UNIVERSITAT POLITÈCNICA DE CATALUNYA (UPC) – BarcelonaTech

Facultat d'Informàtica de Barcelona (FIB)

# Implementació d'una interfície gràfica per a un motor d'anàlisi de xarxes

**Autor**
Sergi Férriz Terensi

**Directors**
Luis Velasco Esteban, Marc Ruiz Ramírez

**Titulació**
Grau en Enginyeria Informàtica

**Especialitat**
Tecnologies de la Informació

# Resum

L'arribada de nous serveis i tecnologies provoca que les operadors de telecomunicacions hagin d'introduir canvis constants per millorar la seva infraestructura i la gestió d'aquesta. La planificació de xarxa, l'anàlisi tecno-econòmic i l'estudi de rendiment de xarxa són tasques que requereixen mètodes de computació i eines de simulació de xarxes, entre altres. És per això que s'ha proposat CURSA-SQ com a metodologia escalable per realitzar simulacions ràpides i fiables en escenaris de xarxa complexes. No obstant, la implementació actual de CURSA-SQ no disposa d'una interfície gràfica adequada per a facilitar l'ús d'aquest potent motor de simulacions als planificadors de xarxa i altres usuaris de l'ecosistema d'operadors de telecomunicacions. L'objectiu d'aquest projecte és dissenyar i implementar una interfície gràfica per a millorar la usabilitat del servidor de simulacions existent basat en CURSA-SQ.

# Resumen

La llegada de nuevos servicios y tecnologías provoca que las operadores de telecomunicaciones tengan que introducir cambios constantes para mejorar su infraestructura y la gestión de ésta. La planificación de red, el análisis tecno-económico y el estudio de rendimiento de red son tareas que requieren métodos de computación y herramientas de simulación de redes, entre otras. Es por eso que se ha propuesto CURSA-SQ como metodología escalable para realizar simulaciones rápidas y fiables en escenarios de red complejos. No obstante, la implementación actual de CURSA-SQ no dispone de una interfaz gráfica adecuada para facilitar el uso de este potente motor de simulaciones a los planificadores de red y otros usuarios del ecosistema de operadores de telecomunicaciones. El objetivo de este proyecto es diseñar e implementar una interfaz gráfica para mejorar la usabilidad del servidor de simulaciones existente basado en CURSA-SQ.

# Abstract

The advent of new services and technologies pushes telecom network operators to continuously upgrade and improve their infrastructures and the way they are managed. Network planning, techno-economic analysis, and network performance evaluation are common network management tasks that require the use of computing methods and tools including, among others, network simulation. In this regard, CURSA-SQ has been recently proposed as a scalable method that allow fast and accurate simulations involving complex network scenarios. However, current implementation of CURSA-SQ lacks from a proper interface to facilitate the use of such a powerful simulation engine by network planners and other users from the telecom network operator ecosystem. The goal of this project is to design and implement a graphic interface to improve the usability of an already existent simulations server based on CURSA-SQ.

# Agraïments

A la meva mare, per ser un pilar incansable a la meva vida i bolcar-se tant en mi.

Al meu pare, per encoratjar-me a lluitar pel que vull a la vida.

A la Joana, per endolcir tot el procés i per donar-me tota la tendresa del món.

Als meus amics, sense els seus recordatoris aquest projecte potser encara estaria per fer.

Gràcies.

# 1 Contextualització

## 1.1 El Projecte CURSA-SQ

Cada cop que s'introdueix una nova tecnologia de xarxa (com per exemple, l'imminent 5G), apareixen tot un seguit de nous serveis que no eren possibles en la tecnologia anterior. Aquests serveis proposen un repte tecnològic als operadors de xarxes, que tenen molt poc temps per adaptar-les a les noves necessitats.

Davant d'aquesta problemàtica, el projecte CURSA-SQ ofereix una metodologia per analitzar el tràfic generat en una xarxa en la qual s'injecten uns serveis[1]. D'aquesta forma, els operadors poden reaccionar a temps basant-se en els resultats d'aquestes simulacions.

En el moment de l'escriptura d'aquest treball, l'única implementació de la metodologia proposada pel projecte CURSA-SQ és un sistema basat en *Python* que, com veiem a la Figura 1 consta de les següents parts:

- **Servidor:** respon a peticions de simulació, i realitza tots els càlculs.
- **Client:** a partir d'uns fitxers *JSON* de configuració, realitza les crides pertinents al Servidor per a realitzar la simulació, i retorna els resultats en *JSON* també.

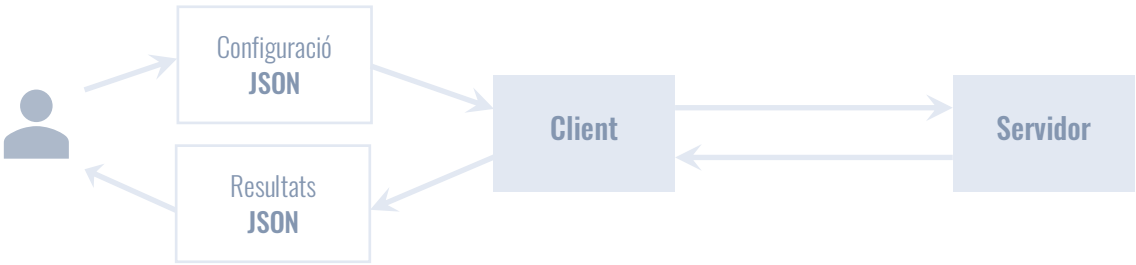


*Figura 1: Estructura de la implementació existent de CURSA-SQ*

Cal comentar que gran part de la terminologia d'aquest treball està en anglès, així com tot el codi. Això és degut a que seguim la nomenclatura proposada en l'article [1], publicat de forma internacional, i sovint aquesta nomenclatura no té traducció o la traducció és massa literal (per exemple: *sink*).

## 1.2 Formulació del problema

El problema que es presenta és la poca usabilitat que suposa la implementació esmentada anteriorment.

L'objectiu del treball, per tant, és la implementació d'una interfície gràfica que en faciliti la usabilitat i configuració de les simulacions, i a l'hora s'abstregui de càlculs reutilitzant el codi existent del Servidor de simulacions.

Aquest és un Treball de Fi de Grau de l'especialitat de Tecnologies de la Informació de la Facultat d'Informàtica de Barcelona (Universitat Politècnica de Catalunya), amb la modalitat A.

