Universitat
de les Illes Balears

TREBALL DE FI DE GRAU

DESENVOLUPAMENT DE SIMULADORS PER A LA DOCÈNCIA DE SISTEMES OPERATIUS

JAUME ALOY VICH

Grau d'Enginyeria Informàtica

Escola Politècnica Superior

Any acadèmic 2020-21

DESENVOLUPAMENT DE SIMULADORS PER A LA DOCÈNCIA DE SISTEMES OPERATIUS

JAUME ALOY VICH

Treball de Fi de Grau

Escola Politècnica Superior

Universitat de les Illes Balears

Any acadèmic 2020-21

Paraules clau del treball: simuladors docència, sistemes operatius, aplicació web

Tutora: Adelaida Delgado Domínguez

Autoritz la Universitat a incloure aquest treball en el repositori institucional per consultar-lo en accés obert i difondre'l en línia, amb finalitats exclusivament acadèmiques i d'investigació	Autor/a Sí	Autor/a No	Tutor/a Sí	Tutor/a No
	☑	☐	☑	☐

Vull agrair a n'Antònia, en Miquel i en Miquel Àngel
per donar-me suport durant aquest llarg camí.
I als meus companys per acompanyar-m'hi.

SUMARI

ÍNDEX DE FIGURES

ACRÒNIMS

JS JavaScript

HTML Hyper Text Markup Language

CSS Cascading Style Sheets

DOM Document Object Model

TS TypeScript

MVC Model Vista Controlador

MVP Model Vista Presentador

JSON JavaScript Object Notation

IGU Interfície Gràfica d'Usuari

Resum

Els simuladors didàctics tenen com objectiu explicar un determinat concepte a través de la visualització d'un exemple i la interacció de l'usuari amb el simulador, permetent configurar i manipular de manera simple i intuïtiva els paràmetres de la simulació per entendre com afecten als resultats.

En l'àmbit dels sistemes operatius, existeixen una gran varietat d'algorismes que s'expliquen a les assignatures d'aquesta temàtica. Per ajudar a l'explicació i assoliment d'aquests conceptes, docents i alumnes recorren a simuladors creats per a la docència. En el cas d'aquestes assignatures, existeixen una gran varietat de simuladors que faciliten aquest aprenentatge.

Ara bé, els simuladors existents han quedat obsolets pel fet d'haver deixat de rebre suport, no estan pensats per a l'usuari final o no estan complets. A més, molts d'ells varen ser desenvolupats abans de la popularització dels dispositius mòbils, fent que les aplicacions no estiguin pensades per ser utilitzades amb aquests dispositius.

Per resoldre aquest problema s'ha desenvolupat un portal de simuladors per a la docència de les assignatures de sistemes operatius. El nou portal de simuladors inclou els principals algorismes dels diferents temes tractats a les assignatures. També s'intenten resoldre totes les mancances que presenten els simuladors existents, ja que és una aplicació web multiplataforma amb una interfície d'usuari consistent entre els diferents simuladors.
A més, disposa d'exemples i tutorials que guien a l'usuari durant les seves primeres passes utilitzant el simulador amb l'objectiu de crear una experiència d'usuari satisfactòria.

Un dels objectius dels simuladors per a la docència és contribuir a l'aprenentatge autònom per part dels alumnes. El portal de simuladors desenvolupat s'ha pogut provar amb usuaris reals durant el curs 20/21 degut a la situació excepcional de docència semipresencial a l'assignatura de Sistemes Operatius II. D'aquesta manera s'ha pogut comprovar que els simuladors sí tenen una utilitat perquè els alumnes han pogut realitzar un seguiment dels algorismes que s'han introduït a la teoria i la tutora els ha pogut utilitzar per fer les classes online a través de videoconferència.

CAPÍTOL 1

INTRODUCCIÓ

1.1 Simuladors a la docència

L'aprenentatge dels conceptes d'una matèria es pot aconseguir utilitzant diferents metodologies: des de la transmissió oral i lectura dels conceptes fins a realització d'un exemple que il·lustri qualque concepte.

En el món de la informàtica molts de conceptes es presenten conjuntament amb un algorisme, que a primera vista pot parèixer difícil d'entendre. L'objectiu és disposar d'una eina que faciliti l'aprenentatge d'aquest concepte. Per exemple, mitjançant el seguiment passa a passa d'un algorisme i la modificació dels seus paràmetres es pot facilitar l'aprenentatge d'un concepte.

A les assignatures de sistemes operatius s'expliquen diferents algorismes relacionats amb la planificació de processos, la planificació d'entrada/sortida i la gestió de memòria. Moltes vegades aquests algorismes queden com un concepte teòric sense posar-lo en pràctica, per tant, és possible que l'alumne no assoleixi correctament els conceptes.

Els simuladors d'algorismes són una eina didàctica que permeten visualitzar el funcionament d'un algorisme de manera gràfica a través d'un exemple. Aquestes eines poden ser utilitzades tant per:

- els docents, com a eina auxiliar per presentar el conceptes de l'assignatura.
- els alumnes, per veure i acabar d'entendre els conceptes explicats a l'assignatura o resoldre els problemes proposats.

Idealment, aquests simuladors han de permetre a l'usuari manipular el màxim possible la quantitat dels paràmetres de la simulació per veure com afecten al resultat de la simulació i facilitar la comprensió de l'algorisme.

1.2 Simuladors de sistemes operatius existents

Un dels aspectes a considerar abans d'iniciar un projecte és avaluar si ja existeixen solucions que intenten resoldre el mateix problema, i si existeixen comprovar que ho fan de manera satisfactòria. El desenvolupament d'un portal de simuladors d'algorismes de sistemes operatius no és una excepció i és necessari analitzar els simuladors existents.

A l'hora de realitzar aquest estudi s'ha decidit agrupar els simuladors per la seva temàtica. Així, es poden trobar simuladors de planificació de processos, de planificació d'entrada/sortida i de gestió de memòria. De cada un dels simuladors trobats s'han analitzat diferents aspectes:

- *la compatibilitat amb el sistemes actuals*, és a dir, si l'aplicació funciona correctament sobre les màquines actuals i disposa d'instruccions d'instal·lació i ús.

- *la completesa* per indicar si conté els algorismes més destacables de la seva temàtica.

- *la interfície d'usuari i la interacció de l'usuari* que inclou com es presenten els resultats de la simulació i com l'usuari pot modificar les configuracions del simulador.

- altres característiques com l'existència d'una secció d'ajuda que expliqui el funcionament dels algorismes, l'existència d'exemples predeterminats del simulador i la possibilitat d'emmagatzemar i carregar les simulacions.

1.2.1 Planificació d'E/S

Un simulador d'E/S ha de poder realitzar simulacions amb qualsevol dels algorismes tradicionals de la planificació d'E/S. El resultat d'aquest simulador ha de ser un llistat de les peticions ordenades pel seu ordre d'atenció.

Els simuladors més complets i funcionals d'E/S trobats es poden resumir al següent llistat:

- ***Disk Scheduling Simulation*** de *Scott Bouloutian.* [2]
  És una aplicació web que permet simular el funcionament dels algorismes de planificació de disc fent ús dels algorismes tradicionals (FCFS, SSTF, SCAN, C-SCAN, LOOK, C-LOOK).

  Aquesta aplicació està desenvolupada utilitzant l'entorn de treball *NodeJS* fent ús de llibreries com *Bower* i *Grunt*. Tot i que està realitzada amb una tecnologia *relativament recent* les llibreries utilitzades estan obsoletes i no s'ha proporcionat una versió compilada del projecte.

Aspectes positius	Aspectes negatius
– Desenvolupat amb una tecnologia compatible amb qualsevol dispositiu actual però desactualitzada.	– El resultat de la simulació és una llista simple dels resultats amb l'ordre final i una animació del capçal. – L'aplicació web no és *responsive*.

- ***Disk Scheduling*** de *Brian Kirotich*. [3]
  És una aplicació per consola que permet realitzar les simulacions d'una seqüència de peticions amb qualsevol dels algorismes tradicionals. L'aplicació està pensada per ser executada en sistemes basats en Linux.

Aspectes negatius
– Les simulacions són execucions del programa independents cosa que impossibilita reproduir les simulacions sense haver de reintroduir el llistat de peticions. – El resultat de la simulació és una llista ordenada de les peticions. – És una aplicació per consola amb una experiència d'usuari pobre degut al flux del programa.

- ***Disk Scheduling*** de *Safiyat Reza*. [4]
  És una aplicació de consola que permet realitzar simulacions de planificació d'E/S fent ús dels algorismes FCFS, SSTF i SCAN. Aquesta aplicació està desenvolupada per sistemes Linux amb dependència de la llibreria *ncurses*.

Aspectes positius	Aspectes negatius
– Permet introduir les peticions que s'utilitzaran durant la simulació a l'inici del programa o carregar-les des d'un fitxer de text. – El resultat és un gràfic del moviment del capçal i el llistat ordenat de les peticions.	– És una aplicació de consola.

- ***Disk scheduling and memory management*** de *Raghav Arora*. [5]
  És una aplicació d'escriptori realitzada amb OpenGL que implementa els algorismes de gestió de memòria i de planificació a disc (FCFS, SCAN, SSTF).

Aspectes negatius
– No es proporcionen versions compilades de l'aplicació i no s'ha aconseguit compilar el projecte degut a la falta de les dependències de l'aplicació. – La interfície d'usuari és fixada perquè tot està realitzat a partir d'imatges.

- ***Disk Scheduling Simulation*** de *Menno Sijben.* [6]
  És un simulador per escriptori de la planificació de disc amb els algorismes tradicionals. L'aplicació està desenvolupada amb la tecnologia .NET, fent-la exclusiva per sistemes amb Windows com a sistema operatiu.

Aspectes positius	Aspectes negatius
– Té un interfície d'usuari gràfica però presenta poca informació. – No es permet guardar ni carregar les simulacions a memòria secundària, però sí que existeixen exemples predeterminats.	– El simulador té poques opcions de configuració, no es poden modificar paràmetres com la posició inicial del capçal o la quantitat de pistes. – La gestió d'excepcions no és completa, ja que en algunes situacions quan l'usuari no utilitza el programa de la manera esperada deixa de funcionar.

- ***OS Sim*** d'*Alex Macia.* [7]
  És una aplicació d'escriptori que conté múltiples simuladors de sistemes operatius. De la temàtica de planificació de disc permet realitzar simulacions amb els algorismes tradicionals. L'únic requisit és disposar d'una sistema que permeti executar aplicacions Java.

Aspectes positius	Aspectes negatius
– Permet controlar la simulació passa a passa i ajustar la velocitat de la simulació automàtica. – Es poden guardar i carregar les simulacions des de fitxers. – Aplicació d'escriptori multiplataforma amb interfície d'usuari basada en finestres.	– No és possible tornar enrere a les passes de les simulacions, només es pot avançar i reiniciar la simulació. – Els resultats de la simulació és un gràfic en què no es pot fer el seguiment de com s'atenen les peticions.

1.2.2 Planificació de processador

Un simulador de planificació de processos ha mostrar en quin ordre s'executen els processos i com canvia la seva execució en funció de l'algorisme seleccionat. El resultat del simulador és el propi ordre d'execució dels processos.

A continuació es comenten els principals simuladors de planificació de processos i es realitza una valoració seguint els criteris anteriors:

- ***CPU Scheduling Algorithm Simulator*** de la *Universitat Soonchunhyang*. [8]
  És una simulador dels algorismes de planificació de processador que inclou els algorismes tradicionals de planificació. Està desenvolupat per sistemes basat en Linux i Windows.

Aspectes positius	Aspectes negatius
– El resultat de la simulació és una taula completa que mostra els indicadors rellevants sobre la planificació per cada un dels processos.	– Només es poden realitzar simulacions a partir d'un fitxer de text, del qual no s'especifica en quin format s'ha de generar. – No és interactiu, només es mostra una imatge del resultat final de la simulació.

- ***Scheduler*** de *Jason Marcel*. [9]
  És un simulador no interactiu dels algorismes tradicionals de planificació de processos. És compatible amb sistemes basats en Linux i Windows.

Aspectes positius	Aspectes negatius
– Es mostra a cada cicle de processador l'estat d'aquest i l'acció que s'està realitzant.	– Només es poden realitzar simulacions a partir d'un fitxer de text. No es poden afegir processos en temps d'execució. – És una aplicació de consola. És difícil de configurar els paràmetres de la simulació perquè s'ha de realitzar a través de la consola.

- ***Scheduling Simulator*** de *Ruchiranga Wickramasinghe*. [10]
  És un simulador amb una interfície gràfica basada en finestres desenvolupada amb Java.

Aspectes positius	Aspectes negatius
– Interfície gràfica amb gràfics de l'estat del processador a cada cicle i una taula resum dels resultats de la simulació. – Permet simular interrupcions per bloquejar els processos actius i veure com afecta a la planificació.	– No és possible aturar la simulació una vegada s'ha iniciat ni es pot realitzar passa a passa. – Els paràmetres de la simulació no es poden modificar. – Només un simulador de l'algorisme *Round Robin* i presenta errors visuals dels resultats de la simulació.

- ***Operating System Scheduling*** de *Ahmad Yayha* i *Hamed Hijazi*. [11]
  És un simulador dels algorismes de planificació de processos que inclou els algorismes tradicionals. És una aplicació desenvolupada en Java fent ús de llibreries de tercers per la interfície gràfica.

Aspectes positius	Aspectes negatius
– Visualització dels canvis que s'han produït durant la simulació, cosa que permet realitzar un seguiment de l'algorisme. – Aplicació amb interfície gràfica basada en finestres amb possibilitat de modificar els paràmetres de la simulació.	– No es proporcionen les dependències necessàries per la compilació del projecte ni instruccions d'instal·lació.

- ***OS Sim*** d'*Alex Macia*. [7]
  És una aplicació multiplataforma amb múltiples simuladors. El simulador de planificació de processador conté els algorismes tradicionals.

Aspectes positius	Aspectes negatius
– Visualització de l'estat del processador i de les coes de processos en qualsevol instant. – Permet simular bloquejos de processos per E/S.	– No es mostra un gràfic o taula resum de la simulació, només l'estat del sistema en un instant determinat.

1.2.3 Gestió de memòria

La gestió de memòria es pot dividir en diferents simuladors. Els simuladors que s'han cercat són d'assignació de particions variables i de reemplaçament de pàgines. Tot i que degut als pocs simuladors trobats es mostren tots a la mateixa llista.

- ***OS Sim*** d'*Alex Macia.* [7]
  El simulador de memòria d'aquesta aplicació és el menys complet. Permet realitzar simulacions tant d'assignació com de paginació.

Aspectes positius	Aspectes negatius
– Es mostra un gràfic de la memòria en tot instant cosa que permet visualitzar quins processos estan carregats.	– No inclou tots els algorismes d'assignació de particions variables i el simulador de paginació no inclou cap algorisme de reemplaçament. – Les opcions de configuració no són clares i estan molt fixades.

- ***SiGeM*** de la *Universitat de Pàdua.* [12]
  És un simulador de paginació i segmentació desenvolupat en Java que conté tots els algorismes tradicionals.

Aspectes positius	Aspectes negatius
– És un simulador complet amb molts de paràmetres configurables per realitzar la simulació. – Inclou la simulació de l'intercanvi de memòria i mostra els resultats mitjançant gràfics.	– La configuració del simulador és un procés llarg i és fàcil equivocar-se. No té exemples predefinits per començar a utilitzar el simulador ràpidament.

- ***Memory Management Simulator*** de *Jamie Goodson.* [13]
  És una aplicació web que permet simular l'assignació de particions variables.

Aspectes positius	Aspectes negatius
– Desenvolupat amb una tecnologia compatible amb els dispositius actuals, tant ordinadors com mòbils.	– És compatible amb els dispositius actuals però no s'adapta correctament a les dimensions del dispositiu. – No és un simulador complet, només es pot utilitzar un únic algorisme. – No es pot controlar la simulació, el simulador sempre es troba en execució.

- ***Memory Management Simulation*** de *Dakota Szabo.* [14]
  És un simulador dels algorismes d'assignació de memòria variable realitzat en

Java.

Aspectes positius	Aspectes negatius
– Es proporcionen instruccions per compilar i executar el programa i arxius d'exemple per utilitzar amb el simulador.	– És una aplicació per consola, la interacció amb l'aplicació no és clara i no s'especifica què es pot realitzar en cada instant. – El resultat és l'estat de cada posició de memòria cosa que dificulta entendre el que està passant.

