN	POL	MARTINA	SANDRA	DANI	ALADID	IVET
`round-robin`	325,25	183,25	222,25	166,75	383,00	260,25	399,75
`power-of-2`	312,50	301,00	272,50	200,00	232,25	318,50	303,75
`least-loaded`	281,25	272,00	279,75	279,50	277,50	279,75	270,75
`custom-1`	275,75	276,00	273,75	282,50	282,75	279,75	270,00
`custom-2`	281,00	278,75	282,00	267,75	278,75	276,50	275,75

Taula 2: Càrrega mitjana per execució i cambrer $L(c)$ (mitjana de 4 execucions).

`power-of-2` Millora respecte a `round-robin` en equitat i temps, però continua sent clarament inferior als mètodes basats en càrrega. En particular, presenta una variabilitat elevada ($s_T = 2,41$) i un desequilibri notable ($\sigma_L = 45,14$). En aquest entorn, triar només dos candidats no és suficient per evitar que certes seqüències d'assignacions acabin penalitzant repetidament algun cambrer.

`least-loaded` Mostra un comportament molt estable: bon repartiment ($\sigma_L = 4,15$ i $\Delta L = 10,50$) i un temps mitjà competitiu ($\overline{T}_{prep} = 21,48$ s). En altres paraules, és una política robusta que evita saturacions persistents i manté un equilibri molt alt entre cambrers.

`custom-1` És el millor resultat en temps mitjà ($\overline{T}_{prep} = 21,05$ s) mantenint una equitat molt alta ($\sigma_L = 4,70$). El comportament és coherent amb la seva definició: combina càrrega global i càrrega dins la comanda amb una penalització per reutilització, aconseguint una assignació que (i) evita concentrar massa línies d'una comanda en un sol cambrer i (ii) no degrada el repartiment acumulat.

`custom-2` Obté un temps mitjà molt proper als millors ($\overline{T}_{prep} = 21,50$ s) i una equitat també elevada ($\sigma_L = 4,73$). En comparació amb `custom-1`, mostra una mica més de dispersió en aquest conjunt de dades, però continua dins el mateix rang d'alta equitat. És una política especialment pensada per dispersar participants dins la comanda, però amb aquests paràmetres i conjunt de comandes queda pràcticament empatada amb `least-loaded`.

7.2.3 Conclusió de l'estudi comparatiu

Amb els resultats obtinguts (100 comandes per execució i 4 execucions amb seeds diferents), es confirma que les polítiques basades en **càrrega** (`least-loaded`, `custom-1` i `custom-2`) són clarament superiors als baselines `power-of-2` i `round-robin` tant en **temps** com en **equitat**. En particular, els tres algoritmes capdavanters presenten valors de $\overline{T}_{prep}$ molt propers (entorn de 21–22 s) i una dispersió de càrrega baixa ($\sigma_L \approx$ 4–5), mentre que `power-of-2` i `round-robin` mostren desequilibris elevats i temps sensiblement pitjors.

El resultat més rellevant d'aquest experiment és que es pot concloure que **equilibrar la càrrega entre cambrers és el factor que millor explica la millora**

del temps. Quan la política assigna de manera consistent al cambrer amb menor càrrega acumulada, s'eviten saturacions i colls d'ampolla: cap cambrer queda "carregat" de manera persistent i, per tant, el temps fins que el sistema completa totes les línies d'una comanda es redueix. Això es reflecteix en el fet que `least-loaded`, `custom-1` i `custom-2` són pràcticament equivalents en rendiment temporal dins del marge experimental.

En canvi, les heurístiques addicionals orientades a **seleccionar cambrers de manera més "fina"** (per exemple, evitar repetir cambrers en comandes consecutives o afavorir certa dispersió dins d'una comanda) **no aporten una millora apreciable** en aquest escenari. La diferència entre `least-loaded` i les variants `custom-*` és petita i no altera el missatge principal: un cop el sistema manté el balanç de càrrega, la resta de detalls de selecció tenen un impacte menor en el temps mitjà global.