## 1.3 Terminologia

- **Single Page Application (SPA):** pàgina web que interacciona amb l'usuari canviant dinàmicament el continguts sense la necessitat de causar una nova càrrega de pàgina. Això implica incloure tot el codi necessari (*HTML*, *CSS* i *JS*) a la càrrega inicial, i carregar els demés continguts de forma dinàmica.
- **Javascript:** llenguatge de programació que s'executa al navegador i que permet fer pàgines web dinàmiques.
- **Typescript:** llenguatge de programació, definit com un superconjunt tipat de *Javascript* que transpila a *Javascript*.

## 1.4 Actors implicats

Els diferents actors implicats en el projecte són els següents:

- **Directors del projecte:** Luis Velasco Esteban i Marc Ruiz Ramírez. A més a més de ser els directors del TFG, són els creadors de la metodologia CURSA-SQ, i per tant, tenen interès en tenir un accés més usable a l'eina.

  S'encarregaran de definir els criteris d'acceptació del software i de validar-lo.
- **Desenvolupador:** l'estudiant que realitza aquest treball, Sergi Férriz Terensi.
- **Usuaris:** totes aquelles persones a les quals va destinada l'aplicació, principalment personal acadèmic i de recerca, així com personal dels departaments de planificació dels operadors de xarxes.

# 2 Estat de l'art

## 2.1 Sistemes de cues

Per a realitzar simulacions de tràfic de xarxes s'utilitzen models, que són abstraccions teòriques de les xarxes que ens permeten estudiar el seu funcionament. Per a xarxes de gran escala el que s'utilitza són models basats en cues, que són nodes que tenen una entrada i una sortida de tràfic de dades.

Les cues se solen representar utilitzant la notació de Kendall[2], que les descriu en tres factors A/S/c:

- **A:** descriu com és el flux de dades d'entrada.

  Alguns dels codis acceptats són *M* (de Markovià), on els paquets arriben de forma discreta seguint una distribució de Poisson, o *G* (de general), on els paquets poden arribar seguint qualsevol distribució estadística.

- **S:** descriu el temps de servei o temps que tarda la cua en processar les dades de sortida.

  Alguns dels codis acceptats són *M* (de Markovià), on el temps de processat segueix una distribució exponencial, que és l'invers a Poisson, o *G* (de general), on el temps de servei pot estar descrit per qualsevol distribució estadística.

- **c**: el número de servidors que té la cua. Per simplificar, sempre considerarem 1.

Analitzant la literatura existent sobre els sistemes de cues, veiem que la major part de contribucions proposen simulacions discretes basades en cues M/M/1[3]. Aquestes, però, no serveixen a l'hora de modelar un tràfic dinàmic.

Cal doncs dissenyar un sistema basant en cues G/G/1, on els tràfics generats siguin realistes. Els autors de l'article [4] proposen una metodologia basada en patrons de tràfic diaris i distribucions estadístiques per modelar la variació. No obstant, aquesta solució tampoc contempla la generació de patrons per els nous serveis que puguin aparèixer, ja que encara no disposem de dades reals.

Per a que els tràfics siguin realistes, la solució que proposa la metodologia CURSA-SQ és derivar el tràfic generat en funció d'informació addicional dels nous serveis que descriguin el comportament esperat dels usuaris o les pròpies característiques del servei.

Un altre factor decisiu a l'hora de dissenyar les simulacions és l'escalabilitat. Com destaca l'article [5], els sistemes de cues basats en paquets d'informació discrets suposen un problema a l'hora de realitzar simulacions amb una gran quantitat de paquets, ja que el temps d'execució sovint supera el del propi període simulat.

Per això, CURSA-SQ es basa en fluxos de tràfic continus, que són molt més ràpids de simular.

## 2.2 Estudi de mercat

Un cop fixat l'objectiu del treball, cal decidir quines tecnologies s'utilitzaran per desenvolupar la solució. Per fer-ho, s'han realitzat varis estudis de mercat a nivells tecnològics diferents.

### 2.2.1 Alternatives existents

Abans de decidir si s'implementava una solució des de zero o es feia ús d'algun simulador existent, s'han avaluat algunes de les alternatives existents:

- **OMNeT++**[6]

  OMNeT++ és un framework extensible i modular escrit en *C++* creat per desenvolupar simuladors de xarxes. Porta integrada una interfície per visualitzar les xarxes, un llenguatge propi per definir les topologies de xarxa i fins i tot un IDE propi basat en Eclipse.

- **Flow Simulator**[7]

  Flow Simulator és una eina per a realitzar simulacions de xarxes basades en cues, especialment dissenyat per posar a prova l'arquitectura de xarxa *FAN* (*Flow Aware Networking*).

Tot i que ambdues solucions ofereixen un bon sistema per implementar un simulador de xarxes, no aporten eines per a la construcció d'interfícies gràfiques que s'adaptin al nostre cas de generació de tràfics de forma parametritzada.

Per tant, donat que la part de simulador ja està implementada i el valor afegit de la part gràfica no és suficient, s'ha decidit utilitzar tecnologies d'ús més general especialitzades en desenvolupament d'interfícies gràfiques.

### 2.2.2 Alternatives de frontend

En aquest estudi, s'han tingut en compte diferents frameworks i llibreries (per simplificar, d'ara endavant, frameworks) basats en *Javascript*, partint de la base que el desenvolupador del treball té experiència professional desenvolupant en aquest llenguatge.

Una primera cerca per Internet ens indica que els frameworks més populars són els següents:

- **Angular:** versió millorada d'*angularJS*, el framework de Google que va popularitzar el model de SPA. La nova versió es presenta com una versió més moderna i ràpida per construir webs mòbils i d'escriptori. Té terminologies i eines pròpies per facilitar una bona estructuració del codi.
- **React:** desenvolupat per l'equip de Facebook, és una llibreria que facilita la creació d'UIs interactives. No és un framework pròpiament dit, ja que no inclou eines per gestionar l'enrutament, la lògica o les dades. Ofereix, per tant, molta

llibertat al desenvolupador a nivell de quines eines vol fer servir, però a la vegada exigeix més coneixements de programació.

- **Vue.js:** creada per un ex-treballador de Google i inspirada en les millors parts d'*angularJS*, es presenta com una llibreria per crear UIs interactives a la vegada que ofereix un conjunt d'eines opcionals com a framework per construir SPAs.