1.2.4 Valoració global

Després d'haver analitzat les solucions existents de cada un dels diferents simuladors, es poden extreure diferents conclusions:

1. La majoria dels simuladors estan desenvolupats amb una tecnologia que **ha quedat obsoleta** o no és possible fer-los funcionar correctament degut a unes instruccions d'instal·lació incompletes o inexistents.

2. La compatibilitat amb els sistemes actuals no és completa. És possible que el simulador funcioni amb algun dels sistemes operatius principals (Windows, les distribucions de Linux i OSX) però no solen funcionar sobre **tots** els sistemes operatius actuals.

   Tampoc són compatibles amb dispositius mòbils amb sistemes operatius com Android o iOS, que és un mitjà molt utilitzat a l'actualitat. A més, dins el context d'una classe de sistemes operatius té sentit suportar aquest mitjà perquè és un dispositiu que els alumnes, generalment, tenen a mà.

3. Les interfícies d'usuari basades en consola no proporcionen la millor d'experiència d'usuari perquè no permeten la visualització de gran quantitat d'informació ni d'animacions que acompanyin la simulació. És cert que proporcionen el resultat de la simulació de manera directa i clara, però no permeten tanta interactivitat com les Interfície Gràfica d'Usuari (IGU) actuals.

4. Els simuladors són diferents aplicacions, és a dir, les diferents temàtiques no es troben agrupades dins una mateixa aplicació. Dels diferents simuladors analitzats, només un disposava de les tres temàtiques però no estaven completes.

5. Les aplicacions existents no estan pensades per a **l'usuari final**. Pràcticament totes no tenen manual d'usuari com les instruccions d'instal·lació o d'ús.

   A més, començar a utilitzar un dels simuladors és una tasca que requereix temps perquè no s'inclouen exemples predefinits dins el simulador. Això requereix que l'usuari hagi de pensar un exemple, distraient-lo de l'objectiu principal del simulador que és visualitzar un algorisme.

Per tant, el desenvolupament d'un nou simulador dels algorismes tradicionals de sistemes operatius té sentit perquè s'ha comprovat que les solucions actuals no resolen de manera satisfactòria les necessitats.

Els nous simuladors han d'intentar resoldre les mancances dels simuladors existents, millorar en aspectes com la completesa i la facilitat d'ús i intentar introduir noves funcionalitats que aportin valor a l'aplicació.

CAPÍTOL 2

ANÀLISI I PLANIFICACIÓ

En aquest capítol es defineixen els requisits de l'aplicació, es mostra la metodologia i el calendari de desenvolupament.

2.1 Anàlisi de requisits

L'anàlisi de les solucions existents ha permès tenir una visió més general de com ha de ser el nou simulador i conèixer en quins aspectes es pot millorar.

El simulador és una aplicació que és utilitzada per un **únic** tipus d'usuari. Aquest usuari és una persona qualsevol que té la necessitat d'utilitzar un simulador dels algorismes de sistemes operatius. Aquest individu pot ser o bé un alumne d'una assignatura en què s'expliquen conceptes inclosos dins el simulador o bé el personal docent de l'assignatura que l'utilitza per explicar els conceptes.

En qualsevol dels dos casos, aquestes persones utilitzen el simulador de la mateixa manera perquè no existeixen funcionalitats pròpies d'un alumne o un professor, ambdós poden realitzar les mateixes operacions sobre l'aplicació.

La primera passa per definir els requisits de l'aplicació és entendre les necessitats dels usuaris que utilitzaran l'aplicació. En aquest cas, l'aplicació només és utilitzada per un **únic** tipus d'usuari.

A la figura 2.1 es mostra una visió general de les principals funcionalitats i característiques de l'aplicació que es vol desenvolupar. Aquesta es pot dividir en diferents blocs d'idees i característiques:

- La interfície d'usuari que inclou com s'ha d'organitzar visualment l'aplicació i l'experiència d'usuari.

- Les característiques generals dels simuladors, és a dir, aquelles funcionalitats que han d'estar present a tots els simuladors.

- Els simuladors de sistemes operatius que ha d'incloure l'aplicació. Per cada un d'ells s'especifiquen quins són els resultats que s'han d'obtenir i els algorismes que s'han de poder utilitzar.

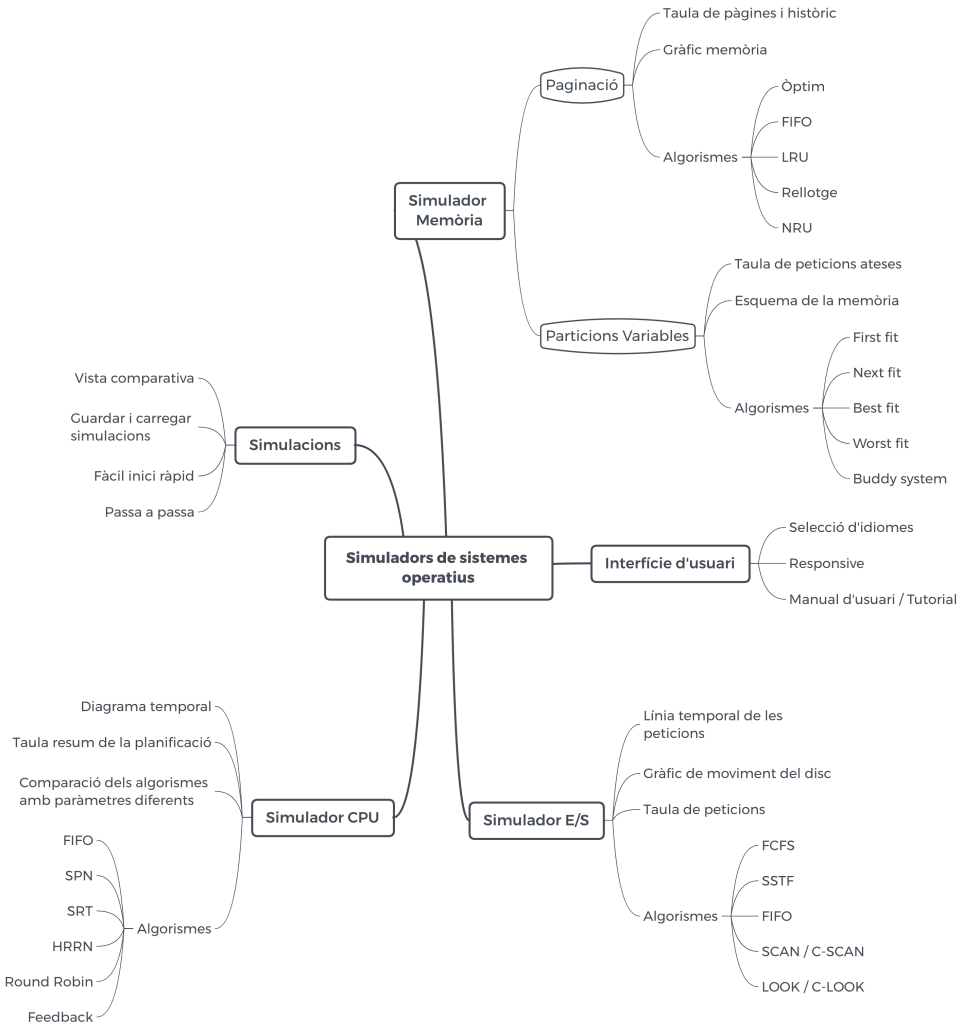


Figura 2.1: Visió general de les característiques del simulador.

A partir d'aquesta idea general del projecte es poden definir els requisits de l'aplicació. Aquests requisits s'han obtengut utilitzant diferents mecanismes:

1. A partir de les solucions existents. Amb l'anàlisi realitzat a la secció 1.2 es poden extreure quines característiques han de tenir els diferents simuladors i quines restriccions ha de tenir l'aplicació.

2. Mitjançant entrevistes amb el client que sol·licita l'aplicació, en aquest cas, la tutora del TFG.

2.1.1 Requisits funcionals

Els requisits funcionals són aquells que descriuen una funcionalitat que l'aplicació ofereix i depenen dels usuaris i del tipus de sistema en què s'executarà el software.

RF-01	**Controlar la simulació**
Descripció	L'usuari ha de poder iniciar, aturar, avançar, tornar enrere i reiniciar la simulació. Els botons de control s'han d'activar i desactivar en funció de si les opcions estan disponibles o no. La velocitat de la simulació automàtica s'ha de poder ajustar. Els paràmetres de la simulació només s'han de poder modificar quan no hi ha una simulació activa.
Objectiu	L'usuari ha de poder controlar la simulació de diferents maneres.
Dependències	-

RF-02	**Emmagatzemar i carregar la simulació**
Descripció	L'usuari ha de poder emmagatzemar la configuració de la simulació a un fitxer i carregar-la des d'un fitxer amb el format adequat. Per guardar la simulació, l'usuari haurà d'introduir el nom del fitxer i aquest es descarregarà automàticament. Per carregar la simulació, l'usuari haurà de seleccionar el fitxer que vol carregar. En cas de ser un fitxer erroni, s'indicarà que s'ha produït un error. Els fitxers de les simulacions només han d'incloure informació sobre l'estat inicial de les simulacions i no del seu progrés.
Objectiu	Permetre a l'usuari guardar i carregar simulacions des d'emmagatzematge secundari.
Dependències	-

RF-03	**Comparar els resultats**
Descripció	L'usuari ha de poder canviar entre una vista simple i una vista comparativa. La vista simple ha de permetre seleccionar un únic algorisme i executar la simulació. La vista comparativa ha de permetre seleccionar múltiples algorismes i ha de mostrar el resultat de la simulació amb cada un dels algorismes seleccionats.
Objectiu	Permetre a l'usuari observar el comportament dels algorismes sense haver d'executar la simulació amb cada algorisme.
Dependències	-

RF-04	**Explicació dels algorismes**
Descripció	L'usuari ha de poder veure una explicació sobre cada un dels algorismes dels simuladors. S'ha de mostrar un modal que expliqui resumidament el funcionament de l'algorisme seleccionat.
Objectiu	Explicar el funcionament dels algorismes a l'usuari.
Dependències	-

RF-05	**Carregar exemple**
Descripció	L'usuari ha de poder seleccionar un conjunt de peticions predeterminades per configurar ràpidament el simulador.
Objectiu	Permetre a l'usuari començar a utilitzar el simulador ràpidament.
Dependències	-

RF-06	**Explicar el funcionament de l'aplicació**
Descripció	L'usuari ha de poder visualitzar un tutorial pas a pas del funcionament de l'aplicació. El tutorial ha d'explicar el funcionament dels elements comuns de l'aplicació i aquells particulars de cada simulador. Aquest tutorial es mostrarà automàticament el primer pic que s'utilitzi un simulador i es podrà consultar en qualsevol moment.
Objectiu	Introduir a l'usuari a la interfície de l'aplicació.
Dependències	RNF-05. Facilitat d'ús

RF-07	**Canviar idioma**
Descripció	L'usuari ha de poder seleccionar l'idioma en què es presentarà l'aplicació. De manera predeterminada, l'aplicació ha de detectar l'idioma de l'usuari i utilitzar-lo si està disponible. Tots els texts de l'aplicació han d'estar ens idiomes suportats: català i castellà.
Objectiu	Presentar l'aplicació en l'idioma de l'usuari per facilitar-ne el seu ús.
Dependències	RNF-01. Sistemes d'idiomes

A continuació es presenten una sèrie de requisits propis de cada un dels simuladors, que defineixen les característiques de cada un d'ells com els algorismes que ha d'incloure i la presentació dels resultats.

Simulador d'E/S

RF-IO-01	**Configurar simulació**
Descripció	L'usuari ha de poder indicar el número de pistes, posició inicial i sentit del disc. També ha de poder seleccionar entre els següents algorismes: FCFS, SSTF, SCAN, C-SCAN, LOOK i C-LOOK.
Objectiu	Permetre a l'usuari introduir els paràmetres de la simulació.
Dependències	-

RF-IO-02	**Introduir i eliminar peticions**
Descripció	L'usuari ha de poder introduir a través d'un formulari el número de pista. També ha de poder eliminar les peticions introduïdes des de la llista mitjançant un botó. Només s'ha de permetre modificar les peticions si la simulació no ha començat.
Objectiu	Permetre a l'usuari introduir els paràmetres de la simulació.
Dependències	-

RF-IO-03	**Llistar peticions**
Descripció	L'usuari ha de poder visualitzar quines peticions estan actualment introduïdes al simulador. Durant la simulació, el llistat de peticions s'anirà actualitzant per indicar si la petició ha estat processada o encara s'ha de processar.
Objectiu	Permetre a l'usuari visualitzar i modificar els paràmetres de la simulació.
Dependències	RF-IO-2. Introduir i eliminar peticions

RF-IO-04	**Visualitzar resultats**
Descripció	L'usuari ha poder veure una taula i un gràfic que indiqui l'ordre en què s'han atès les peticions així com el desplaçament total. També podrà visualitzar una representació d'un gràfic fent el moviment del capçal.
Objectiu	Presentar els resultats de la simulació gràficament a l'usuari.
Dependències	-

Simulador de CPU

RF-CPU-01	**Configurar simulació**
Descripció	L'usuari ha poder seleccionar i configurar els següents algorismes: FIFO, Round Robin, SPN, SRT, HRN, Feedback. Els algorismes Round Robin i Feedback han de permetre establir el quantum de la simulació. A la vista comparativa, s'ha de poder comparar un mateix algorisme amb diferents paràmetres.
Objectiu	Permetre a l'usuari configurar els paràmetres de la simulació.
Dependències	-

RF-CPU-02	**Introduir i eliminar peticions**
Descripció	L'usuari ha poder introduir i eliminar processos de la simulació. Per afegir un procés s'haurà d'omplir un formulari en què es demanarà la durada del procés i en quin instant s'introdueix el procés. Per eliminar un procés haurà de prémer un botó de la llista. Només s'ha de permetre modificar la llista de processos si la simulació no ha començat.
Objectiu	Permetre a l'usuari indicar quines sol·licituds ha d'atendre el simulador.
Dependències	-

RF-CPU-03	**Llistar processos**
Descripció	L'usuari ha poder visualitzar quins són els processos que s'han introduït al simulador i les seves característiques (durada i arribada). Durant la simulació, s'actualitzarà la taula per indicar quin procés s'ha completat.
Objectiu	Permetre a l'usuari visualitzar els paràmetres de la simulació.
Dependències	RF-CPU-02. Introduir i eliminar peticions

RF-CPU-04	**Visualitzar resultats**
Descripció	L'usuari ha poder visualitzar l'estat actual del processador, les coes de processos, un gràfic que indica per cada instant quin procés s'ha estat executant i una taula resultat que per cada procés indica el temps que ha utilitzat (temps de retorn, temps de retorn normalitzat).
Objectiu	Permetre a l'usuari visualitzar gràficament el resultat dels simuladors i obtenir informació del rendiment de cada algorisme.
Dependències	-

Simulador de Memòria

RF-MEM-01	**Configurar simulació**
Descripció	L'usuari ha de poder seleccionar entre els diferents mecanismes de planificació i algoritmes de cada un d'ells: • Assignació de particions variables amb algorismes first fit, best fit, next fit, worst fit i buddy system. • Reemplaçament de pàgines a la paginació amb els algorismes òptim, FIFO, LRU, clock i NRU. S'ha de poder especificar la capacitat total de la memòria i les dimensions de cada un dels mecanismes.
Objectiu	Permetre a l'usuari configurar el simulador d'assignació de memòria.
Dependències	-

RF-MEM-02	**Introduir i eliminar sol·licituds**
Descripció	L'usuari ha de poder introduir sol·licituds i eliminar-les. Per introduir la sol·licitud a les particions variables s'haurà d'omplir un formulari indicant la memòria necessària, l'instant en què arribarà i la durada de la sol·licitud. Per introduir la sol·licitud a la paginació, s'haurà d'introduir el procés i la pàgina que es vol consultar.
Objectiu	Permetre a l'usuari modificar el llistat de peticions de sol·licitud de memòria.
Dependències	-

RF-MEM-03	**Llistar les peticions**
Descripció	L'usuari ha de poder consultar el llistat de peticions que ha introduït. El llistat de peticions canviarà durant la simulació per indicar quines peticions s'han completat.
Objectiu	Permetre a l'usuari conèixer quines peticions processarà el simulador.
Dependències	RF-MEM-02. Introduir i eliminar sol·licituds

RF-MEM-04	**Visualitzar resultats**
Descripció	L'usuari ha de poder consultar a l'assignació de particions variable un gràfic que mostri l'estat actual de la memòria (espai ocupat per cada procés, fragmentació externa i interna). Al simulador de reemplaçament de pàgines s'ha de mostrar l'ocupació de la memòria i la taula de pàgines de cada un dels processos.
Objectiu	Visualitzar gràficament els resultats de la simulació.
Dependències	-

2.1.2 Requisits no funcionals

Els requisits no funcionals són les propietats i restriccions del sistema que poden condicionar l'etapa de desenvolupament. Aquests poden imposar restriccions i són més difícils d'avaluar i comprovar que se satisfan.