En conseqüència, la conclusió operativa és clara: si l'objectiu és minimitzar el temps de preparació mantenint un repartiment just, és suficient utilitzar una política basada en menor càrrega (`least-loaded`) o una de les variants personalitzades (`custom-1`/`custom-2`), ja que totes tres ofereixen un comportament robust i molt similar, mentre que els baselines no informats per la càrrega queden descartats per la seva pèrdua d'eficiència i equitat.

8 Conclusions i Treball Futur

8.1 Conclusions

Aquest Treball de Fi de Grau ha abordat un problema operatiu habitual en entorns de bar amb alta concurrència: la dificultat de coordinar la preparació de comandes quan els cambrers de barra són intercanviables i els productes tenen costos de preparació heterogenis. A partir de l'experiència real que motiva el projecte i dels resultats del qüestionari, s'ha justificat que el coll d'ampolla no és només la presa de comandes, sinó la **micro-assignació de productes** a la barra i el **repartiment equitatiu de tasques** en hores punta.

En aquest context, la contribució principal del treball és doble. En primer lloc, s'ha dissenyat i implementat un **motor d'assignació** basat en càrrega acumulada (*load*) i cost estimat de producte, amb diverses polítiques comparables entre si (baselines clàssics i variants personalitzades). En segon lloc, s'ha desenvolupat un **entorn de simulació** determinista i reproduïble que permet executar torns controlats, registrar mètriques i observar patrons de saturació i infrautilització.

Els resultats experimentals mostren un patró clar: les polítiques informades per càrrega (`least-loaded`, `custom-1` i `custom-2`) dominen els baselines simples (`round-robin` i `power-of-2`) tant en temps com en equitat. En l'experiment (100 comandes per execució, 4 execucions amb llavors diferents), els tres millors mètodes se situen al voltant de **21–22 s** de temps mitjà per comanda, mentre que `power-of-2` i `round-robin` empitjoren de manera notable, i alhora generen disper-

sions de càrrega molt més altes.

La conclusió més rellevant és conceptual: **equilibrar la càrrega entre cambrers és el factor que millor explica la millora del temps**. Quan la decisió assigna de forma consistent al cambrer amb menor càrrega acumulada, s'eviten saturacions persistents i colls d'ampolla, i això redueix el temps global necessari per completar les comandes. Per aquest motiu, un cop s'imposa aquest criteri, les heurístiques addicionals de selecció fina (dispersió dins la comanda, evitar repetir cambrers en comandes consecutives) tenen un impacte menor en el temps mitjà global dins del marge experimental.

En termes operatius, el missatge és clar: si l'objectiu és minimitzar el temps de preparació mantenint un repartiment just, és suficient utilitzar una política basada en menor càrrega (`least-loaded`) o una variant personalitzada (`custom-1`/`custom-2`), ja que totes tres mostren un comportament robust i molt similar en aquest escenari, mentre que les estratègies no informades per la càrrega queden descartades per la seva pèrdua d'eficiència i equitat.

8.2 Objectius assolits

A continuació es relacionen els objectius definits a l'inici del treball amb els resultats obtinguts:

- **Analitzar el funcionament real d'un bar i identificar factors clau.** S'ha descrit el flux operatiu (terrassa–barra), els estats de comanda i producte, i s'han formalitzat factors que afecten la càrrega (cost base, heterogeneïtat, probabilitat d'incidències com reposicions, i economies d'escala en preparació consecutiva). Això ha permès traduir el problema real a un model computable.

- **Implementar un entorn de simulació realista i reproduïble.** S'ha desenvolupat un simulador temporal amb bucle discret, model d'estats de cambrers (ocupat/esperant) i recollida de mètriques (temps de preparació, càrrega final i inactivitat), amb generació de comandes determinista per llavor per garantir comparabilitat entre polítiques.

- **Dissenyar estratègies d'assignació basades en criteris explícits.** S'han implementat baselines (`round-robin`, `least-loaded`, `power-of-2`) i dues polítiques pròpies. `custom-1` combina càrrega global, càrrega dins la comanda i penalització per reutilització; `custom-2` prioritza dispersar cambrers a l'inici de comanda sense perdre l'equitat global.