Si comparem en la Figura 2 els tres frameworks pel nombre de cerques a Internet veiem que **Angular** ha tingut més popularitat des del principi (gràcies a *angularJS*), **React** ha anat guanyant popularitat en els últims 4 anys, fins a superar els demés, i **Vue.js** no és tant popular, tot i que està experimentant un creixement important.

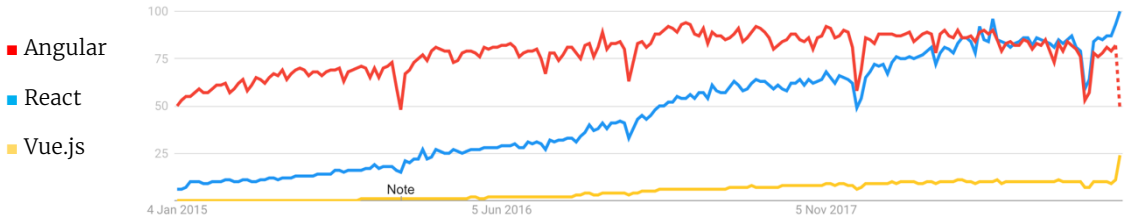


*Figura 2: Tendències de cerques per framework o llibreria (Google Trends)*[8]

Una altra opció és estudiar la satisfacció dels mateixos desenvolupadors, expressada a través d'una enquesta per la pàgina web *Stack Overflow*. Considerem que són dades rellevants, donat que és una comunitat molt gran i activa que inclou desenvolupadors de tota mena.

Observant la Figura 3, veiem que **React** és la llibreria més satisfactòria per programar.

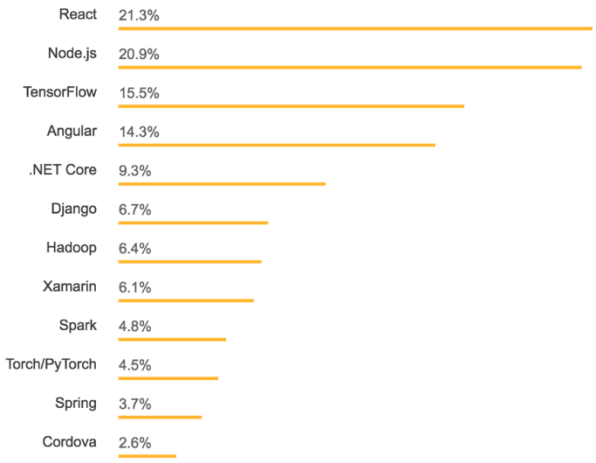


*Figura 3: Most Wanted Frameworks, Libraries, and Tools (Stack Overflow)*[9]

L'últim criteri a considerar és la demanda d'aquests frameworks en el mercat laboral. La mesura s'ha realitzant buscant ofertes de feina amb els següents criteris:

- Contrastant les ofertes de diferents proveïdors d'ofertes: LinkedIn, Infojobs i Glassdoor.
- Buscant ofertes que continguin la paraula *Angular*, *React* o *Vue.js.*
- Localitzades a Espanya, i publicades amb menys d'un mes.

El resultat que es mostra en la Figura 4 és que **Angular** guanya en la major part dels buscadors, seguit per **React**. La presència de **Vue.js** és bastant escassa.

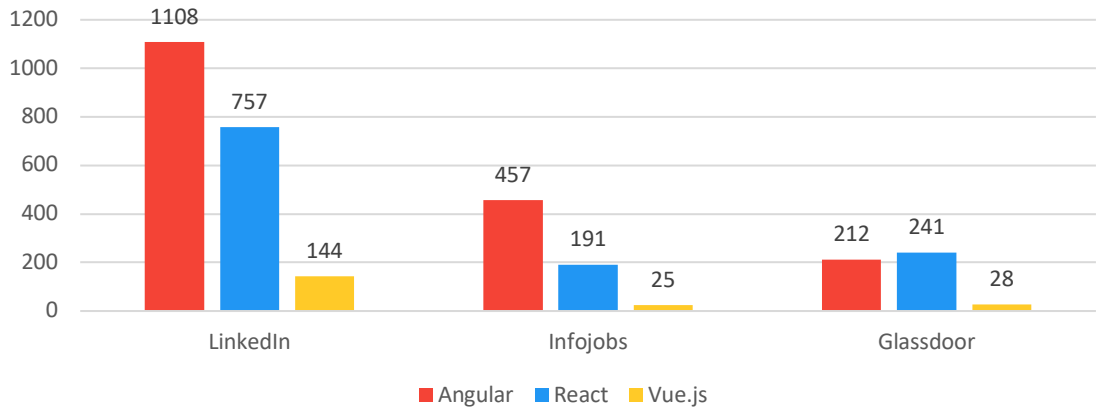


*Figura 4: Comparativa de frameworks en buscadors de feina*

Tenint en compte les dades anteriors, el framework escollit és **React** pels motius següents:

- Ofereix flexibilitat a l'hora de programar, fet que anirà bé al tractar-se d'un software poc convencional.
- Està suficientment establert per assegurar que, en cas de que es vulgui continuar el software a posterior, es podrà trobar un desenvolupador capaç.
- El repte personal i professional que ofereix i la opinió pròpia del desenvolupador.

### 2.2.3 Alternatives de backend

A l'hora d'escollir la tecnologia pel *backend*, el pes de la decisió radica en escollir el llenguatge, més que el propi framework. De fet, cada llenguatge sol tenir un o dos frameworks preferits pel públic a l'hora de desenvolupar.

Les alternatives són:

- **Javascript:** gràcies al projecte *Node.js*, que permet executar codi *Javascript* en entorns que no siguin un navegador. *Node* està dissenyat per desenvolupar aplicacions escalables en xarxa.
- **C#:** llenguatge fortament tipat, orientat a objectes i funcional, dissenyat per Microsoft per fer servir les seves llibreries. Inicialment, només es podien desenvolupar aplicacions per Windows, però les noves versions del framework *.NET* són multi-plataforma i de software lliure.