RNF-01	**Sistema d'idiomes**
Descripció	S'ha d'utilitzar un sistema que permeti introduir nous llenguatges a l'aplicació. El canvi d'idioma no ha de suposar una pèrdua d'informació. Cada idioma ha de tenir el seu arxiu de texts.
Objectiu	Facilitar la creació, integració i manteniment dels idiomes a l'aplicació.
Dependències	-

RNF-02	**Compatibilitat de dispositius**
Descripció	L'aplicació ha de funcionar correctament als dispositius (mòbils, tablets o ordinador) que permeti executar un dels següents navegadors web: Chrome 87, Firefox 84, Safari 14.
Objectiu	Definir els requisits mínims del dispositiu sobre el qual l'aplicació ha de funcionar correctament.
Dependències	-

RNF-03	**Resolució del dispositiu**
Descripció	L'aplicació s'ha de visualitzar correctament a dispositius amb una amplada horitzontal igual o superior, en píxels, a les següents: 320, 375, 425, 768, 1024, 1440.
Objectiu	Definir les dimensions amb les quals l'aplicació es visualitzarà correctament.
Dependències	-

RNF-04	**Ús offline**
Descripció	L'aplicació s'ha de poder utilitzar sense una connexió activa a Internet.
Objectiu	Permetre l'ús de l'aplicació sense accés a Internet.
Dependències	-

RNF-05	**Facilitat d'ús**
Descripció	L'aplicació ha de ser fàcil d'utilitzar. Es considerarà que l'aplicació és fàcil d'utilitzar si després d'haver realitzat el tutorial passa a passa l'usuari coneix el funcionament de l'aplicació.
Objectiu	Permetre a l'usuari entendre el funcionament de l'aplicació.
Dependències	-

2.2 Planificació temporal

Per poder especificar la planificació temporal del projecte és necessari determinar quina metodologia de desenvolupament es seguirà per realitzar el projecte.

Aquest projecte consisteix en desenvolupar una aplicació **no crítica** per un client. L'aplicació té una gran component d'interfície d'usuari, que pot canviar així com s'avança en el desenvolupament de l'aplicació.

En aquest projecte no té sentit realitzar un desenvolupament en cascada perquè el client pot proporcionar la seva opinió sobre una versió del producte no final que provoqui canvis en els requisits. Per tant, té sentit utilitzar una **metodologia incremental** perquè aquest sistema permet presentar diferents versions del producte i obtenir una retroalimentació del client que pot condicionar el desenvolupament.

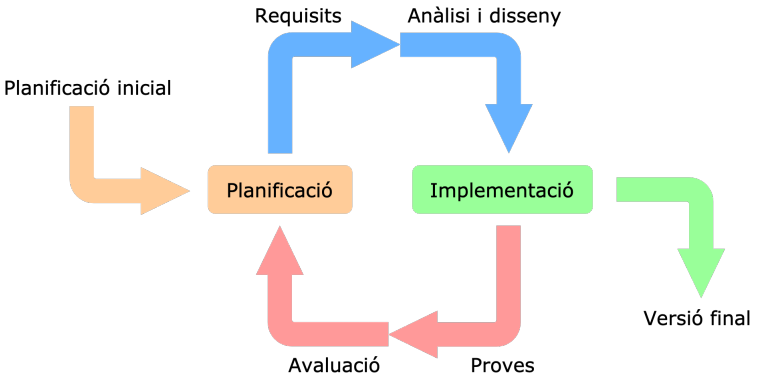


Figura 2.2: Diagrama d'una metodologia incremental.

L'aplicació es pot dividir en tres grans seccions, que són cada un dels simuladors. Cada una d'aquestes seccions ha estat una de les versions de l'aplicació.

- **Primera versió**. La primera versió està formada per l'estructura general de l'aplicació i el simulador d'E/S. Aquesta versió no inclou el sistema d'idiomes ni seccions d'ajuda.
- **Segona versió**. La segona versió incorpora els canvis suggerits pel client després de la primera entrega i el simulador de CPU. També s'ha introduït el sistema d'idiomes amb un únic idioma i la secció d'ajuda dels simuladors existents.
- **Tercera versió**. Aquesta versió inclou els canvis que s'han de realitzar després de la segona entrega i el simulador de memòria, tant el simulador de particions variables com el de reemplaçament de pàgines.
- **Versió final**. La darrera versió, a més d'introduir els canvis del client, també inclou la traducció de l'aplicació a tots els idiomes i la creació de les versions executables de l'aplicació.

La planificació temporal de la figura 2.3 i la taula 2.1 són l'estimació inicial de les hores de feina de cada un dels diferents paquets de feina.

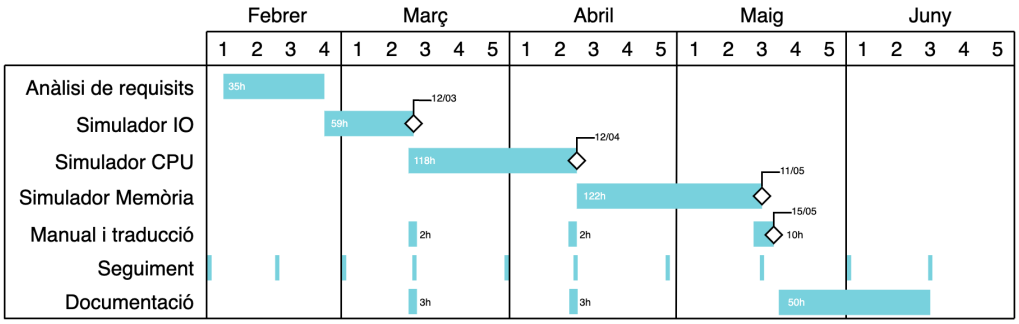


Figura 2.3: Cronograma simplificat de la planificació temporal.

Tasca		**Hores**
	Anàlisi de requisits	35
Implementació	Simulador IO	59
	Simulador CPU	118
	Simulador Memòria	122
	Manual i traducció	14
	Seguiment	7.5
	Documentació	56
	Total	411,5

Taula 2.1: Hores previstes de cada una de les tasques.

- L'etapa **d'anàlisi de requisits** consisteix en analitzar les solucions existents i determinar quins són els requisits de la nova aplicació.
- L'etapa pròpiament **d'implementació** de l'aplicació està dividida en els grups dels diferents simulador, que són cada una de les versions.
- Les tasques incloses al grup **"Manual i traducció"** són tasques relacionades amb la redacció dels manuals d'usuari, els tutorials de l'aplicació i la traducció dels texts.
- Les **reunions de seguiment**, aproximadament d'uns 45 minuts cada una d'elles, serveixen per comprovar que el projecte s'està desenvolupant correctament, és a dir, satisfent les necessitats del *product owner*. Aquestes reunions estan programades cada dues setmanes, tot i que es poden moure per fer-les coincidir amb l'entrega d'una versió.
- La **documentació** consisteix en documentar els problemes que s'han resolt durant el desenvolupament d'una versió i la redacció del document del projecte.

Sobre el cronograma s'han establert unes fites, que són unes dates en què s'han d'haver assolit uns determinats objectius. Les fites marcades en el diagrama temporal són cada una de les entregues de l'aplicació, que es poden veure a la taula 2.2.
També s'ha de tenir en compte que després d'entregar una versió poden aparèixer tasques per implementar els canvis suggerits pel client, cosa que pot provocar alguna

desviació sobre les dates d'entrega de les versions.

Fita	Data
Simulador IO	12/03/2021
Simulador CPU	12/04/2021
Simulador Memòria	11/05/2021
Versió final	15/05/2021

Taula 2.2: Fites del projecte

El fet de començar pel simulador amb una estimació de feina més baixa no és una decisió arbitrària. El desenvolupament del simulador amb una quantitat menor de funcionalitats permetrà establir el disseny de l'aplicació, tant visual com lògic. D'aquesta manera, la resta de simulador amb un nivell de complexitat més elevat, podran seguir el mateix disseny, facilitant el procés de desenvolupament.
A més, aquest període també servirà per acabar de conèixer les eines de desenvolupament i les bones pràctiques de la tecnologia seleccionada.

CAPÍTOL 3

TECNOLOGIES I DISSENY

En aquest capítol s'explica quines tecnologies s'han seleccionat per desenvolupar l'aplicació i com s'ha dissenyat l'arquitectura de l'aplicació.

3.1 Tecnologies de desenvolupament

A l'hora de seleccionar les tecnologies s'han de tenir en compte els requisits no funcionals, que poden determinar l'elecció d'una tecnologia o una altra. En aquest cas el projecte té uns requisits no funcionals (RNF-02 i RNF-03) que obliguen que el resultat funcioni en una gran varietat de plataformes i dispositius, inclosos els dispositius mòbils.

La primera aproximació pot ser dissenyar una aplicació nativa per cada una de les plataformes. Aquesta aproximació, com a mínim, suposarà una aplicació per les plataformes d'escriptori si s'utilitza una tecnologia multiplataforma i una aplicació per *iOS* i *Android*. El desenvolupament de tres aplicacions independents requereix d'una quantitat de temps superior a la disponible per aquest projecte.

Per evitar el desenvolupament d'aplicacions independents s'ha d'utilitzar una tecnologia compatible amb les diferents plataformes i dispositius. Les aplicacions webs són unes aplicacions compatibles amb tots els dispositius que suporten un navegador web i es poden adaptar a les dimensions dels dispositius.

A més, si és necessari entregar una versió per escriptori que no requereixi d'un navegador web, es poden crear versions executables de l'aplicació web. Les aplicacions web d'escriptori es poden aconseguir fent ús de diferents tècniques:

- Les eines proporcionades pels navegadors instal·lats per l'usuari, donant lloc a les *progressive web apps* que són aplicacions web executades amb el mateix navegador de l'usuari.

- La creació d'una versió empaquetada de l'aplicació que inclou el seu propi navegador. En aquest cas, l'aplicació es presenta com un executable que conté el seu propi navegador web, utilitzat únicament per visualitzar l'aplicació web.

Per tant, desenvolupar els simuladors com una aplicació web és una opció acceptable perquè s'aconseguirà una **única versió** de l'aplicació que serà **compatible** amb una gran varietat de dispositius i plataformes, agilitzant el procés de desenvolupament i manteniment de l'aplicació.

Ara bé, una vegada s'ha decidit que l'aplicació es desenvoluparà com una aplicació web s'ha de seleccionar amb quina tecnologia concreta es realitzarà, ja que existeix una gran varietat d'entorns de treball i llibreries per realitzar una aplicació web.

3.1.1 Entorn de treball

El desenvolupament d'aplicacions web es realitza fent ús de diferents llenguatges i cada un d'ells té diferents objectius:

- L'**Hyper Text Markup Language (HTML)** és un llenguatge de marques que permet definir l'estructura de les pàgines web. Els elements del document s'organitzen seguint una jerarquia. També és amb aquest llenguatge que s'especifica el contingut de la pàgina, com texts o imatges.

- El **Cascading Style Sheets (CSS)** o fulles d'estil serveixen per indicar les característiques visuals dels elements del document HTML. Aquest llenguatge s'utilitza per definir l'aspecte visual de la pàgina web.
  L'adaptació de les pàgines a les diferents resolucions es realitza fent ús d'aquest llenguatge, ja que es tracta d'un aspecte visual.

- El **JavaScript** és el llenguatge de programació amb el qual es pot modificar el contingut de la pàgina web de manera dinàmica. El navegador proporciona les funcions necessàries per poder modificar el contingut i donar resposta als esdeveniments que genera l'usuari.

Les tecnologies que es compararan a continuació es faran tenint en compte diferents criteris:

- *La corba d'aprenentatge* és la dificultat que té una persona per aprendre a utilitzar una tecnologia, quan més ràpidament es domini una tecnologia més fàcil serà el desenvolupament.

- *La comunitat i l'estabilitat de la tecnologia*. Les tecnologies noves o presentades recentment solen tenir canvis en el seu funcionament de manera freqüent fet que pot dificultar el desenvolupament i el manteniment de l'aplicació. També és important que la tecnologia tengui una comunitat extensa que documenti les funcionalitats, i idealment, que realitzi contribucions a la tecnologia.

- *La manipulació del contingut del document* és el grau de facilitat per canviar la jerarquia del document i el seu contingut de manera dinàmica en funció de la interacció de l'usuari. Aquest punt és important perquè el simulador ha de mostrar els resultats a través d'una interfície gràfica.

- *El coneixement previ per part del programador* és l'experiència que té el programador del projecte amb la tecnologia. Si el programador ja té coneixements de la tecnologia podrà invertir més hores en la implementació de l'aplicació i no en conèixer la tecnologia.

Vanilla JS

El *vanilla JS* o JavaScript és el propi llenguatge de programació JS sense cap tipus de llibreria addicional, és a dir, és una versió *pura* del llenguatge de programació que suporten els navegadors.

Per un costat els principals avantatges d'utilitzar aquesta tecnologia són els següents:

- La corba d'aprenentatge és molt baixa perquè es tracta del propi llenguatge de programació i de les funcions per manipular el document.
- El programador té experiència amb aquest llenguatge de programació.
- La comunitat de JS és extensa i en constant creixement. Existeixen una gran varietat de repositoris de projectes de codi lliure que poden facilitar el funcionament.

Per l'altre té un principal inconvenient amb la manipulació del document de manera dinàmica. La manipulació és una tasca simple perquè les funcions disponibles són de baix nivell d'abstracció, però la creació d'una interfície dinàmica amb gràfics és una tasca complexa per realitzar a aquest nivell.

Es tracta d'una tecnologia senzilla per realitzar tota una aplicació dinàmica fent ús exclusiu de JS. Tot i això, alguns aspectes només es poden realitzar fent ús de les funcions bàsiques del llenguatge, per tant, és important conèixer-les.

jQuery

El *jQuery* és una llibreria desenvolupada sobre el llenguatge JS creada al 2006. Aquesta llibreria facilita el recorregut sobre el Document Object Model (DOM), la manipulació d'esdeveniments, les animacions entre d'altres. [15]

Aquesta llibreria proporciona una capa més d'abstracció sobre el desenvolupament d'aplicacions webs perquè canvia la forma de modificar el document amb funcions pròpies. A més, introdueix un sistema d'animacions, que pot resultar útil per mostrar els resultats del simulador.

Els principals aspectes positius són els següents:

- La corba d'aprenentatge també és baixa si ja es coneix el llenguatge de programació JS, ja que la llibreria només introdueix noves funcions i no canvia la manera de programar.
- El programador coneix i ha treballat amb aquesta tecnologia però no amb les darreres versions de la llibreria.

Tot i això, existeixen alguns aspectes negatius de fer feina amb aquesta tecnologia:

- És un projecte del 2006 que s'ha anat actualitzant al llarg dels darrers anys però la comunitat de desenvolupadors prefereix altres tecnologies. Tot i això segueix sent una llibreria present a la majoria de les pàgines webs actuals. [16]

- La manipulació del document es realitza amb un nivell d'abstracció baix, igual que al JS pur. La llibreria introdueix funcions per manipular de manera més directe el document però des d'un paradigma imperatiu.

Els següents entorns de treball o llibreries, a diferència dels anteriors fan ús d'**un paradigma declaratiu**. Als paradigmes declaratius no s'especifiquen les instruccions exactes per realitzar una determinada acció, sinó que s'indica què es vol aconseguir i és la pròpia llibreria que s'encarrega d'aconseguir realitzar l'acció desitjada.

Si aquest plantejament s'adapta al món de les interfícies d'usuari, el programador no especifica com s'ha de representar una determinada informació, sinó que indica **l'estat intern** de l'aplicació i és l'aplicació que es renderitza respectant l'estat intern.

Així, aquest plantejament permet definir interfícies d'usuari fàcilment perquè és suficient en determinar com s'ha d'actualitzar l'estat intern de l'aplicació i com s'ha de visualitzar aquest estat, però no és necessari indicar les instruccions específiques per representar l'estat.