- **Comparar variants mitjançant mètriques quantitatives.** S'ha definit un disseny experimental amb repeticions (n=4) i mètriques d'eficiència temporal (T_{prep}) i equitat (dispersió σ_L i rang Δ_L), amb intervals de confiança orientatius per al temps mitjà.

- **Avaluar l'impacte en temps i equitat buscant un equilibri.** Els experiments confirmen que les polítiques basades en càrrega aconsegueixen si-

multàniament un temps mitjà baix i una dispersió de càrrega molt reduïda, mentre que els baselines simples degraden ambdues dimensions.

8.3 Limitacions

Tot i els resultats positius, el treball presenta limitacions que cal tenir en compte a l'hora d'interpretar les conclusions:

- **Validació basada en simulació.** L'avaluació és experimental i controlada; no s'ha desplegat el sistema en un bar real ni s'ha mesurat l'impacte amb dades operatives reals (temps reals, errors humans, coordinació efectiva, etc.).
- **Model simplificat de l'entorn.** El simulador captura estats i temps, però no modela amb detall tots els costos del món real (interrupcions, comunicació entre cambrers, congestió física de l'espai, prioritats d'urgència, canvis de criteri sota estrès, etc.). La visualització és útil qualitativament, però no introdueix aquesta complexitat de manera completa.
- **Costos de producte i càrrega com a estimacions.** El criteri central (cost estimat i *load* acumulada) depèn de com d'acurada és l'estimació del cost i de si aquesta reflecteix fidelment l'esforç real. En un entorn real, aquests costos poden variar entre cambrers i entre moments del servei.
- **Escenari concret i nombre de repeticions limitat.** Els resultats es basen en 4 execucions amb llavors diferents i una configuració fixa del simulador. Això és suficient per observar tendències clares entre famílies d'algoritmes, però limita la generalització fina (p. ex. diferències petites entre `least-loaded` i `custom-*`).
- **Abast del projecte.** S'ha prioritzat l'estudi algorísmic per sobre del desenvolupament complet d'una aplicació orientada a producció (UX, integració amb POS/KDS comercials, pagaments/facturació i estudi d'usabilitat exhaustiu), que queda fora del marc temporal del TFG.

8.4 Treball futur

Com a continuació natural del treball, es proposen diverses línies d'expansió:

- **Validació en entorn real.** Pilotar el sistema en un bar real (o en un entorn de pràctiques controlat) per contrastar les mètriques simulades amb mesures reals: temps d'espera, rendiment en hores punta, percepció de justícia i impacte en l'organització del torn.
- **Calibratge de costos i aprenentatge.** Estimar el cost de productes a partir de dades reals (logs) i ajustar-lo dinàmicament segons el context (volum, reposicions, productes encadenats) i fins i tot segons el rendiment individual de cada cambrer (heterogeneïtat d'habilitat).

- **Model més ric de restriccions operatives.** Incorporar restriccions de l'espai i del procés: estacions de preparació, congestió física, dependències (p. ex. recursos compartits), prioritats i urgències, o regles de batching per grups de producte quan sigui operativament beneficiós.
- **Anàlisi experimental més extensa.** Augmentar el nombre de repeticions, explorar més configuracions (nombre de cambrers, patrons d'arribada, distribució de costos) i afegir mètriques complementàries (percentils de temps, fairness temporal, estabilitat del sistema en pics).
- **Integració amb una aplicació final.** Recuperar la línia inicial del projecte i integrar el motor d'assignació en una aplicació completa de gestió de comandes, amb una interfície pensada per minimitzar interaccions i amb pantalles de barra tipus KDS, de manera que l'assignació algorísmica sigui una peça transparent dins del flux real.
- **Interacció humà-algoritme.** Estudiar com ha de presentar-se l'assignació perquè sigui acceptada per l'equip: explicabilitat de decisions, opcions de sobreescriptura, i mecanismes per gestionar excepcions sense perdre traçabilitat.

Referències

[1] Yossi Azar, Andrei Z. Broder, Anna R. Karlin, and Eli Upfal. Balanced allocations. *SIAM Journal on Computing*, 29(1):180–200, 1999.

[2] Varun Gupta and Mor Harchol-Balter. Analysis of join-the-shortest-queue routing for web server farms. *Performance Evaluation*, 2007. Also available via ACM/Elsevier. Accessed 2026-01-04.