- **Java:** llenguatge fortament tipat i orientat a objectes que està dissenyat per executar-se en una màquina virtual. El principal framework per fer aplicacions de backend és *Spring*.
- **Python:** llenguatge de scripting que té com a objectiu fer fàcil i ràpid el desenvolupament. Compta amb una comunitat molt activa, oferint mòduls per facilitar tot tipus de tasques, entre elles la de construir aplicacions de backend.
- **PHP:** llenguatge de scripting inicialment pensat per desenvolupament web. La seva gran popularitat en el passat va fomentar el creixement d'innumerables frameworks que oferien eines per donar solució a les carències que té el llenguatge. Alguns d'aquests frameworks són *Symfony*, *Laravel* i *CodeIgniter*.

L'elecció del llenguatge, en aquest cas, és mes subjectiva, ja que tots ofereixen alternatives viables. Al final s'ha escollit fer servir **Javascript**, com en el *frontend*, partint de la premissa de reduir el temps d'aprenentatge i el nombre d'eines diferents.

### 2.2.4 Alternatives de llibreries de gràfics

L'última decisió clau a l'hora de planificar el projecte és decidir quina llibreria de gràfics farem servir per representar les xarxes. Les alternatives que hem trobat són les següents:

- **projectstorm/react-diagrams:** llibreria senzilla per fer diagrames en *React*. Actualment no rep suport i s'està reescrivint en una nova llibreria (projectstorm/react-canvas).
- **mxGraph:** llibreria totalment extensible per fer diagrames en *Javascript* pur. Utilitzat per l'editor de diagrames online *draw.io*. No disposa de tipat amb *Typescript* però si que disposa d'una documentació online.
- **A mida:** si cap de les dues anteriors s'adequa a les necessitats, sempre existeix l'opció de fer una llibreria pròpia a mida de les necessitats del projecte.

Després de proves exhaustives amb les tres opcions, s'ha acabat triant l'opció de **mxGraph**, ja que és la més completa. Tot i que el projecte és molt gran i el fet de no tenir tipus dificulta entendre com va el codi, fer una solució a mida seria massa complicat pel treball. L'opció de **react-diagrams** té masses limitacions i ha estat descartada.

## 2.3 Conclusió

La solució del projecte s'implementarà amb les següents tecnologies:

- **React** amb *Typescript* al *frontend*, i **mxGraph** per la gestió de gràfics.
- **NodeJS** amb *Typescript* pel *backend*.

S'ha afegit *Typescript* a les dues parts, ja que es pretén que el codi sigui el més mantenible i clar possible.

# 3 Abast del projecte

## 3.1 Objectiu

El primer objectiu del treball és la implementació d'una interfície gràfica que permeti dissenyar xarxes basades en models de cues i definir els serveis a partir dels quals es generarà el tràfic seguint la metodologia CURSA-SQ.

El segon objectiu és implementar una infraestructura que permeti connectar la interfície amb el servidor de CURSA-SQ existent, de forma que l'usuari pugui executar simulacions i veure'n els resultats. Aquesta infraestructura ha de ser escalable.

## 3.2 Sub-objectius i requeriments

Partint dels objectius esmentats a l'apartat anterior, s'ha realitzat un llistat de característiques que s'esperen del sistema inspirat en el *backlog* de la metodologia àgil *Scrum*.

Podem veure el *backlog* a la Taula 1, organitzat per grups de tasques (èpiques en terminologia *Scrum*) i ordenat per prioritat. Es considera que les tasques en verd és el mínim requeriment de tasques per a tenir un producte funcional o **Minimum Viable Product (MVP)**.

*Taula 1: Backlog de tasques amb el MVP*

| Esquemes de xarxa | |
|---|---|
| ★ | **Gestió bàsica**<br>Crear, modificar, guardar i eliminar esquemes de xarxa. |
| ★ | **Generadors**<br>Crear, configurar i eliminar generadors. Definir perfils d'usuaris i serveis. |
| ★ | **Cues**<br>Crear, configurar i eliminar cues. |
| ★ | **Sinks**<br>Crear, configurar i eliminar *sinks*. |
| ★ | **Connexions**<br>Connectar generadors, cues i *sinks* entre si. |
| ★ | **Validació**<br>Validar la correctesa dels esquemes i mostrar errors. |

| | |
|---|---|
| | **Perfils**<br>Crear perfils de generadors, cues, *sinks*, etc. que permetin reutilitzar elements. |
| | **Enrutadors**<br>Crear, configurar i eliminar enrutadors. Configurar els ports i cues internes. |
| | **Upstream / downstream**<br>Configuració d'un esquema en els dos sentits del tràfic. |
| **Simulacions** | |
| ★ | **Gestió bàsica**<br>Crear, modificar, guardar i eliminar simulacions. |
| ★ | **Gràfics de tràfic**<br>Mostrar gràfics de tràfic amb les dades obtingudes, i en temps real. |
| ★ | **Simulacions concurrents**<br>Connectar el sistema a múltiples instàncies del simulador. |
| ★ | **Exportar simulació**<br>Generar un fitxer comprimit amb totes les dades de la simulació. |
| | **Upstream / downstream**<br>Simular tràfic en els dos sentits. |
| **General** | |
| ★ | **Inst·lalador**<br>Crear un instal·lador per a facilitar la configuració del sistema. |
| | **Gestió d'usuaris**<br>Crear, modificar i eliminar usuaris amb contrasenya des de l'aplicació. |
| | **Idiomes**<br>Suport a múltiples idiomes. |

## 3.3 Anàlisi d'obstacles

En el desenvolupament d'aquest projecte, existeixen els següents obstacles:

- **Restricció temporal:** principalment deguda als horaris del desenvolupador, que haurà de combinar un horari laboral de jornada completa, amb el temps que exigeixi la realització. A més a més, tot això en el marc del calendari del TFG.

  Per a superar aquest obstacle, el TFG no s'ha matriculat fins a tenir bastant acabat el desenvolupament.

També s'ha definit molt bé els terminis de les tasques, de forma que es pugui portar un control de si és viable acabar el projecte en el calendari establert.

- **Aprenentatge de la tecnologia:** l'elecció de frameworks que el desenvolupador no coneix pot suposar un obstacle temporal.

  Tot i ser llibreries desconegudes, estan basades en un llenguatge conegut, i per això s'espera que l'aprenentatge sigui més fàcil.

  Previ al començament del treball, el desenvolupador ha fet un projecte petit com a pràctica que consta d'un tres en ratlla basat en un tutorial de *React.*

- **Capacitat de la màquina de treball:** les simulacions que tractarà el projecte poden ser molt grans en quantitat de dades. Per això, pot ser que en algun moment l'ordinador del desenvolupador es quedi curt.