Vue

Vue.js és un entorn de treball JavaScript creat al 2014 utilitzat per crear interfícies d'usuari. La principal diferència amb la resta de llibreries tradicionals com *jQuery* és el plantejament declaratiu de la llibreria. [17]

La renderització de l'estat intern és realitza mitjançant un ***Virtual DOM***, que no és més que una representació interna de la jerarquia dels elements del DOM. Ara bé, quan s'ha de representar un canvi, el canvi no es realitza directament sobre el DOM, sinó que es realitza sobre la representació virtual del document i si és necessari s'actualitzarà el document real.

Amb aquest plantejament s'eviten canvis innecessaris sobre la interfície d'usuari, ja que si les dades del problema no han canviat, tampoc ho ha fet la interfície d'usuari. És cert que aquesta tècnica introdueix altres processos addicionals com la manipulació del *virtual DOM*.

El principal aspecte positiu d'aquesta tecnologia és que la manipulació del document és gestionada per la pròpia llibreria fet que facilita la creació d'interfícies dinàmiques.

En contra, es poden destacar els següents aspectes negatius:

- La corba d'aprenentatge és moderada perquè suposa conèixer les peculiaritats del nou entorn de treball. Ara bé, el llenguatge de programació segueix sent JS, per tant, només es tractaria d'una adaptació a les funcions de l'entorn.
- És un entorn de treball mantengut per la comunitat que ha petit canvis importants al llarg dels anys. Aquest canvis han provocat que les aplicacions desenvolupades amb versions anteriors no siguin compatibles amb les noves versions, cosa que dificulta el manteniment de l'aplicació.
- El programador no té experiència amb aquest *framework*, cosa que dificultaria les primeres setmanes de desenvolupament.

Svelte

Svelte és una eina creada al 2016 per crear interfícies d'usuari declaratives però en lloc d'utilitzar la tècnica del *Virtual DOM* i determinar si la interfície s'ha d'actualitzar és un compilador que genera codi que s'encarrega d'actualitzar el DOM directament quan és necessari. [18]

La principal diferència d'aquesta eina respecte les altres llibreries declaratives és que aquesta és un compilador que genera codi JS que s'encarrega d'actualitzar la interfície quan l'estat intern canvia, és a dir, no és necessari comparar entre una representació virtual del DOM i la realitat.

A més, pel fet de generar codi executable directament, els resultats són més petits perquè no s'ha de transmetre la pròpia llibreria i són més ràpids perquè el procés d'actualització del DOM és més eficient.

L'avantatge d'utilitzar aquesta llibreria és el plantejament declaratiu que facilita la creació d'interfícies d'usuari. A més de l'eliminació del concepte del *virutal DOM* que provoca que el codi resultant sigui més lleuger i ràpid.

En contra es poden comentar els següents aspectes negatius:

- És una eina de *recent* creació que pot introduir canvis importants en l'entorn de treball fent que el codi de les aplicacions de versions antigues no sigui compatible amb les noves versions. La comunitat tampoc és tan extensa com les altres alternatives.
- L'aprenentatge d'aquesta tecnologia és més complex perquè no només és un canvi de paradigma respecte els tradicionals, també canvia lleugerament la sintaxi i la nomenclatura del codi HTML.
- El programador no té experiència amb aquesta tecnologia.

React

React és una llibreria publicada per primera vegada al 2013 i mantenguda per *Facebook* que s'utilitza per crear interfícies d'usuari. [19] Igual que *Vue* fa ús d'un arbre virtual

per determinar quan s'ha d'actualitzar el document real.

L'ús d'aquesta llibreria té els següents aspectes positius:

- És una llibreria estable: els canvis que s'introdueixen no obliguen a realitzar canvis importants al codi existent i té una gran comunitat.
- La creació d'interfícies dinàmiques és una tasca senzilla gràcies al paradigma declaratiu de la llibreria.
- El programador té experiència amb aquesta tecnologia.

Tot i que la corba d'aprenentatge és moderada: és fàcil començar a desenvolupar una aplicació però és important dominar les peculiaritats de la llibreria.

La llibreria amb què es desenvoluparà l'aplicació és ***React*** perquè és una llibreria **estable** que introdueix **el paradigma declaratiu** a les interfícies d'usuari i té **una extensa comunitat** que documenta la llibreria. A més, el programador té **experiència prèvia** amb aquesta tecnologia cosa que agilitzarà el procés de desenvolupament.

3.1.2 TypeScript

El llenguatge de programació JS és un llenguatge **no tipat**, és a dir, les variables no tenen un tipus associat i poden canviar al llarg de l'execució del programa. En aquest tipus de llenguatges és responsabilitat del programador assegurar-se que el tipus de les variables és l'adequat en tot moment, i en cas de ser un tipus no esperat, gestionar correctament la situació.

Els llenguatges de programació **estrictament tipats** tenen uns avantatges respecte els llenguatges no tipats:

- Els programes resultants són coherents des d'una perspectiva de tipus. Aquest fet elimina una gran quantitat d'errors que es poden cometre durant el desenvolupament.
- Els tipus de dades són un tipus de documentació, ja que totes les variables i funcions tenen un tipus associat.
- Els editors de codi proporcionen assistència amb l'autocompletat de codi, que seria impossible sense un tipat estricte.

En contra, aquests llenguatges són més pesats d'escriure perquè és necessari definir els tipus a tots els punts del codi, cosa que pot retardar la implementació de les funcionalitats pròpies de l'aplicació durant les primeres etapes del desenvolupament.

Per desenvolupar aquesta aplicació s'ha considerat que els avantatges dels llenguatges tipats són superiors als seus inconvenients. Per aquest motiu, s'ha decidit utilitzar ***TypeScript (TS)*** per desenvolupar l'aplicació.

TypeScript és un superconjunt del llenguatge de programació JS, és a dir, és un llenguatge de programació que inclou totes les funcionalitats de JS i n'introdueix de noves. Les característiques més destacables són:

- Les *interfaces*, que són un tipus que defineix les propietats i mètodes que ha d'incloure un tipus que l'implementa.
- Els tipus genèrics, que permet definir tipus que facin ús d'un tipus qualsevol, cosa que evita la repetició de codi per treballar amb diferents tipus.
- El tipat estricte, que obliga que totes les variables, els paràmetres de funcions i les funcions tenguin un tipus associat.

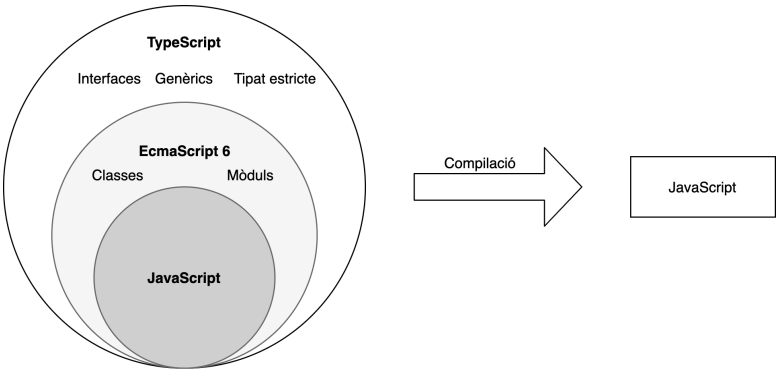


Figura 3.1: Característiques de les variacions de JS.

Els navegadors web i motors d'execució JS només poden executar codi JS. És necessari aplicar un procés de compilació per convertir el codi TS a codi JS que pugui ser executat.
Aquest procés també proporciona un altre avantatge perquè es pot especificar en quin estàndard s'ha de generar el codi JS. D'aquesta manera, amb un mateix codi TS es pot aconseguir un codi JS compatible amb una gran varietat de navegadors, que poden no implementar totes les funcionalitats del darrer estàndard.

3.1.3 React

A la secció 3.1.1 s'ha introduït breument en què consisteix la llibreria però no s'han explicat totes les característiques.

React és una llibreria que facilita la creació d'interfícies d'usuari interactives. El programador només ha de definir com és l'estat intern de l'aplicació i les vistes per cada un dels estats. És la pròpia llibreria que s'encarregarà d'actualitzar eficientment la interfície d'usuari amb les dades de l'estat.

Aquesta llibreria també fomenta els principis de **la programació modular** mitjançant la divisió de l'aplicació en **components**. Aquests components han de ser el més simples possible i han d'encapsular la seva lògica i estat. Existeixen dues maneres de crear components propis a React:

1. Fent ús d'un esquema **orientat a objectes**, que consisteix en crear un classe per cada un dels components. Aquesta classe ha de sobrecarregar mètodes com el de renderitzat o els propis del cicle de vida dels components.

2. Utilitzant l'esquema dels **components funcionals** i els *hooks*. Aquest mètode es va introduir posteriorment amb l'objectiu de resoldre alguns problemes relacionats amb la duplicitat de codi i alguns cicles de vida dels components.
   Els *hooks* són un conjunt de funcions que s'executen en determinats punts del cicle de vida del component que permeten manipular l'estat del component i utilitzar determinades funcions de React.

Tot i que no s'espera eliminar la possibilitat de crear components fent ús de classes és millor aprendre i dissenyar l'aplicació així com recomana el manual oficial de React.

De manera breu, els components tenen un estat intern i unes propietats especificades pel component superior en la jerarquia. El component només s'ha d'actualitzar quan es produeix un canvi a l'estat intern o les propietats. A més, només s'han de modificar aquells elements del DOM que realment hagin estat modificats.

L'ús del llenguatge TS facilitarà la creació i l'ús dels components perquè permetrà definir les propietats i el seu tipus que tenen cada un dels components. En cas d'utilitzar incorrectament un component, es produirà un error de compilació.

Les pàgines webs s'estructuren fent ús de HTML, però en el cas de React s'utilitza una sintaxi modificada del JS per estructurar el document coneguda com **JSX**. Aquesta sintaxi permet definir tots els elements propis de HTML i incloure codi JS dins els propis elements.

3.1.4 Eines de feina

Git

Git és una eina de control de versions que permet gestionar les versions de l'aplicació, tant les versions finals com les que es produeixen durant el desenvolupament.

A més, aquesta eina s'ha combinat amb la plataforma *GitHub* que permet allotjar el repositori de codi al núvol. L'ús d'aquesta plataforma ha permès la implementació d'un cicle semblant al d'entrega contínua o *continuous delivery*.

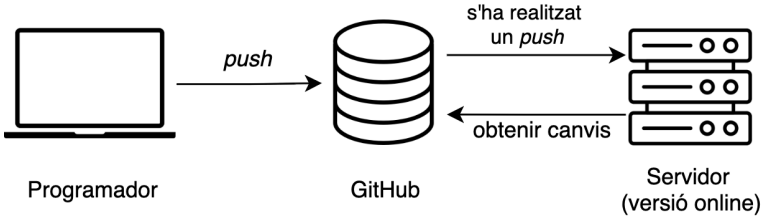


Figura 3.2: Esquema del cicle d'entrega contínua.

El cicle d'entrega contínua permet mostrar els canvis al *product owner* de manera ràpida perquè pot consultar l'aplicació en qualsevol instant.

Una vegada s'ha completat una característica o funcionalitat de l'aplicació es publica al repositori de *GitHub*. El repositori de *GitHub* està configurat per notificar a un servidor remot quan es produeix un canvi a la branca principal del repositori. Una vegada el servidor remot rep aquesta notificació consulta els canvis, compila i publica la nova versió de l'aplicació.

Entorn NodeJS

La configuració de l'entorn *NodeJS* s'ha realitzar mitjançant el gestor de versions *Node Version Manager*. Una vegada s'ha instal·lat aquest programa la instal·lació de les versions de *NodeJS* es pot instal·lar un versió amb la següent instrucció:

```
nvm install v15.0.0
```

Per inicialitzar el projecte de l'aplicació React s'ha utilitzat l'eina `create-react-app` que s'encarrega de configurar totes les dependències de React. També permet inicialitzar el projecte a partir d'una plantilla, com pot ser una plantilla amb suport de TS.

```
npx create-react-app nom-projecte --template typescript
```

3.2 Disseny

Els **patrons de disseny** o **patrons arquitectònics** descriuen l'organització d'un sistema que s'ha utilitzat amb èxit anteriorment i dels quals es coneixen els avantatges i inconvenients. Aquests patrons es poden utilitzar com una plantilla per dissenyar noves aplicacions semblants a les que ja s'han desenvolupat anteriorment.

Les aplicacions que es basen en la visualització de dades i en què els requisits de les interfícies pot canviar durant el procés de desenvolupament, com és el cas d'aquest projecte, és convenient utilitzar un patró que separi **les dades** de **la vista**. D'aquesta manera és possible canviar fàcilment com es visualitzen les dades sense haver de modificar les dades ni la seva manipulació.

Un patró que separa la vista i les dades és el patró Model Vista Controlador (MVC) en què es diferencien 3 components independents però que es comuniquen entre ells. El patró utilitzat en aquesta aplicació és un derivat del MVC, el patró **Model Vista Presentador (MVP)**.

El patró MVP, que es pot veure a la figura 3.3, també estructura els diferents components en 3 components independents:

- La **vista** és el component que s'encarrega de la visualització de les dades i d'atendre els esdeveniments de l'usuari. L'usuari de l'aplicació interactua amb aquest component però aquest component només s'encarrega de notificar al presentador l'acció que s'ha realitzat.

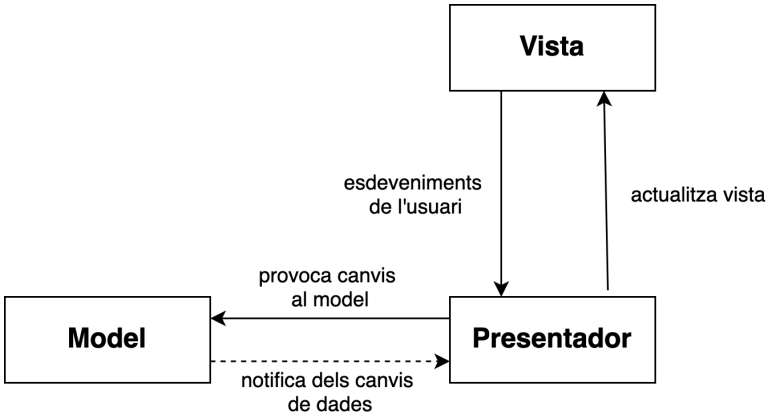


Figura 3.3: Esquema del patró MVP.

- El **model** és el component que conté les dades del problema i la lògica per manipular aquestes dades. També s'encarrega de notificar al presentador si s'ha produït qualque canvi en les dades com a conseqüència d'una determinada acció.

- El **presentador** és una capa entre el model i la vista que s'encarrega d'obtenir les dades del model i de convertir-la en el format adequat per la vista. L'altra funció que executa és processar els esdeveniments de l'usuari i determinar com afecten al model.

3.2.1 Model

Les dades del problema són els simuladors i els resultats d'aquests. El model està format pels diferents tipus de simuladors i les estructures necessàries per poder representar la seva informació.

Existeixen un conjunt d'operacions que són comunes a tots els simuladors que són les funcions de controlar l'estat del simulador, com avançar o retrocedir en l'estat de la simulació i esborrar les dades. Tot i que aquestes operacions són comunes a tots els simuladors, la implementació de les funcions és concreta a cada simulador.

Aquest fet es pot veure a la implementació de l'aplicació, ja que la classe `Simulator` és una classe **abstracte**, és a dir té mètodes que han d'implementar les classes que hereten d'ella.

Per poder satisfer el requisit funcional RF-03, que especifica que s'han de poder comparar múltiples simulacions al mateix temps, és necessari introduir unes estructures addicionals que permetin tenir múltiples instàncies simultànies del simulador.

Aquesta estructura s'ha anomenat `Manager` i s'encarrega d'emmagatzemar les dades introduïdes per l'usuari i gestiona l'existència de múltiples simuladors.