[3] Varun Gupta and Mor Harchol-Balter. Analysis of join-the-shortest-queue routing for web server farms. *Performance Evaluation*, 64(9–12):1062–1081, 2007.

[4] Varun Gupta and Mor Harchol-Balter. Analysis of join-the-shortest-queue routing for web server farms (pdf), 2007. PDF hosted by CMU. Accessed 2026-01-04.

[5] Johnson & Wales University. Technology review: 4 ways kitchen display systems improve food service, 2022. Accessed 2026-01-04.

[6] Lightspeed. About pos users. `https://k-series-support.lightspeedhq.com/hc/en-us/articles/1260804647189-About-POS-users`, 2025. Accessed 2026-01-06.

[7] Michael Mitzenmacher. The power of two choices: A survey of techniques and results, 2001. PDF available online. Accessed 2026-01-04.

[8] Michael Mitzenmacher. The power of two choices in randomized load balancing. *IEEE Transactions on Parallel and Distributed Systems*, 2001. PDF available online. Accessed 2026-01-04.

[9] National Restaurant Association. 7 ways handheld pos systems improve efficiency, 2025. Accessed 2026-01-04.

[10] NCR Voyix. Access levels (aloha pos 14.1), 2025. Accessed 2026-01-06.

[11] Selcen Ozturkcan and Olgun Kitapci. A sustainable solution for the hospitality industry: The qr code menu. *Journal of Hospitality and Tourism Technology*, 2025. PDF available online. Accessed 2026-01-04.

[12] Z. Shahril, N. S. A. R. Den, N. A. S. S. Bahari, and N. I. M. Asnawi. Customer satisfaction in using digital qr code menu ordering in restaurant. *Journal of Tourism, Hospitality & Culinary Arts*, 16(1):820–831, 2024. PDF available online. Accessed 2026-01-04.

[13] The Wall Street Journal. The qr backlash has won. restaurants are ditching them for good., 2024. Accessed 2026-01-04.

[14] Toast. The true value of handheld pos systems in restaurants, 2024. Accessed 2026-01-04.

[15] Uber Eats Merchants. What is a kds? a restaurant's guide to kitchen display systems, 2025. Accessed 2026-01-04.

[16] WebstaurantStore. Restaurant's guide to kitchen display systems (kds), 2024. Accessed 2026-01-04.

Annexos

A Qüestionari 1

El qüestionari emprat per a l'anàlisi d'usuari estava format per les següents preguntes:

1. Has vist problemes per recordar comandes demanades a taula?
2. Creus que el procés de cobrament és lent?
3. Els cambrers de barra tenen dificultats per recordar les comandes rebudes?
4. Creus que un sistema digital podria ajudar a reduir els problemes anteriors?
5. Et semblaria útil prendre comandes des d'una tauleta o mòbil amb enviament automàtic a barra?
6. Veus útil un sistema digital per preparar comptes, dividir pagaments i cobrar més ràpid?
7. Veus útil un QR a la taula per consultar la carta?
8. Per prendre comandes, preferiries un mòbil o una tauleta?
9. Seria útil que els cambrers de barra veiessin una pantalla amb totes les comandes en procés?
10. Seria útil una pantalla tàctil perquè els cambrers indiquin quins productes estan preparant?
11. Explica breument alguna situació real on un sistema digital t'hauria facilitat la feina. (Pregunta oberta)

A.1 Reproducció de l'execució

Per reproduir el sistema s'han d'executar dos components: el backend (API) i la simulació.

Execució del backend Des del directori `backend/`:

```
cd ./backend/
uvicorn app.main:app --host 0.0.0.0 --port 8000 --reload
```

Execució de la simulació Executant el fitxer principal de la simulació:

```
C:/Users/polfe/AppData/Local/Programs/Python/Python311/python.exe \
d:/TFG/PROJECT/SmartBarAI/simulation/main.py
```

Determinisme i repetició dels experiments Els experiments es poden repetir fixant la *seed* de generació de comandes i mantenint constants la configuració de l'escenari (nombre de cambrers, costos de producte i nombre de comandes per execució).