  En aquests casos, es dependrà més de que els directors del projecte realitzin les proves pertinents en màquines més preparades.

  També es planteja la possibilitat d'executar el codi en un ordinador de sobretaula, que podria emmagatzemar més dades però les simulacions serien més lentes.

- **Males pràctiques de programació:** *React*, a la vegada que dóna llibertat d'implementació, pot resultar perillós si no es programa de forma ben estructurada o si es duen a terme males pràctiques de programació.

  Això requerirà d'un estudi previ exhaustiu de la llibreria, de bones pràctiques i investigar projectes existents com a font d'inspiració.

# 4 Metodologia

## 4.1 Metodologia de treball

La intenció del projecte és adoptar, en la mesura del possible, la metodologia *Scrum*. Així, en comptes de dissenyar funcionalment el producte en la seva totalitat de forma prèvia seguint el model en cascada, ens centrarem en objectius més a curt termini.

La Figura 5 mostra les diferents passes que segueix la metodologia *Scrum*:

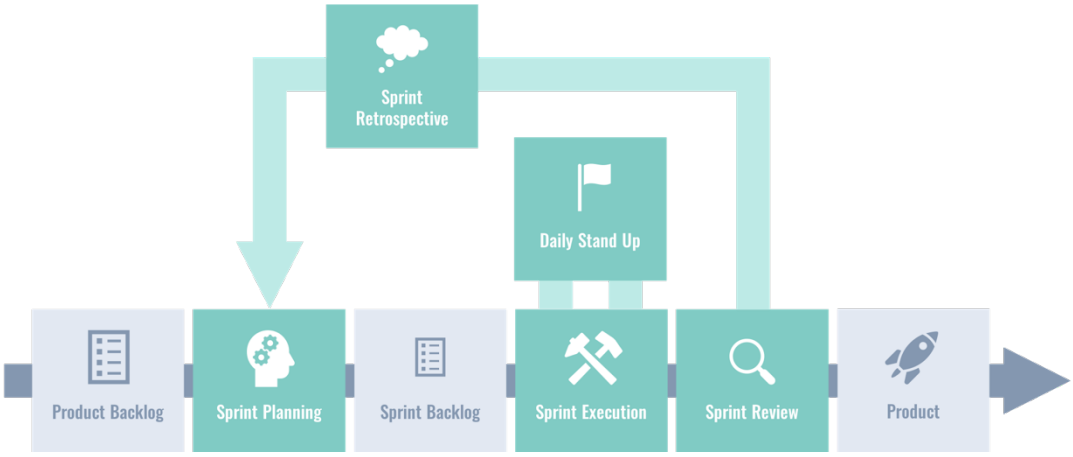


*Figura 5: Cicle Scrum*

La metodologia *Scrum* comença a l'hora de definir el *backlog* del producte, que consisteix en definir tasques petites i ordenar-les per prioritat. Cada tasca té una descripció, una valoració del cost, i uns criteris d'acceptació que s'han de complir per a considerar-se acabada. Aquesta feina la realitzaran conjuntament els directors i el desenvolupador.

A continuació, s'inicia un cicle de *sprints*, on cada *sprint* té una durada fixa. En el nostre cas serà de 2 setmanes, tot i que podrà variar en funció del calendari.

L'*sprint* comença amb una *Planning*, on es defineix quines tasques es faran en aquell *sprint* (*sprint backlog*). Durant l'*sprint* es van realitzant aquestes tasques i es realitzen reunions diàries (*daily stand up*) per compartir amb el grup el progrés. Un cop s'acaba l'*sprint*, es revisa el que s'ha acabat i el que no (*review)* i es fa una *retrospectiva* amb el grup per parlar de quines coses han anat bé i quines es poden millorar.

De tot aquest cicle d'*sprint*, només ens quedarem amb la *planning* i la *review*. Aquestes es faran a les reunions amb els directors, que es faran en acabar cada *sprint* i començar el següent.

Les altres dinàmiques s'han descartat perquè no hi ha un grup de desenvolupadors.

## 4.2 Eines

Les eines que es faran servir per desenvolupar el projecte són les següents:

- **MacBook Pro:** màquina de treball del desenvolupador.
- **Visual Studio Code:** IDE multiplataforma, de software lliure i desenvolupat per Microsoft.
- **Git:** control de versió per al codi.
- **Treescale:** repositori de contenidors *Docker*.
- **Yarn:** gestor de dependències per a projectes amb *Javascript*. Alternativa a *NPM*.
- **Google Chrome:** navegador sobre el qual es realitzarà el major gruix de proves.
- **Trello:** panell de tasques per gestionar el *backlog*. Permet gestionar les tasques pendents, en progrés i completades.
- **Skype:** per a les reunions no presencials amb els directors.
- **Word:** per a redactar la memòria del projecte.

## 4.3 Mètodes de validació

Per a validar que el projecte s'està desenvolupant de forma correcta i no hi ha errors de funcionament ni de plantejament, s'ha acordat amb els directors fer reunions periòdiques cada una o dos setmanes segons conveniència.

Prèviament a aquestes reunions, el desenvolupador pujarà al repositori els contenidors actualitzats amb els canvis de forma que els directors puguin fer proves prèvies. Un cop a la reunió, es discutirà sobre les noves implementacions i sobre les següents passes. L'objectiu és que els possibles errors es vagin arreglant a cada sessió i no es deixi tota la validació al final.

Aquestes reunions, inicialment, seran presencials, degut a que al principi el projecte estarà molt obert, i a mesura que es vagi tancant i els canvis siguin més concrets, passaran a ser mitjançant videotrucades.

Extraordinàriament, si el temps ho permet, es generaran tests unitaris en els diferents components del *frontend* per validar tots els casos de funcionament d'aquests i per fer el codi més mantenible de cara al futur.

Un cop acabat el MVP, es passarà a un període de **UAT** (*User Acceptance Test*), on es valida que el producte funciona d'acord amb les especificacions[10]. En aquest període els directors realitzaran proves amb dades d'escenaris reals i validaran que tots els fluxos funcionin correctament. Un cop validat, en cas de disposar de temps abans de redactar la memòria, s'avançarà amb la resta de tasques que no formen part del MVP.