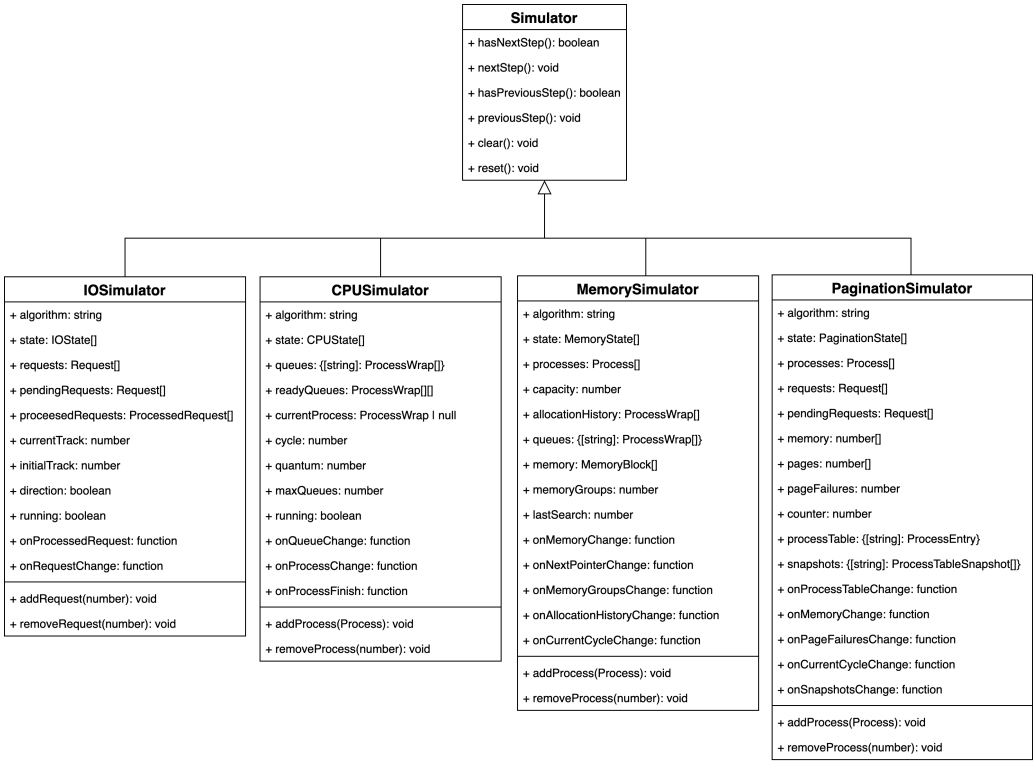


Figura 3.4: Diagrama de classes dels simuladors.

La classe `Manager` és un tipus genèric, és a dir, que es pot utilitzar amb *qualsevol* tipus. Ara bé, en aquest cas no pot ser qualsevol tipus perquè només s'accepten tipus que hereten de la classe `Simulator`.

De manera resumida, aquesta classe s'encarrega d'utilitzar un únic simulador o múltiples simuladors en funció de si l'usuari es troba a la vista simple o comparativa, respectivament.

En aquest cas sí té sentit implementar funcions de la classe `Manager` que controlin els simuladors perquè **tenen el mateix significat** independentment del tipus de simulador.

- `hasNextStep` i `hasPreviousStep` determinen si existeix una passa següent o anterior a la simulació. En el cas de la vista simple el valor està determinat per l'únic simulador, en canvi, a la vista comparativa, depèn de l'estat de cada un dels simuladors.

- De la mateixa manera que les funcions anteriors, `nextStep` i `previousStep`, s'encarreguen d'avançar o tornar a l'estat anterior, respectivament, a un simulador o als diferents simuladors de la comparació.

El `Manager` de cada un dels simuladors s'encarregarà d'implementar les funcions específiques de cada un dels simuladors, especialment els mètodes que permetin al presentador accedir a la informació que es vol representar.

Figura 3.5: Diagrama de classes de les estructures auxiliars.

3.2.2 Vista

La vista s'ha organitzat en components que s'encarreguen de realitzar tasques molt concretes, així com estableix el manual de React. Aquests components s'han ajuntat per construir les diferents seccions de l'aplicació, formant una jerarquia de components representada a la figura 3.6.

El component `App` és el de major nivell, que és el conté tota l'aplicació, i s'encarrega de determinar quin és el contingut que s'ha de mostrar.

Els simuladors estan organitzats en diferents pàgines, que són components diferents. Cada simulador té els seus propis components propis per representar les dades, ja que canvien d'un simulador a l'altre. La part important d'aquests components és que no depenen de la lògica del simulador, és a dir, el contingut que mostren només depèn dels seus paràmetres d'entrada.

3.2.3 Presentador

El presentadors, dins l'esquema de l'aplicació React, s'han implementat com les funcions que manipulen l'estat intern dels components i contenen les referències al model. Aquestes funcions s'encarreguen de gestionar tant els esdeveniments de l'usuari i

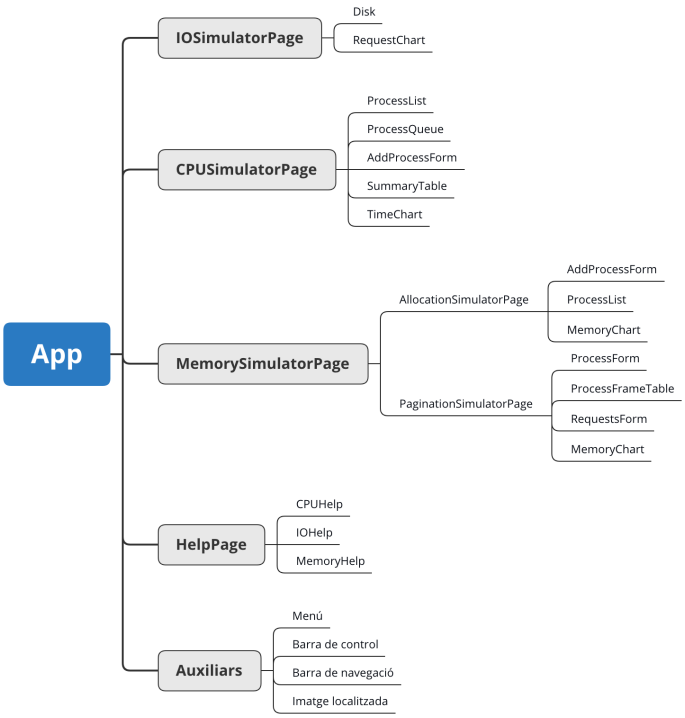


Figura 3.6: Components propis de l'aplicació.

redirigir-los correctament al model com d'actualitzar la vista.

Cada un dels simuladors conté el seu propi presentador que s'encarrega de manipular l'estat intern del component que té el simulador per actualitzar la vista. Aquesta actualització de l'estat intern es realitza mitjançant un sistema basats en esdeveniments: el model notifica al presentador que la informació ha canviat i el presentador adapta aquesta informació per a la vista.

La separació entre la presentació de les dades a la vista i l'actualització d'aquestes dades al model permet canviar la manera de representar la informació sense haver de realitzar altres canvis. Aquesta independència entre els components facilita el procés de desenvolupament i el manteniment de l'aplicació.

CAPÍTOL 4

IMPLEMENTACIÓ

En aquest capítol es comenten els detalls de la implementació de les funcionalitats, els resultats i els problemes superats durant aquest procés.

4.1 General

Aquesta secció tracta aspectes generals que afecten a tota l'aplicació, com pot ser el disseny responsiu de l'aplicació o el sistema d'idiomes.

4.1.1 Compatibilitat amb els dispositius

Un dels principals requisits no funcionals de l'aplicació és que s'ha de poder executar correctament en diferents dispositius de característiques diverses. El primer problema relacionat amb **poder executar l'aplicació** està resolt perquè s'ha dissenyat com una aplicació web. El següent problema consisteix en resoldre **com es visualitza** el contingut de l'aplicació.

El contingut que es mostra a l'aplicació s'ha d'adaptar correctament a les dimensions del dispositiu. Els resultats de les simulacions, els texts i els elements de control s'han de poder visualitzar independentment de les dimensions de la pantalla amb què s'utilitza l'aplicació. Per això, s'utilitzaran tècniques de disseny web *responsive* i els resultats de les simulacions es mostraran com a imatges vectorials o elements natius del navegador.

Disseny responsive

El disseny *responsive* és una tècnica de disseny que consisteix crear i desenvolupar aplicacions web que s'adapten a les característiques dels dispositius dels usuaris.

A l'actualitat existeixen eines que faciliten la creació de pàgines webs responsives com *Tailwind* o *Bootstrap*, que són llibreries que introdueixen estils CSS predefinits responsius. Per desenvolupar l'aplicació s'ha utilitzat *Bootstrap*.

(a) Disseny escriptori (b) Disseny mòbil

Figura 4.1: Estructura general del disseny.

La idea general per utilitzar *Bootstrap* consisteix en dividir la pàgina en diferents files, i aquestes files es poden dividir en columnes. En funció del dispositiu utilitzat les columnes ocuparan més o menys espai.

Ara bé, no tot el disseny responsiu s'ha pogut completar amb els estils de *Bootstrap*. En algunes situacions s'han hagut d'implementar estils propis responsius fent ús de `media-queries`, que són unes directives CSS que permeten definir estils de manera condicional en funció d'una condició.

La condició utilitzada és, generalment, l'amplada màxima del dispositiu. D'aquesta manera es creen diferents estils per cada una de les resolucions estàndards.
Per exemple, la barra de control de la simulació sempre mostra les mateixes opcions, però per la seva correcta visualització s'han de diferenciar els dispositius amb una amplada màxima més reduïda per disminuir la dimensió de les fonts i les icones.

Imatges vectorials

Les imatges tradicionals estan formades per un conjunt de píxels i cada un d'ells té un determinat color. El principal avantatge d'aquest format és que és un format fàcil de transmetre. Ara bé, l'inconvenient es que la resolució de la imatge està limitada i es perd qualitat si s'escala.

Les imatges vectorials es construeixen a partir d'una sèrie de figures geomètriques (rectangles, línies o camins arbitraris) i es renderitzen sense perdre qualitat perquè existeix un model matemàtic de la imatge que permet determinar píxels que s'han de visualitzar a una determinada regió.

Per tant, a l'hora de representar els resultats d'un simulador s'haurà d'utilitzar una imatge vectorial per garantir que es pot visualitzar correctament independentment de

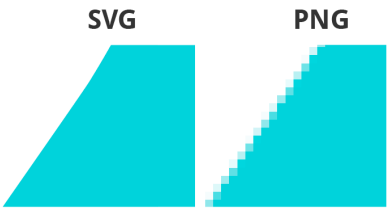


Figura 4.2: Comparació d'una imatge vectorial i una imatge tradicional. [1]

la resolució del dispositiu.

El DOM proporciona les eines necessàries per crear i manipular imatges vectorials directament des de JS però amb un nivell d'abstracció molt baix. Per aquest motiu s'ha considerat més adequat utilitzar una llibreria que faciliti la manipulació d'aquests elements.

La llibreria seleccionada per realitzar aquesta tasca és *SVG.js*, una llibreria lleugera per manipular i animar imatges vectorials de manera dinàmica. [22]
Les principals operacions que s'han utilitzat per realitzar els diferents gràfics dels simuladors són:

- El dibuixat i pintat de figures geomètriques com rectangles i línies. La combinació d'aquestes primitives permet representar gràfics complexos.
- La representació de texts i l'obtenció de les dimensions del text per situar-lo de manera adequada en funció de l'espai disponible.
- L'escalat és l'operació que permet canviar les dimensions de la imatge vectorial. Aquesta operació canvia les dimensions del model matemàtic de la imatge i per tant es renderitza sense perdre qualitat.

De manera general, a l'hora de representar un gràfic aquest té un espai disponible assignat i es renderitza amb aquest espai. Quan aquest espai disponible canvia, és a dir, l'usuari canvia les dimensions de la finestra de l'aplicació l'espai disponible pot haver canviat i s'ha de repintar el component del resultat del simulador amb el nou espai disponible.

Aquestes eines i llibreries han permès satisfer el requisit no funcional RNF-03 perquè permeten adaptar la representació del contingut de l'aplicació en funció de la resolució del dispositiu, aconseguint que l'aplicació es visualitzi correctament en les principals resolucions dels dispositius actuals.

4.1.2 Sistema d'idiomes

La selecció d'idiomes és una de les funcionalitats de l'aplicació (RF-07 i RNF-01) que permeten presentar tots els continguts en diferents idiomes. El sistema d'idiomes que s'ha d'implementar ha de:

- permetre a l'usuari seleccionar l'idioma en qualsevol moment.
- canviar els texts de manera dinàmica, sense necessitat d'haver de recarregar l'aplicació o perdre les dades de la simulació.
- ser fàcilment extensible, és a dir, afegir nous idiomes a l'aplicació sense haver de realitzar moltes modificacions.

A més, tenint en compte que tota l'aplicació ha de funcionar de manera offline s'ha de descartar la traducció per part de serveis externs, com *Google Translate* o equivalents.

El fet d'utilitzar una llibreria declarativa com React facilita la implementació d'un sistema d'idiomes dinàmic perquè l'idioma seleccionat pot ser una variable d'estat del component pare de l'aplicació i tots els components descendents la poden consultar.

Una primera aproximació per implementar aquest sistema és fer ús de condicionals per mostrar un text en un idioma o un altre, però aquest sistema suposaria haver de modificar cada un dels components quan s'introdueix un nou idioma a l'aplicació.

Un sistema més genèric es pot basar en l'ús d'unes funcions auxiliars que retornen el text adequat en funció de l'idioma seleccionat. Aquestes funcions rebran com a paràmetre un identificador únic que representarà un determinat text dins l'aplicació. Aquest identificador permetrà obtenir el text adequat de dins un fitxer de l'aplicació, de manera local.

La llibreria *i18next* és llibreria de JS que implementa els estàndards d'internacionalització o *i18*. Aquesta llibreria és una solució completa per localitzar l'aplicació en qualsevol tipus de dispositiu. [23]

Dins l'entorn de React existeix una llibreria que s'encarrega d'adaptar totes les funcionalitats de la llibreria original anomenada `react-i18next`. Aquesta s'encarrega d'integrar les funcions d'internacionalització dins l'entorn de React. [24]

Ús i estructura

Els components funcionals que utilitzen el sistema de traducció han d'importar el paquet de `react-i18n` per poder accedir a les seves funcionalitats. Per accedir a un text localitzat s'ha de fer ús d'una funció anomenada `t`, que rep un identificador únic.

Dins el codi JSX, en lloc d'introduir un text de manera literal s'ha d'invocar la funció de traducció de la següent manera:

```
<div>{t("identificador únic")}</div>
```

A nivell d'arxius, les traduccions s'emmagatzemen dins un fitxer en format JavaScript Object Notation (JSON) situats dins una carpeta anomenada `locales`. D'aquesta manera es garanteix que afegir un nou idioma consisteix en afegir un nou arxiu dins la carpeta.

Aquests arxius no s'han organitzat de qualsevol manera, sinó que estan dividits en les diferents seccions de l'aplicació:

- Una secció de texts comuns per tota l'aplicació, com poden ser els texts de botons comuns a tots els simuladors o de la barra de control del simulador.

- Una secció per cada un dels simuladors. Aquestes seccions inclouen tots els texts propis d'un simulador, dels seus resultats i del seu tutorial.
- La secció d'ajuda inclou els texts que apareixen a la pàgina d'ajuda, que es divideix en diferents subseccions, una per cada simulador.

En determinades situacions és necessari introduir un contingut variable dins el text traduït. L'estàndard `i18n` permet introduir variables dins els arxius de traduccions. Per exemple, un text que requereix variables es podria representar de la següent manera:

```
Has d'introduir un valor numèric entre {{min}} i {{max}}.
```

A la funció de traducció s'hauria d'especificar el valor de cada una de les variables mitjançant un paràmetre addicional.

Imatges localitzades

L'aplicació també té contingut que no és text però que necessita ser traduït, com les imatges que s'utilitzen a la secció d'ajuda. Aquestes imatges són explicacions sobre els conceptes dels simuladors o sobre el propi simulador, i per això no es poden representar d'una altra manera.

L'opció més simple és considerar que la imatge és una adreça a una imatge i per tant es pot tractar de la mateixa manera que un text qualsevol. D'aquesta manera, en funció de l'idioma seleccionat es carregarà una imatge o una altra.

Ara bé, la inclusió d'imatges dins les aplicacions en React es realitza de manera diferent a una aplicació tradicional, ja que si no es realitza correctament no queden incloses dins la versió compilada de l'aplicació.

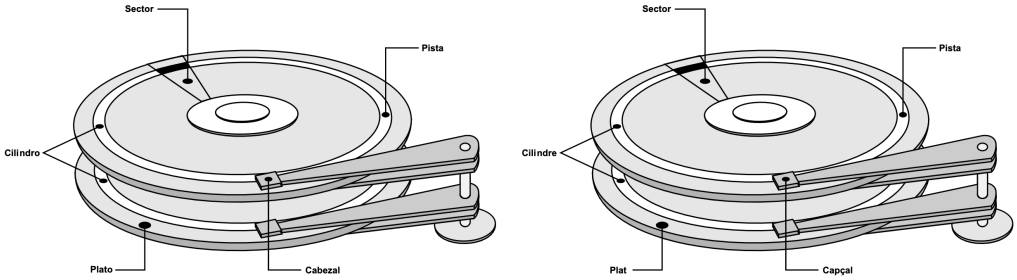
Per resoldre aquest problema s'ha dissenyat un component independent que mostra una imatge o una altra en funció de l'idioma de l'usuari. Aquest component té diferents paràmetres:

- El paràmetre obligatori `default` indica quina és la imatge que s'ha de mostrar quan l'usuari té un idioma seleccionat del qual no es disposava d'una imatge.
- Un paràmetre opcional que indica per cada idioma la imatge que s'ha de mostrar.

A més, fent ús de les propietats de l'herència de TS aquest component també pot rebre qualsevol paràmetre propi dels elements del tipus imatge d'HTML, com pot ser l'amplada i l'altura de la imatge.

A la figura 4.3 es pot veure com es presenta una imatge diferent a l'usuari en funció del seu idioma. Per utilitzar aquest sistema és necessari proporcionar una imatge amb cada un dels idiomes de l'aplicació.

El sistema d'idiomes comentat satisfà els requisits tant funcionals com no funcionals de l'aplicació perquè es pot canviar l'idioma de l'aplicació sense perdre la informació introduïda gràcies al paradigma declaratiu de React i l'usuari pot utilitzar l'aplicació en el llenguatge que triï.



(a) Imatge en un idioma (b) Imatge en un altre idioma

Figura 4.3: Exemple d'una imatge localitzada a la secció d'ajuda.

4.1.3 Components funcionals

A la secció 3.1.3 s'introdueixen dues maneres de crear components propis a les aplicacions de React i s'indica que es recomana utilitzar els components funcionals.

Constructors amb components funcionals

Els components funcionals eviten repetir codi en els diferents esdeveniments del cicle de vida dels components, però tenen el desavantatge que el codi del component és avaluat cada vegada que es renderitza el component. Les funcions, a diferència de les classes, no tenen un mètode constructor que només s'executi una única vegada quan s'instancia el component. Així, totes les crides a funcions i instruccions que es troben al cos de la funció s'executen, cosa que si no es gestiona correctament pot provocar problemes de rendiment.

Ara bé, React proporciona eines per aconseguir que determinades instruccions només s'executin una única vegada. Per exemple, una d'aquestes eines consisteix en definir un objecte que sempre serà el mateix al llarg de la vida del component, cosa que permet executar funcions una única vegada.

La simulació d'un constructor es pot aconseguir creant un objecte que emmagatzemi un valor booleà. A la primera execució del component, aquest objecte tendrà un valor `fals` i s'executarà el codi i s'assignarà el valor `vertader`. A les següents avaluacions del component funcional, el valor serà `vertader` i no s'executarà, aconseguint simular el comportament d'un constructor. Amb aquesta tècnica s'evitarà crear instàncies d'objectes que no seran utilitzades, fent l'aplicació més eficient tant en velocitat com en consum de memòria.

Comunicació amb els components

La manera més habitual de passar els paràmetres als components funcional és mitjançant els seus atributs o *props*. Aquests permeten definir com s'ha de visualitzar un determinat component. Aquests atributs poden tenir qualsevol tipus, des de valors literals a funcions que actuen com a *callbacks*.

Ara bé, en determinades situacions pot ser necessari haver de consultar un determinat valor del component o provocar algun canvi a l'estat del component sense modificar els atributs del component. En aquestes situacions és interessant fer ús de la

programació imperativa, que és un paradigma en què el programador especifica què s'ha de fer.

React permet exportar mètodes dels components funcionals fent ús de les referències i la creació de funcions imperatives. Aquestes funcions exportades poden ser invocades com si el component funcional fos un objecte dins la programació orientada a objectes.

Pel fet d'utilitzar TypeScript (TS) la definició de tipus dels components es complica quan es volen exportar funcions imperatives perquè és necessari definir múltiples tipus de dades:

- El tipus de dades associat als atributs del component, és a dir, quins i com han de ser els atributs del component.
- El tipus de dades corresponent a les funcions imperatives, que especifica la capçalera de les diferents funcions.

4.1.4 Desament i càrrega de simulacions

El desament i càrrega de simulacions és una funcionalitat que permet guardar i carregar la configuració del simulador a un fitxer de text. Aquesta opció és comuna a tots els simuladors i està implementada en el component de la barra de control del simulador.

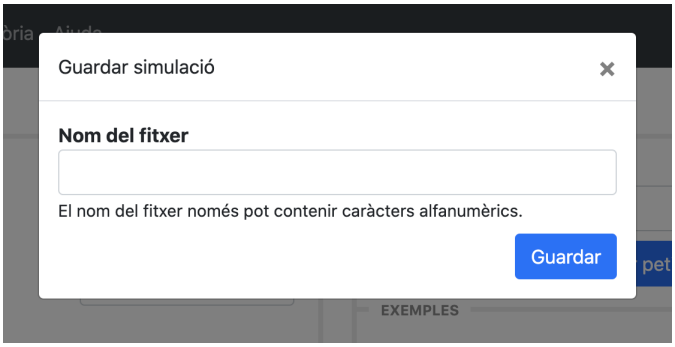


Figura 4.4: Modal per guardar la configuració d'un simulador.

La primera passa per implementar aquest sistema és definir quina és l'estructura del fitxer que s'ha de generar, és a dir, com es guardarà la informació. Pel fet d'estar fent feina amb JS és recomanable generar arxius en format JSON, tot i no ser la codificació més eficient.

Els arxius han de tenir informació de control que permeti identificar a quin simulador pertany el fitxer. D'aquesta manera es pot detectar si s'intenta carregar un fitxer incorrecte al simulador i conèixer quins valors s'esperen al fitxer.

Els simuladors tenen una funció per representar en format JSON la configuració del simulador. Aquesta funció és utilitzada per la barra de control, que té la lògica per guardar els arxius, a l'hora de guardar l'arxiu al dispositiu de l'usuari.

Per evitar problemes amb els permisos per escriure a la màquina de l'usuari, s'ha decidit que l'arxiu generat es guardi a la carpeta de descàrregues de l'usuari. Amb l'HTML modern es pot provocar la descàrrega d'un arxiu afegint l'atribut `download` a un element de tipus enllaç, que es guardarà dins la carpeta de descàrregues.

El procés de lectura de fitxers també es realitza a la barra de control, que és la que s'encarrega de mostrar el diàleg de selecció i realitzar la lectura del fitxer. Ara bé, el processament del fitxer està implementat a cada un dels simuladors, que comprova la validesa del fitxer de text i carrega la configuració.

4.1.5 Tutorial dels simuladors

Els tutorials dels simuladors són una funcionalitat que permeten introduir les diferents opcions de configuració i els resultats d'un simulador. L'objectiu d'aquests tutorials específics és que l'usuari pugui utilitzar el simulador després d'haver-lo completat.

Per implementar el tutorial s'ha decidit utilitzar la llibreria `react-tour` que permet mostrar a l'usuari diferents seccions de manera gràfica. La principal característica d'aquesta llibreria es que permet marcar una zona de la finestra de l'aplicació, aconseguint centrar l'atenció de l'usuari en aquell punt, com es pot veure a la figura 4.5.

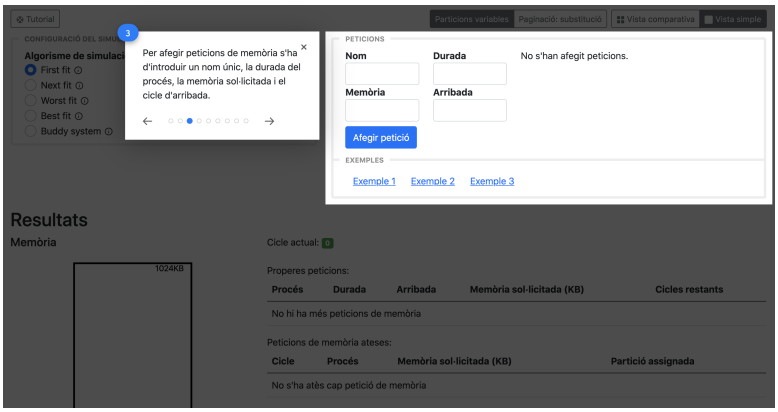


Figura 4.5: Visualització d'una part del tutorial d'un simulador.

Per a la correcta integració de la llibreria amb l'aplicació és necessari poder identificar els elements del DOM de manera única, mitjançant el seu identificador o qualsevol altre atribut. D'aquesta manera, la llibreria podrà conèixer la posició de l'element dins la pantalla i marcar-lo de manera adequada.

L'opció més senzilla, per no haver de modificar els identificadors dels elements existents, consisteix en fer ús dels atributs `data-*`. Aquests atributs permeten al programador introduir informació als elements de manera simple sense haver de fer ús d'altres atributs destinats a altres qüestions. L'atribut utilitzat és `data-tut` i el seu valor representa el seu identificador, dins el context del tutorial.

A continuació, només s'ha de definir la llista d'elements que s'han de marcar i el text que ha d'acompanyar a l'element. Els texts del tutorial també han d'aparèixer en

l'idioma seleccionat, que es pot aconseguir fent ús de les eines definides a la secció 4.1.2.

Ara bé, la llibreria de manera predeterminada només marca les diferents zones de l'aplicació i permet mostrar un text explicatiu. El simulador està buit inicialment, sense cap tipus de resultat ni configuració addicional, per tant, no hi ha contingut per explicar. És necessari adaptar la llibreria per permetre realitzar determinades accions a cada una de les passes.

Per poder executar funcions quan l'usuari canvia d'una passa a l'altra, s'han substituït les funcions predeterminades de la llibreria per unes pròpies. El fet de substituir les funcions pròpies de la llibreria obliga gestionar totalment el canvi de passes, cosa que proporciona molta flexibilitat.

D'aquesta manera s'ha definit un sistema basat en esdeveniments en què és possible executar un codi personalitzat en instants crítics del canvi de passa. Els esdeveniments definits són:

- L'esdeveniment `beforeReach`, que s'executa abans de produir-se el canvi de passa. Aquest esdeveniment es pot utilitzar quan es vol canviar l'estat del simulador i aquests canvis han d'aparèixer visibles a la següent passa.

- L'esdeveniment `reach` s'executa quan l'usuari assoleix una passa. També es pot utilitzar per actualitzar l'estat del simulador. La principal diferència entre aquest esdeveniment i l'anterior és el moment en què s'invoca. Aquest s'executa després d'haver realitzat el canvi d'estat de la passa actual, i per tant, els canvis que es produeixin es poden no renderitzar.

- L'esdeveniment `finish` es produeix quan l'usuari abandona una determinada passa. Aquest esdeveniment s'executarà abans que `beforeReach`, `reach` i el canvi d'estat de la passa. Es pot utilitzar per desfer els canvis que ha realitzat una passa al seu inici.

El sistema d'esdeveniments és suficientment flexible per poder executar canvis sobre l'estat del simulador sempre i que les funcions de manipulació siguin accessibles des del punt en què es defineix la passa. En funció de la jerarquia de components aquestes funcions estaran disponibles o no.

En cas que no es trobin disponibles de manera directa, es pot fer ús de la programació imperativa per definir accions molt concretes sobre el simulador. No seria adequat fer ús del paradigma declaratiu en aquest cas perquè l'estat del simulador és gestionat de manera interna i no pel component que conté el tutorial, i el que es vol realitzar són canvis d'estats sobre el simulador.

Les figures 4.5 i 4.6 mostren el canvi que s'ha produït de manera automàtica al simulador entre una passa i l'altra. A la figura 4.6 es pot veure com s'han introduït una sèrie de peticions al simulador i s'ha avançat la simulació per poder observar els resultats.

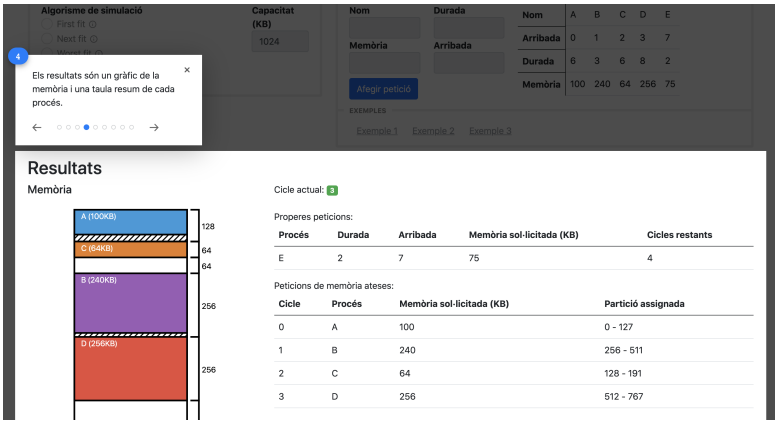


Figura 4.6: Visualització del tutorial després d'haver canviat de passa.

Renderitzat de LaTeX

Algunes seccions del manual d'usuari, dels modals d'ajuda dels algorismes o dins els resultats dels simulador ha estat necessari renderitzar una imatge a partir de d'una codi LaTeX.

La renderització de codi LaTeX s'ha implementat a través d'un component independent que rep una cadena de text seguint la sintaxi de LaTeX. Aquest component s'encarrega de crear els elements HTML pel correcte renderitzat fent ús de la llibreria KaTeX. [25]

Aquest component funciona correctament amb el sistema d'idiomes, ja que fa ús de cadenes de text, que es poden representar dins els arxius de les traduccions. Tampoc és necessari una connexió a Internet per fer ús d'aquesta llibreria perquè està empaquetada dins la pròpia aplicació. D'aquesta manera es poden representar expressions matemàtiques com la de la figura 4.7.

$$\text{Ràtio} = \frac{\text{Temps esperant} + T_s}{T_s}$$

Figura 4.7: Expressió matemàtica amb LaTeX dins la secció d'ajuda.

4.1.6 Implementació dels algorismes

Els simuladors realitzen una determinada funció com pot ser atendre peticions d'E/S o determinar quin procés s'ha d'executar en cada instant. Tots ells poden utilitzar diferents algorismes per decidir quina és la petició que han de seleccionar en cada moment. Una vegada la petició s'ha seleccionat, el tractament d'aquesta petició és independent de l'algorisme.

L'objectiu és dissenyar de la manera més genèrica possible aquest procés de selecció de peticions de manera que afegir un nou algorisme no impliqui realitzar canvis en

el tractament de les diferents peticions.

Un plantejament genèric consisteix en definir un conjunt de peticions

$$R = \{r_1, r_2, \ldots, r_n\}$$

i una funció $f(x)$ que selecciona un element sobre aquest conjunt de peticions. La funció de selecció *només* s'encarrega de determinar el següent element que s'ha de processar. Aquesta funció té un tipus de retorn, que dependrà del tipus de simulador, i serà el mateix per cada un dels algorismes.

La implementació d'aquest sistema a TS es pot realitzar de manera simple, aprofitant que els simuladors tenen una propietat que indica quin és l'algorisme seleccionat. D'aquesta manera es pot definir un objecte en què les propietats seran els diferents algorismes, i els valors, les diferents funcions.

Per exemple, si un simulador té els algorismes `FIFO`, `SCAN` i `LOOK` es pot definir el següent objecte:

```
{
    FIFO: this.FIFO.bind(this),
    SCAN: this.SCAN.bind(this),
    LOOK: this.LOOK.bind(this)
}
```

on les funcions `FIFO`, `SCAN` i `LOOK` són funcions pròpies de la classe del simulador. Amb aquesta estructura de dades es pot executar la funció de selecció sense utilitzar cap tipus de condicional a partir de l'algorisme seleccionat. A l'actualització de l'estat intern es pot invocar com `algoritmes[algoritme]()` i aquesta expressió retorna la següent petició que s'ha de processar.

4.2 Simulador d'E/S

El simulador d'entrada i sortida permet simular els algorismes utilitzats pels sistemes operatius per atendre les sol·licituds. [26] En aquest simulador les peticions es representen mitjançant un valor numèric que indica la pista que es vol consultar. Les funcions dels diferents algorismes sempre retornen un valor numèric que és l'índex de la petició sobre la llista de peticions pendents per atendre.

4.2.1 Disc

El disc és un dels resultats del simulador d'entrada i sortida que mostra una simulació del moviment del capçal durant l'execució de la simulació. Aquest gràfic està realitzat com una imatge vectorial fent ús de la llibreria *SVG.js*, com s'ha indicat a la secció 4.1.1.

El component es pot configurar fent ús de diferents atributs que especifiquen com s'ha de representar el disc:

- El número de pistes que conté el disc. Les diferents pistes es representen com a circumferències concèntriques.