# 5 Planificació

## 5.1 Duració del projecte

La duració del projecte és d'aproximadament de 12 mesos ampliables a 15 delimitats per les següents fites:

- Primera reunió amb els directors (juliol 2018)

  Aquesta reunió és la primera presa de contacte amb el tema del treball, i estableix els objectius i el camí a seguir.

- Gestió del Projecte (setembre – octubre 2018)

  Les diferents fases de la gestió del projecte (contextualització, abast, planificació, etc.) es realitzaran al principi del projecte, i s'entregaran més tard en la Fita Inicial quan es matriculi el treball.

- Entrega del producte (juny 2019)

  Es realitzarà al mateix temps que l'entrega de la memòria, ja que els últims retocs es faran duran l'escriptura d'aquesta.

- Entrega de la memòria (juny 2019)

  L'entrega s'ha de realitzar amb suficient antelació respecte al torn de lectura, i en aquest període es tindrà temps per preparar la defensa.

- Torn de lectura (juliol 2019)

  En cas de no disposar de temps per presentar el treball en aquesta data, es disposa d'un segon torn de lectura de TFG a l'octubre. És té en compte només com a mesura de contingència.

## 5.2 Tasques

A la **Error! Reference source not found.** es mostren les tasques que s'han definit a l'abast, valorades en hores de treball, i dividides en sub-tasques. S'inclouen, a més a més, les tasques que no són pròpiament de programació sinó del treball.

En verd apareixen les tasques que no estan dins del MVP. Els totals d'hores apareixen separats en funció de si tenen en compte les tasques fora del MVP (en verd) o no (en negre).

*Taula 2: Estimació de tasques*

| Tasca | Hores | |
|---|---|---|
| **Disseny i planificació** | **55** | 55 |
| Reunió inicial i creació del *backlog* | 3 | - |
| Aprenentatge dels frameworks | 15 | - |
| Creació de repositoris i estructures bàsiques de codi | 4 | - |
| **Gestió del Projecte** | **33** | 33 |
| Abast i contextualització | 15 | - |
| Planificació temporal | 4 | |
| Gestió econòmica i sostenibilitat | 4 | |
| Documentació | 10 | |
| **Desenvolupament del projecte** | **370** | 480 |
| **Esquemes de xarxa** | **200** | 260 |
| Gestió bàsica | 45 | - |
| Generadors | 70 | - |
| Cues | 25 | - |
| *Sinks* | 5 | - |
| Connexions | 30 | - |
| Validació | 10 | - |
| Perfils | 15 | - |
| Enrutadors | - | 40 |
| Upstream/downstream | - | 20 |
| **Simulacions** | **150** | 165 |
| Gestió bàsica | 80 | - |
| Gràfics de tràfic | 35 | - |
| Simulacions concurrents | 15 | - |
| Exportar simulació | 20 | - |
| Upstream/downstream | - | 15 |
| **General** | **20** | 55 |
| Instal·lador | 20 | - |
| Gestió d'usuaris | - | 15 |
| Idiomes | - | 20 |
| **Proves finals** | **15** | 15 |
| Realització de les proves | 10 | - |
| Correcions i millores | 5 | - |
| **Documentació** | **50** | 50 |
| Redactar la memòria | 40 | - |
| Preparar la presentació final | 10 | - |
| **TOTAL** | **490** | 600 |

Aquestes tasques es poden veure organitzades seguint el model de diagrama de Gantt en la Figura 6. Els punts negres o *milestones* són les reunions de seguiment amb els directors.