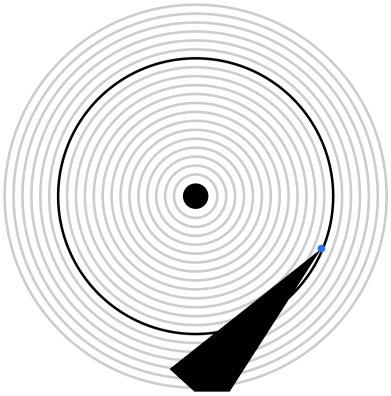


Figura 4.8: Gràfic del disc.

- La pista actual, que és la pista en què es trobarà el capçal a l'instant inicial.
- La pista objectiu és la pista que ocuparà el capçal una vegada finalitzi l'animació del moviment. Aquest paràmetre és opcional, i quan no s'especifica simbolitza que no s'ha de realitzar cap animació.
- La duració de l'animació. Aquest paràmetre ha de canviar quan la velocitat de la simulació canvia, ja que s'ha de garantir que l'animació s'ha completat quan s'executa la següent passa.

Càlcul de la posició del capçal

Tot i que l'objectiu del gràfic és mostrar que el disc dur està format per diferents components que es mouen no és adequat mostrar el capçal en una pista aleatòria o que no correspon amb les dades introduïdes.

L'opció més fàcil per resoldre aquest problema és calcular l'orientació que hauria de tenir el capçal si vol estar situat sobre una determinada pista. Per realitzar aquest càlcul és necessari realitzar una abstracció sobre el problema. L'opció per resoldre el problema ha estat obtenir un model matemàtic del problema i resoldre'l de manera analítica.

A la figura 4.9 es pot veure com s'han representat els diferents components del disc dur:

- Les circumferències de color vermell representen les diferents pistes del disc dur.
- La circumferència blava representa el capçal i el seu possible moviment.

D'aquesta manera calcular l'orientació del capçal per estar situat sobre una determinada pista consisteix en calcular la intersecció entre dues circumferències. [27] La intersecció entre dues circumferències donarà, com a molt, dos punts sobre el pla i se'n seleccionarà un. Amb el punt de la intersecció es podrà calcular l'orientació del capçal i representar-lo correctament.

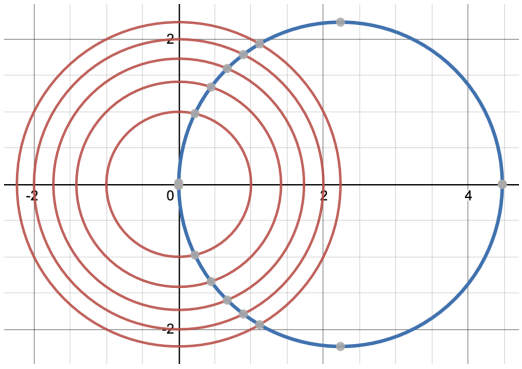


Figura 4.9: Model matemàtic d'un disc dur.

Animació amb SVG.js i React

L'animació del capçal és un moviment suau entre la pista d'inici i la pista objectiu amb la durada establerta per paràmetre mitjançant els atributs del component.

Una primera aproximació pot ser guardar l'orientació dins una variable d'estat i actualitzar-la periòdicament fins assolir l'orientació objectiu. Ara bé, l'actualització de variables d'estat provoca la renderització de tot el component i no interessa pintar de nou les pistes del disc, ja que no han canviat.

Una altra solució més eficient consisteix en guardar l'orientació dins una variable d'estat que no provoqui la renderització i actualitzar l'orientació del capçal però sense repintar tots els components del disc.

El fet de no pintar tot el disc provocarà una millora de rendiment quan la quantitat de pistes és elevada, ja que les pistes es pintaran una única vegada quan s'inicialitzi el component. El capçal s'actualitzarà mitjançant la funció que permet canviar l'orientació d'un polígon de *SVG.js*.

L'animació s'ha de realitzar amb una actualització de l'orientació de manera periòdica. L'execució d'un codi cada un cert interval es pot aconseguir mitjançant un interval. Tot i que per utilitzar un interval dins React és necessari tenir en compte que el component funcional s'avalua múltiples vegades i l'interval es defineix tantes vegades com és avaluat.

Per evitar aquests inconvenients, s'ha creat una funció que facilita la gestió dels intervals a React. Aquesta funció rep com a paràmetres la funció que executarà a cada un cert període de temps, el període de temps i un valor per indicar si l'interval està en execució o no. La combinació d'aquesta funció amb la lògica de l'actualització de l'orientació permetrà crear una animació del moviment del capçal.

4.2.2 Gràfic de peticions

El gràfic de peticions ateses permet veure el moviment que ha realitzat el capçal per atendre les diferents peticions. El fet d'utilitzar una llibreria de baix nivell com *SVG.js*

obliga a implementar el funcionament d'un gràfic.

Aquest gràfic no s'obté de manera directa amb una funció de la llibreria, és el resultat de combinar instruccions per representar línies, polígons i texts. En aquest cas s'ha hagut de gestionar correctament com representar els diferents punts de valors, és a dir, que l'escala s'ajusti correctament a l'espai disponible i que tots els valors del gràfic es puguin llegir.

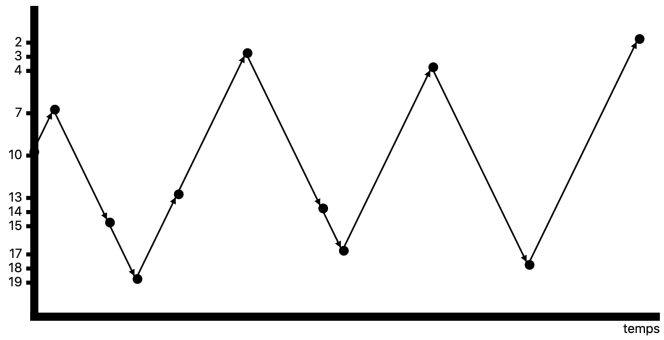


Figura 4.10: Gràfic de les peticions ateses.

La figura 4.10 és un exemple del gràfic que es mostra al resultat del simulador. Com es pot veure l'eix horitzontal representa el temps, però no inclou l'escala, ja que el que realment importa és l'ordre en què s'atenen les peticions i veure que el temps és major a major distància.

4.3 Simulador de CPU

El simulador de processador permet observar com funciona la planificació de processador amb els diferents algorismes a través del seguiment cicle a cicle dels processos al processador. [26]

A diferència dels altres simuladors, els algorismes de planificació seleccionen els processos de diferents coes, com la coa de processos preparats o la coa de processos bloquejats. A més, alguns algorismes funcionen amb coes dinàmiques, és a dir, que és necessari disposar d'una estructura que permeti gestionar diferents coes i que aquestes es puguin crear en temps d'execució.

D'aquesta manera, la funció de selecció, que indica quin és el següent procés en executar-se, ha d'indicar el procés que s'ha d'executar i de quina coa el selecciona. Les funcions de selecció poden retornar un valor especial per indicar que no hi ha cap procés que es pugui executar en el següent cicle, o bé perquè tots els processos estan bloquejats o bé perquè no hi ha processos a la coa de preparats.

4.3.1 Vista comparativa

La vista comparativa del simulador de CPU permet comparar diferents algorismes al mateix temps, igual que als altres simuladors. Ara bé, alguns algorismes d'aquest si-

mulador tenen paràmetres que canvien el seu comportament en funció d'un paràmetre.

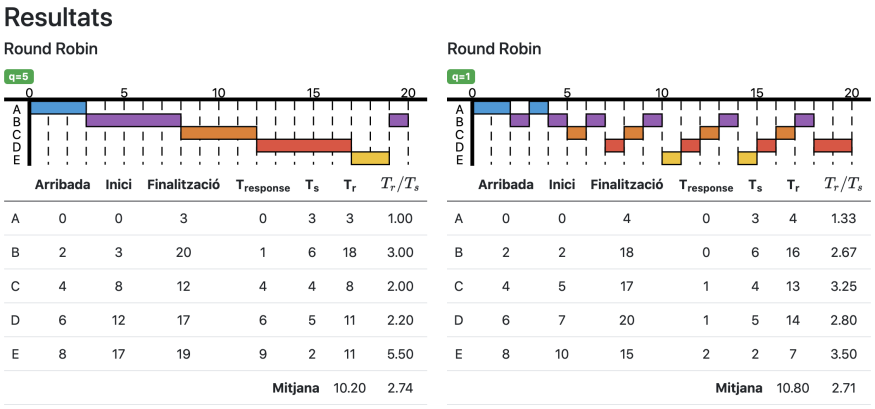
Si el funcionament de la vista comparativa d'aquest simulador és igual al dels altres simuladors, l'usuari només podrà seleccionar un algorisme una única vegada. Aquest fet no permetrà comparar un mateix algorisme amb diferents paràmetres de configuració.

Per tant, és necessari implementar un sistema que permeti seleccionar un mateix algorisme múltiples vegades i configurar-lo de manera diferent. El sistema que s'ha dissenyat permet a l'usuari comparar diferents configuracions dels algorismes.

Aquest sistema suposa dissenyar una estructura de dades que permeti tenir múltiples instàncies del simulador per un mateix algorisme. Una possible estructura que resol aquest problema és un objecte en què les propietats són els diferents algorismes i el seu valor és una llista de simuladors. Els elements de la llista són cada un dels simuladors amb la seva configuració. Aquest sistema també funciona per aquells algorismes que no admeten configuració, la llista només tindrà un element que serà l'única configuració del simulador.

Figura 4.11: Menú per configurar un algorisme.

A la figura 4.11 es pot veure com d'un mateix algorisme s'han afegit diferents configuracions i es mostra el menú per afegir una nova configuració.



	Arribada	Inici	Finalització	$T_{response}$	T_s	T_r	T_r/T_s
A	0	0	3	0	3	3	1.00
B	2	3	20	1	6	18	3.00
C	4	8	12	4	4	8	2.00
D	6	12	17	6	5	11	2.20
E	8	17	19	9	2	11	5.50
				Mitjana	10.20		2.74

	Arribada	Inici	Finalització	$T_{response}$	T_s	T_r	T_r/T_s
A	0	0	4	0	3	4	1.33
B	2	2	18	0	6	16	2.67
C	4	5	17	1	4	13	3.25
D	6	7	20	1	5	14	2.80
E	8	10	15	2	2	7	3.50
				Mitjana	10.80		2.71

Figura 4.12: Visualització dels resultats amb múltiples configuracions.

A la figura 4.12 es mostra com es visualitzen els resultats de la vista comparativa. En particular, com es pot veure que s'està comparant el mateix algorisme amb diferents

configuracions.

4.3.2 Components principals

La pàgina del simulador s'ha dividit en diferents components, seguint l'estil de programació de React, i cada un d'ells realitza una tasca molt concreta.

Llista de processos

Els components relacionats amb la llista de processos són el formulari per afegir els processos i el llistat de processos que s'han introduït.

Figura 4.13: Formulari per introduir un procés al simulador.

El formulari de la figura 4.13 permet a l'usuari introduir peticions al simulador. Per facilitar la creació d'aquests processos, i permetre a l'usuari centrar-se en el que realment importa, el camp "Nom" té un valor automàtic amb una lletra de l'alfabet, tot i que es permet la introducció d'un nom qualsevol.

A més, aquest component es pot dividir en els propis camps de text i la selecció de la distribució dels cicles de processador. El segon component és reutilitzat en altres seccions del mateix simulador perquè té atributs que faciliten la seva reutilització, com indicar si és editable o no i quin és el cicle actual.

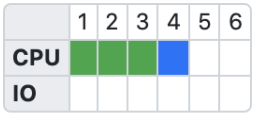


Figura 4.14: Component de distribució dels cicles amb atributs diferents.

Com es pot veure a la figura 4.14 les diferents caselles no són seleccionables i el cicle actual està marcat d'un color diferent.

L'altre component relacionat amb el llistat de processos és la pròpia llista de processos que mostra els processos que s'han introduït i la seva informació. Aquest component també fa ús del component de la distribució de cicles.

A més la llista també permet eliminar els processos mitjançant un botó que apareix quan els atributs ho permeten. Ara bé, la lògica d'eliminar els processos no es realitza a aquest component perquè el component només s'encarrega de representar la llista de processos passada com a atribut del component, és a dir, els processos estan emmagatzemats al component pare.

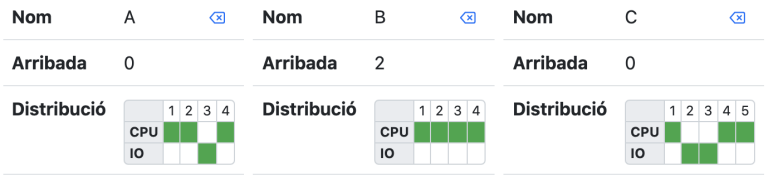


Figura 4.15: Llista dels processos introduïts.

L'acció d'eliminar es notifica al component pare mitjançant un sistema d'esdeveniments: quan l'usuari selecciona un procés per eliminar-lo es notifica al component pare mitjançant la funció especificada als atributs del component.

Diagrama temporal

El diagrama temporal mostra l'estat de cada procés a cada cicle de processador. D'aquesta manera es pot tenir una visió general del funcionament de l'algorisme una vegada s'ha completat la simulació.

Aquest gràfic és una imatge vectorial formada a partir de figures geomètriques simples. Aquest fet permet representar correctament el gràfic independentment de l'espai disponible, ja que sempre ocupa tot l'espai horitzontal disponible i l'altura de cada procés és fixada.

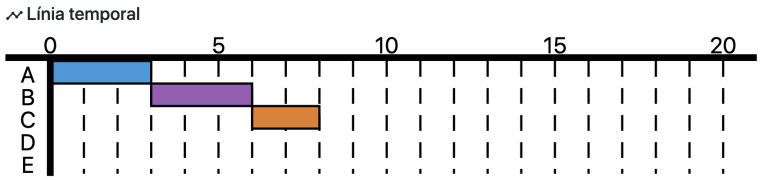


Figura 4.16: Diagrama temporal de l'execució dels processos.

Una vegada resolt el problema de com representar el gràfic és necessari determinar l'escala horitzontal del gràfic, que és el temps. Per evitar canvis a l'escala és necessari conèixer el el màxim número de cicles que s'han de representar abans de començar la simulació. Aquest valor seria la suma de la durada de cada un dels processos si els processos no poguessin estar bloquejats i s'executassin un darrere l'altre.

Aquestes condicions no se satisfan, per tant és necessari calcular la quantitat de cicles que tarda en executar-se una determinada configuració abans d'executar-la. Per realitzar aquest càlcul és suficient en crear una instància del simulador i configurar-lo de la mateixa manera. A continuació es compten quantes passes s'executen fins que acaba la simulació, aquest valor serà la quantitat de cicles.

A la figura 4.16 es pot veure com la simulació no està completada i es coneix la durada de la simulació per poder representar correctament el gràfic.

Taula de resultats

Al simulador apareixen diferents taules que representen diferents resultats: la taula de cada una de les coes de processos i la taula resum de la planificació.

Arribada de processos

Procés	Cicles restants
C	1
D	3
E	5

Preparats

Procés	Cicles esperant
B	1

Figura 4.17: Taules de les coes de processos.

Les taules de coes permeten veure l'estat actual d'una determinada coa, indicant quins processos es troben a la coa i quant de temps han estat dins la coa. La creació d'un component per aquesta taula té sentit perquè és una taula que pot aparèixer múltiples vegades i la informació mostrada es pot passar mitjançant els atributs.

A la figura 4.17 es pot veure com es visualitzen les diferents coes de processos. En el cas d'algorismes amb coes dinàmica es mostra la seva prioritat.

Resum planificació

	Arribada	Inici	Finalització	Temps de resposta ($T_{response}$)	Temps de servei (T_s)	Temps de retorn (T_r)	T_r/T_s
A	0	0	3	0	3	3	1.00
B	2	3	9	1	6	7	1.17
C	4	9	13	5	4	9	2.25
D	6	13	18	7	5	12	2.40
E	8	-	-	-	2	-	-
					Mitjana	-	-

Figura 4.18: Taula resum de la planificació.

L'altra taula que es mostra és la taula resum de la planificació, que indica per cada procés les mètriques més rellevants sobre la seva execució.

A la figura 4.18 es pot veure com es representa aquesta taula. Només es mostra informació dels processos que s'han executat completament.

4.4 Simulador de Memòria

4.4.1 Assignació de memòria variable

L'assignació de memòria variable és un simulador que permet observar l'espai que ocuparan els processos a la memòria fent ús d'algorismes d'assignació de memòria de particions variables. En aquest simulador els algorismes fan ús dels blocs de memòria per determinar l'espai que ocuparan els processos.