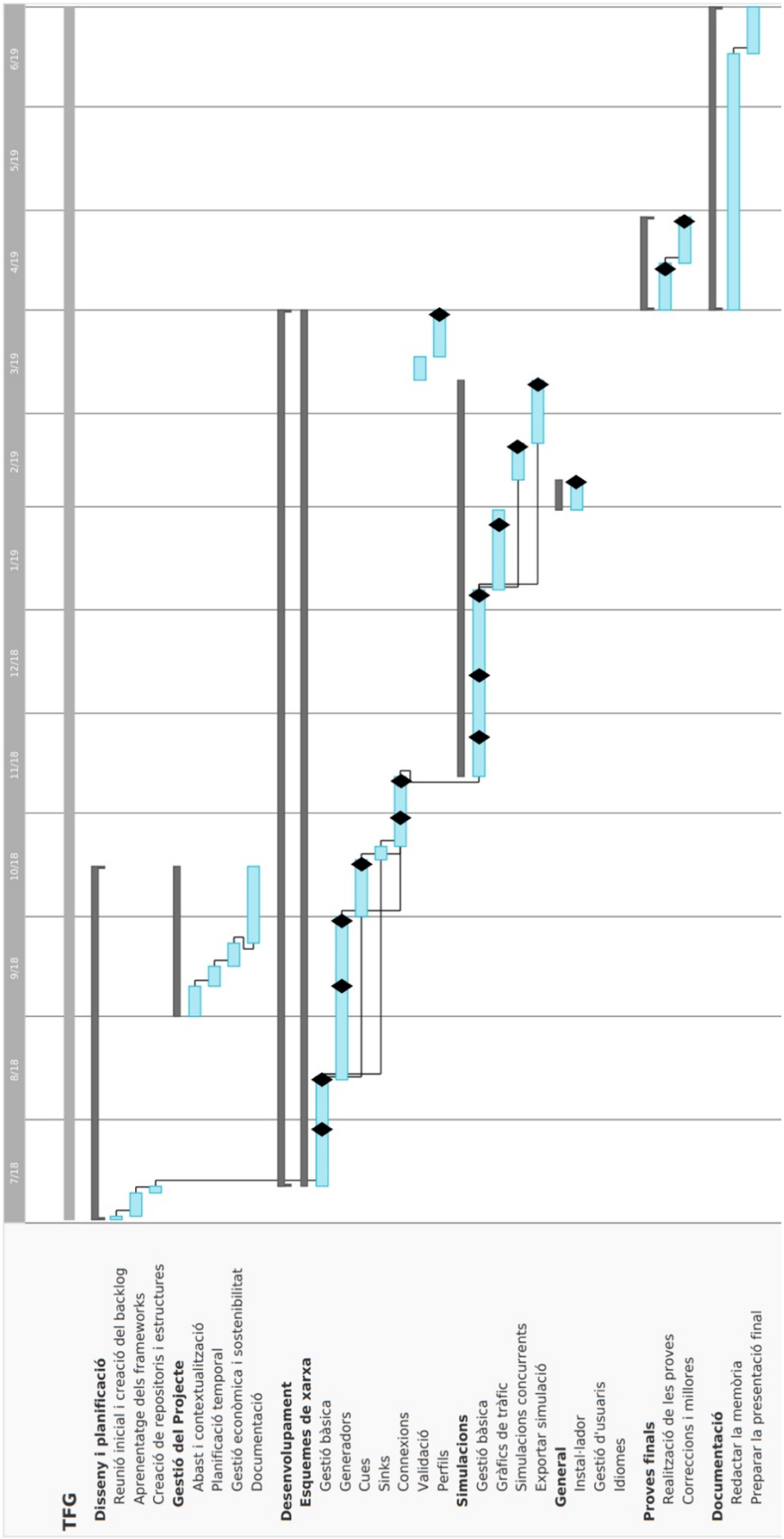


*Figura 6: Diagrama de Gantt*

## 5.3 Anàlisi de riscs i alternatives

En aquest apartat analitzem els obstacles esmentats anteriorment i els plans alternatius:

- **Restricció temporal:** en cas de no arribar a temps al període de lectura, es planteja l'opció de demanar una pròrroga i presentar el treball en el torn de lectura d'octubre.

  Analitzant el nombre d'hores del treball, veiem que hi ha una mitjana aproximada de 40 hores mensuals. Si considerem una pròrroga de 3 mesos, seria una càrrega extra de 120 hores.

- **Aprenentatge de la tecnologia:** en cas de necessitar més temps per aprendre el framework, ampliaríem el període d'aprenentatge a 30 hores.

- **Capacitat de la màquina de treball:** aquest obstacle no suposa cap risc a nivell d'hores, ja que en cas de veure que la capacitat és insuficient simplement hauríem de fer les proves en ordinadors més potents.

- **Males pràctiques de programació:** com que l'aprenentatge dels frameworks es farà també durant el procés de desenvolupament, podem deixar un marge de 20 hores per refactoritzar codi dolent.

# 6 Gestió econòmica

## 6.1 Estimació de costos

### 6.1.1 Recursos humans

Els costos de recursos humans es poden veure a la Taula 3, i s'obtenen a partir de les estimacions del *backlog*, assignant un cost hipotètic per hora a cada tasca en funció del preu de mercat del rol (veure Taula 4) que duria a terme cada tasca.

Per simplificar, s'han agrupat els rols de dissenyador, desenvolupador de *frontend*, *backend* i *tester* en desenvolupador.

*Taula 3: Pressupost de recursos humans*

| Tasca | Cap de projecte | Desenvolupador | Cost |
|---|---|---|---|
| **Disseny i planificació** | **36 h** | **19 h** | **1.915 €** |
| Reunió inicial i creació del *backlog* | 3 h | - | 120 € |
| Aprenentatge dels frameworks | - | 15 h | 375 € |
| Creació de repositoris i estructures bàsiques de codi | - | 4 h | 100€ |
| **Gestió del Projecte** | **33 h** | **-** | **1.320 €** |
| Abast i contextualització | 15 h | - | 600 € |
| Planificació temporal | 4 h | - | 160 € |
| Gestió econòmica i sostenibilitat | 4 h | - | 160 € |
| Documentació | 10 h | - | 400 € |
| **Desenvolupament del projecte** | **15 h** | **355 h** | **9.475 €** |
| **Esquemes de xarxa** | **9 h** | **191 h** | **5.135 €** |
| Gestió bàsica | 2 h | 45 h | 1.205 € |
| Generadors | 3 h | 70 h | 1.870 € |
| Cues | 0,5 h | 20 h | 520 € |
| *Sinks* | 0,5 h | 3 h | 95 € |
| Connexions | 0,5 h | 30 h | 770 € |
| Validació | 1,5 h | 8 h | 260 € |
| Perfils | 1 h | 15 h | 415 € |
| **Simulacions** | **6 h** | **144 h** | **3.840 €** |
| Gestió bàsica | 3 h | 70 h | 1.870 € |
| Gràfics de tràfic | 1,5 h | 40 h | 1.060 € |

| | | | |
|---|---|---|---|
| Simulacions concurrents | 1 h | 16 h | 440 € |
| Exportar simulació | 0,5 h | 18 h | 470 € |
| **General** | **-** | **20 h** | **500 €** |
| Instal·lador | - | 20 h | 500 € |
| **Proves finals** | **10 h** | **5 h** | **525 €** |
| Realització de les proves | 10 h | - | 400 € |
| Correcions i millores | - | 5 h | 125 € |
| **Documentació** | **50 h** | **-** | **2.000 €** |
| Redactar la memòria | 40 h | - | 1.600 € |
| Preparar la presentació final | 10 h | - | 400 € |
| **TOTAL** | **111 h** | **379 h** | **13.915 €** |

*Taula 4: Preu de mercat dels rols del projecte*

| Rol | Preu hora |
|---|---|
| Cap de projecte | 40 € / hora |
| Desenvolupador | 25 € / hora |

### 6.1.2 Recursos hardware

Aquest projecte es realitzarà en la seva majoria amb un equip portàtil. A la Taula 5 es mostra el càlcul de l'amortització del hardware considerant que el portàtil té una vida útil de 4 anys, que equival a un 25% d'amortització anual.

*Taula 5: Amortització de hardware*

| Hardware | Preu total | Temps | Amortització |
|---|---|---|---|
| MacBook Pro 13" | 1.199 € | 1 any (25%) | 299,75 € |

### 6.1.3 Recursos software

La gran majoria de software que es farà servir en aquest projecte és lliure o gratuït, a excepció de l'editor de documents amb el qual es realitzarà la memòria.

A la Taula 6 es mostra el cost de l'editor de documents, que al ser una subscripció anual s'adapta perfectament al període del treball i no cal calcular-ne l'amortització.

*Taula 6: Pressupost de recursos software*

| Software | Preu anual |
|---|---|
| Microsoft Office 365 (Word) | 61,65 € |

### 6.1.4 Altres costos indirectes

En la Taula 7 es contemplen els costos indirectes que no són software ni hardware, com el consum energètic i la connexió a Internet.

Pel consum energètic, s'ha tingut en consideració l'energia necessària per carregar el portàtil, la llum i la calefacció a l'hivern, i s'ha estimat un consum diari de 2kWh.

Per la connexió a Internet, la tarifa inclou 300MB simètrics de fibra òptica i una línia fixa de telèfon per 60€. També s'ha dividit en 3 per considerar la part de consum aportada pel treball.