Particularitats d'un algorisme

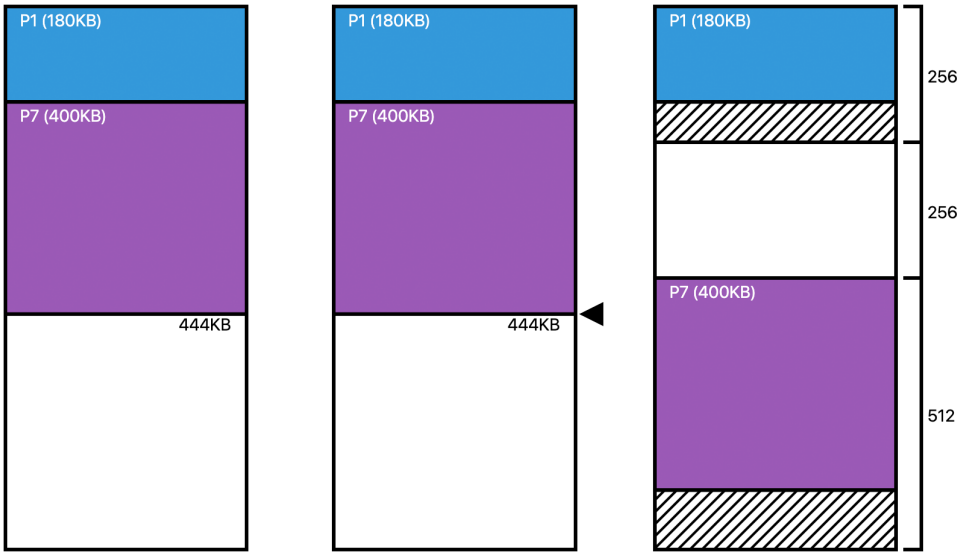
Els algorismes més simples de l'assignació de memòria, a nivell d'implementació, funcionen de la mateixa manera: realitzen una cerca sobre la memòria i seleccionen un bloc de memòria. La diferència entre els diferents algorismes serà quin bloc seleccionen o des d'on comencen a realitzar la cerca.

L'algorisme *buddy system* funciona de manera diferent perquè pot modificar un bloc disponible per adaptar-lo a l'espai necessari pel procés de manera que la mida del bloc sigui de la forma 2^i. Per tant, aquest algorisme pot generar fragmentació interna i crear nous blocs de memòria disponible. A l'hora d'alliberar els blocs de memòria s'ha de tenir en compte que només es poden fusionar aquells que s'han obtengut a partir del mateix bloc de memòria.

És necessari dissenyar una estructura de dades que permeti gestionar aquests blocs. L'estructura natural per aquest problema és un arbre binari, ja que cada bloc es divideix en dos subblocs. Ara bé, si es té en compte que tots els nodes tenen un valor d'una potència de 2 i que tots els nodes tenen dos fills o no en tenen, es pot utilitzar una estructura més simple, com una llista. En qualsevol dels dos casos, s'haurà de realitzar una gestió correcta de l'estructura de dades.

Gràfic de memòria

El gràfic que representa l'ocupació de la memòria és el principal resultat d'aquest simulador, ja que permet observar l'espai que ocupa cada un dels processos i els espais disponibles pels nous processos.



(a) Gràfic simple (b) Gràfic simple amb punter (c) Gràfic amb grups de blocs

Figura 4.19: Configuracions del gràfic de memòria.

Aquest resultat s'ha dissenyat com un component independent amb múltiples

opcions de configuració per poder-lo reutilitzar amb els diferents tipus d'algorismes, com es pot veure a la figura 4.19. L'altura d'aquest component està fixada i l'escala es determina a partir de la capacitat de la memòria.

- L'opció més bàsica consisteix en representar els diferents blocs de memòria per aquells algorismes que no fan ús de punters, cada procés té el seu bloc amb un color identificatiu mostrant les característiques del bloc.
- Alguns algorismes requereixen mostrar una informació addicional sobre el gràfic tradicional, com pot ser un punter. El component té un atribut per introduir una fletxa sobre la memòria.
- Alguns algorismes generen blocs de memòria que no es poden utilitzar, la fragmentació interna. Aquestes regions es poden representar amb un format diferent per indicar que no estan disponibles. A més, és possible agrupar els diferents blocs en grups, que és útil per un dels algorismes.

Taula de resultats

L'altre resultat del simulador és un conjunt de taules que mostren les pròximes peticions i les peticions que s'han atès. A la figura 4.20 es poden veure les dues taules que mostren les properes peticions i les peticions que s'han atès.

Properes peticions:

Procés	Durada	Arribada	Memòria sol·licitada (KB)	Cicles restants
P8	1	0	150	0
P4	permanent	2	120	2
P5	permanent	2	200	2
P6	permanent	2	80	2

Peticions de memòria ateses:

Cicle	Procés	Memòria sol·licitada (KB)	Partició assignada
0	P1	180	0 - 255
0	P7	400	512 - 1023
0	P2	100	256 - 383

Figura 4.20: Taules del simulador de particions variables.

A la vista comparativa dels algorismes no és possible mostrar tanta informació i només es mostra una taula amb els processos carregats a memòria i indicant el seu cicle de finalització per facilitar el seguiment dels algorismes.

4.4.2 Reemplaçament de pàgines

El simulador de reemplaçament de pàgines mostra com els processos es poden dividir en pàgines i com aquestes es carregaran i s'alliberaran de la memòria principal. Aquest

simulador utilitza els conceptes de processos i pàgines però les peticions es realitzen sobre la pàgina d'un procés. Els algorismes indiquen quina pàgina de la memòria principal s'ha d'alliberar, per tant, les funcions de selecció retornen aquest valor.

Gràfic de memòria i taula de pàgines

El gràfic de la memòria mostra una visió general de l'ocupació dels marcs a la memòria. Aquest gràfic és diferent que el comentat a la secció 4.4.1 perquè aquest divideix la memòria en marcs de mida fixa.

El gràfic, que es pot veure a la figura 4.21, és un component independent que rep tota la informació a través dels atributs, com el noms del processos, la quantitat de marcs totals i l'estat de cada un d'ells.

Figura 4.21: Gràfic de la memòria al simulador de paginació.

L'altre resultat del simulador és la taula de pàgines, que indica per cada pàgina si està a memòria principal i l'estat dels seus bits, si l'algorisme ho requereix. Aquesta taula també és un component que rep la informació mitjançant els atributs.

Pàgina	Marc
0	-
1	-
2	1
3	-
4	0
5	2

Figura 4.22: Taula de pàgines al simulador de paginació.

Seguiment passa a passa d'un algorisme

La majoria d'algorismes realitzen una única cerca per determinar quina és la pàgina que han de seleccionar, com pot ser seleccionar la que duu més temps a la memòria o la que ha estat inutilitzada durant més temps. Tots aquests algorismes realitzen una cerca i seleccionen la que maximitza o minimitza un determinat criteri.

Existeixen alguns algorismes, com el del rellotge o el NRU, que durant el procés de selecció de la pàgina modifiquen l'estat de la taula de pàgines modificant els bits d'aquesta. Si aquests algorismes s'implementen com els altres, l'usuari no podrà veure les passes que ha realitzat l'algorisme per arribar a l'estat final.

A la figura 4.23 es pot que la taula de processos ha canviat però és difícil veure exactament quines passes s'han realitzat per passar d'un estat a l'altre.

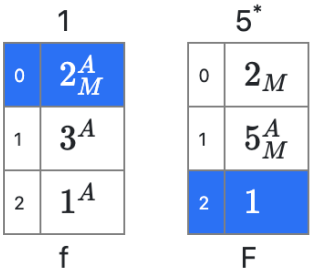


Figura 4.23: Taula de processos abans i després de processar una sol·licitud

Per resoldre aquest problema el que s'ha fet és permetre a l'usuari seguir totes les passes internes que realitza l'algorisme. D'aquesta manera es garanteix que a cada passa que realitza l'usuari, el simulador només realitza un únic canvi com avançar el punter o canviar el valor d'algun bit.

El resultat final serà el mateix que el de la figura 4.23 només que per arribar a l'estat final s'hauran de realitzar més passes del simulador.

CAPÍTOL 5

CONCLUSIONS

En aquest capítol s'avalua si s'ha satisfet la necessitat plantejada i es comenten possibles ampliacions de la feina realitzada.

5.1 Avaluació del resultat

Els simuladors de sistemes operatius per a la docència existents no satisfan completament les necessitats dels docents i alumnes de les assignatures de sistemes operatius, ja que aquestes aplicacions no estan completes i no estan pensades per a l'usuari final. A més, algunes d'elles estan desenvolupades amb tecnologies que han quedat obsoletes degut a la falta de manteniment, convertint-les en inexecutables a l'actualitat o no inclouen instruccions d'ús.

Per donar resposta a aquesta necessitat s'ha desenvolupat una nova aplicació que incorpora els principals temes que es tracten a les assignatures de sistemes operatius. La nova aplicació es caracteritza per ser una aplicació web multiplataforma, tant per mòbils com ordinadors, amb un disseny responsiu que s'adapta als diferents dispositius.

El fet de ser una aplicació web no implica que només pugui ser utilitzada amb una connexió a Internet activa. L'aplicació està dissenyada per funcionar sense connexió sempre i que s'hagi descarregat prèviament.

A diferència de les solucions existents, aquesta combina tots els simuladors en una únic portal, creant una interfície comuna de configuració i funcionament semblant entre els diferents simuladors.

Aquests simuladors representen els principals algorismes de planificació de processador, planificació d'entrada/sortida i de gestió de memòria dels quals es pot visualitzar un petit resum sobre el seu funcionament per ajudar a l'usuari a entendre el seu funcionament. Així mateix els controls dels simuladors permeten realitzar una execució

passa a passa per permetre a l'usuari seguir l'execució de l'algorisme. També existeixen controls per tornar a l'estat anterior tantes vegades com sigui necessari així com per realitzar una execució automàtica.

A més a més, tots els simuladors tenen exemples predefinits extrets de les principals bibliografies de sistemes operatius, i un tutorial que explica passa a passa les diferents seccions del simulador i com utilitzar-les.

5.2 Proves amb usuaris reals

El portal de simuladors desenvolupat s'ha pogut provar en un entorn real durant el segon semestre del curs 20/21 a l'assignatura de Sistemes Operatius II. Els alumnes d'aquesta assignatura han pogut utilitzar els simuladors d'entrada/sortida i de gestió de memòria per aprendre i assimilar els conceptes de l'assignatura.

A més, aquesta prova s'ha realitzat en una situació extraordinària de docència presencial adaptada en què els alumnes han hagut de seguir les classes teòriques a distància. El simulador ha facilitat tant al docent l'explicació dels algorismes com als alumnes el seu aprenentatge autònom.

Per a la realització d'aquesta prova s'ha proporcionat una versió en línia de l'aplicació per facilitar el seu ús a través del següent enllaç https://so.jaumealoy.dev. També s'ha proporcionat el repositori de l'aplicació a GitHub.

Durant aquesta prova es va demanar als alumnes, de manera voluntària, expressar els seus comentaris i idees sobre els simuladors. Tot i que la quantitat de comentaris rebuts és petita per generalitzar, es pot observar que l'aplicació sí té utilitat i que ajuda a l'aprenentatge dels conceptes.

5.3 Possibles ampliacions

L'aplicació inclou els principals simuladors que s'expliquen a les assignatures de sistemes operatius. Tot i això hi ha alguns temes que no s'han tractat i que es podrien afegir com a nous simuladors a l'aplicació. Alguns d'aquests temes són els simuladors de sistemes de fitxers.

Quant a la internacionalització, l'aplicació té un sistema d'idiomes que facilita la introducció de nous idiomes. Actualment només disposa de dos idiomes però es podria afegir suport per nous idiomes amb l'objectiu d'arribar a un públic més ampli.

Un altre tema que no s'ha tractat durant el desenvolupament de l'aplicació és l'automatització de les proves unitàries i d'integració. Així com s'han dissenyat els simuladors i amb les eines existents es podrien haver-ne creat per comprovar el correcte funcionament de l'aplicació de manera automatitzada.

Finalment, es vol presentar el portal de simuladors desenvolupat a les *Jornadas de Enseñanza Universitaria de la Informática 2022* (JENUI 2022), que és un fòrum

sobre l'ensenyament universitari de la informàtica que té com objectiu l'intercanvi de coneixement entre professors per millorar la docència de la informàtica. [28]

Aquest fòrum és una oportunitat per donar a conèixer el portal de simuladors desenvolupat i que sigui utilitzat per altres docents d'assignatures de sistemes operatius d'altres universitats.

BIBLIOGRAFIA

[1] J. Trythall, "Comparació entre un svg i una imatge no vectorial," Aug. 2014. [Online]. Available: https://lincolnloop.com/blog/look-at-svg/ (document), 4.2

[2] S. Bouloutian, *Disk Scheduling Simulation*, 2015. [Online]. Available: https://github.com/ScottBouloutian/Disk-Scheduling-Simulation 1.2.1

[3] B. Kirotich, *Disk Scheduling Simulation*, 2013. [Online]. Available: https://github.com/kirotich/disk-scheduling 1.2.1

[4] S. Reza, *Disk Scheduling*, 2013. [Online]. Available: https://github.com/safiyat/DiskScheduling 1.2.1

[5] R. Arora, *Disk scheduling and memory management*, 2018. [Online]. Available: https://github.com/lennon2298/disk-scheduling-mem-mgmt 1.2.1

[6] M. Sijben, *Disk Scheduling Simulation*, 2017. [Online]. Available: https://github.com/mpsijben/DiskSchedulingSimulation 1.2.1

[7] A. Macia, *OS Sim*, 2015. [Online]. Available: https://sourceforge.net/projects/oscsimulator/ 1.2.1, 1.2.2, 1.2.3

[8] M. Kim, *CPU Scheduling Algorithm Simulator*, 2019. [Online]. Available: https://github.com/alstn2468/CPU_Scheduling_Simulator 1.2.2

[9] J. Marcel, *Scheduler*, 2009. [Online]. Available: https://github.com/jasmarc/scheduler 1.2.2

[10] R. Wickramasinghe, *Scheduling Simulator*, 2014. [Online]. Available: https://github.com/Ruchiranga/SchedulingSimulator 1.2.2

[11] A. Yayha and H. Hijazi, *Operating System Scheduling*, 2019. [Online]. Available: https://github.com/AhmadYahya97/OperatingSystemScheduling 1.2.2

[12] *SiGeM*, Padua University, 2014. [Online]. Available: https://sourceforge.net/projects/sigem/ 1.2.3

[13] J. Goodson, *Memory Management Simulator*, 2017. [Online]. Available: https://github.com/jamiegdsn/memory-management-simulator 1.2.3

[14] D. Szabo, *Memory Management Simulation*, 2012. [Online]. Available: https://github.com/szabodabo/Memory-Management-Simulation 1.2.3

[15] *Manual de jQuery*, 2021. [Online]. Available: https://jquery.com/ 3.1.1

[16] "Developer survey," StackOverflow, Tech. Rep., 2020. [Online]. Available: https://insights.stackoverflow.com/survey/2020 3.1.1

[17] *Manual Vue.js*, 2021. [Online]. Available: https://vuejs.org/ 3.1.1

[18] *Manual Svelte*, 2021. [Online]. Available: https://svelte.dev/ 3.1.1

[19] *Manual React*, 2021. [Online]. Available: https://reactjs.org/ 3.1.1

[20] E. Gamma, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.

[21] E. Freeman and E. Freeman, *Head First: Design Patterns*, M. Loukides, Ed. O'Reilly, 2004.

[22] *Manual SVG.js*. [Online]. Available: https://svgdotjs.github.io/docs/3.0/ 4.1.1

[23] *Documentació de la llibreria i18next*. [Online]. Available: https://react.i18next.com/ 4.1.2

[24] *Documentació de React i18next*. [Online]. Available: https://react.i18next.com/ 4.1.2

[25] *Manual de KaTeX*. [Online]. Available: https://katex.org/ 4.1.5

[26] W. Stallings, *Sistemas Operativos*. Pearson Educación, 2005. 4.2, 4.3

[27] johannesvalks (https://math.stackexchange.com/users/155865/johannesvalks), "How can i find the points at which two circles intersect?" Mathematics Stack Exchange. [Online]. Available: https://math.stackexchange.com/q/1367732 4.2.1

[28] "Jornadas de enseñanza universitaria de la informática." [Online]. Available: http://www.aenui.net/nws/pages/jenui.php 5.3