*Taula 7: Pressupost d'altres costos indirectes*

| Concepte | Estimació | | Preu |
|---|---|---|---|
| Consum energètic | 2 kWh / dia | 0,19 € / kWh | 138,70 € |
| Connexió a Internet | 12 mesos | 20 € / mes | 240 € |
| | | **TOTAL** | **378,70 €** |

### 6.1.5 Imprevistos

La Taula 8 especifica els costos derivats de les diferents alternatives als riscos proposades a l'apartat 5.3.

Per simplificar, s'ha valorat el cost extra de recursos humans amb el preu de mercat de desenvolupador, ja que el més probable és que aquesta càrrega de treball l'hagués d'assumir aquest rol.

*Taula 8: Pressupost d'imprevistos*

| Imprevist | Quantitat | Preu | Cost estimat |
|---|---|---|---|
| **Prórroga de 3 mesos** | | | **622,58 €** |
| Recursos humans | 120 h | 25 € / h | 3.000 € |
| Amortització portàtil | 3 mesos | 6,24 € / mes | 18,72 € |
| Consum energètic | 3 mesos | 11,40 € / mes | 34,20 € |
| Connexió a Internet | 3 mesos | 20 € / mes | 60 € |
| | | Probabilitat | 20 % |
| **Aprenentatge de la tecnologia** | | | **75 €** |
| Recursos humans | 15 h | 25 € / h | 375 € |
| | | Probabilitat | 20 % |
| **Refactorització** | | | **150 €** |
| Recursos humans | 20 h | 25 € / h | 500 € |
| | | Probabilitat | 30 % |
| | | **TOTAL** | **847,58 €** |

### 6.1.6 Pressupost

El cost total del projecte es pot veure a la Taula 9, que inclou un marge de contingència d'un 10%.

*Taula 9: Pressupost*

| Recursos | Preu |
|---|---|
| Recursos humans | 13.915 € |
| Costos hardware | 299,75 € |
| Costos software | 61,65 € |
| Altres costos indirectes | 378,70 € |
| Imprevistos | 847,58 € |
| **Subtotal** | **15.502,68 €** |
| Marge de contingència | 10% |
| **TOTAL** | **17.052,95 €** |

## 6.2 Control de gestió

Per a dur a terme un control de si el pressupost inicial s'està complint i anar actualitzant-lo amb les dades reals, es durà a terme un control de gestió basat en les següents accions:

- **Recursos humans**

  Es portarà un registre de les hores invertides en cada tasca. En cas de que aquestes superin el límit establert pel Diagrama de Gantt, s'actualitzarà el pressupost amb una desviació en eficiència:

  Desviació = (consum d'hores estimat – consum d'hores real) x cost estimat

- **Costos directes**

  Tot i que només s'ha contemplat el cost de l'amortització de l'ordinador portàtil i el de l'editor de textos, poden aparèixer nous costos de hardware o de software durant el desenvolupament del projecte.

  El càlcul de la desviació és el següent:

  Desviació = – consum real x cost real

- **Costos indirectes**

  Per a les desviacions de costos indirectes, farem servir la següent fórmula:

  Desviació = (consum estimat – consum real) x cost real

Aquestes desviacions s'aplicaran al pressupost actualitzat i en tots els casos hauran d'incloure una justificació. S'espera que aquestes desviacions no superin en tot cas el marge de contingència assignat al pressupost inicial.

# 7 Sostenibilitat i compromís social

## 7.1 Autoavaluació

Després de realitzar el qüestionari sobre el nivell de formació en sostenibilitat, he pogut analitzar les meves capacitats actuals en aquest tema.

Considero que les dimensions en les que he tret més bons resultats han estat l'ambiental i l'econòmica, quedant la social en tercer lloc.

En primer lloc, estic bastant conscienciat de l'impacte ambiental que tenen els productes i serveis que consumeixo, i això aplica també a les eines TIC i, en aquest cas, al projecte.

Pel que fa a la dimensió econòmica, considero que és la dimensió sobre la qual he rebut més formació durant la carrera, en assignatures com Economia i Empresa o en el propi GEP. Estic al corrent de les alternatives econòmiques que existeixen, com l'economia circular o els diferents tipus d'entitats que formen l'economia social.

També he tingut en compte que, tot i tenir coneixements sobre planificació econòmica i gestió de projectes, aquests són força bàsics. En tot cas, el nivell és acceptable pel Grau que he cursat.

Per últim, la dimensió social és la menys explorada en el meu cas. A nivell professional, fa temps que tinc l'objectiu d'aprendre quines mesures es poden prendre per millorar l'accessibilitat dels productes. Reconec, però, que sovint són característiques a les quals no se'ls dona suficient importància, i vaig posposant l'objectiu.

L'ergonomia tampoc és un concepte amb el qual estigui molt familiaritzat ni acostumi a tenir en compte, però no és un concepte que crec que apliqui al nostre treball.

En quant a les eines de treball col·laboratiu, degut a que el meu entorn laboral està basat en metodologies àgils que fomenten aquest tipus d'eines, en tinc bastant coneixement.

Altres conceptes en aquesta dimensió com justícia social, equitat, diversitat i transparència no els tinc gaire en compte actualment en els meus projectes, tot i que conec les seves implicacions.

## 7.2 Sostenibilitat econòmica

Considero que el projecte ha realitzat un estudi adequat dels recursos materials i humans que requerirà, així com un pressupost detallat dels costos que suposaran cadascun d'aquests recursos.

El pressupost total de 17.000€ és un valor una mica elevat per a un treball acadèmic, però seria més que assequible si considerem que els clients finals són grans empreses de telecomunicacions que poden tenir molt interès en el producte.

En quant a l'estudi econòmic de la vida útil del projecte, hem de tenir en compte que el projecte permetrà reduir considerablement el període d'adaptació de les grans empreses

de comunicacions als nous serveis de dades que puguin anar apareixent o inclús als ràpids canvis d'aquests serveis.

Això pot suposar una optimització molt gran dels recursos de xarxa que tenen aquestes empreses, que a gran escala suposarà una reducció de costos molt gran i una millor oferta de serveis traduïda en majors beneficis.

## 7.3 Sostenibilitat ambiental

L'impacte ambiental considerat per aquest projecte és mínim, degut a que la major part dels recursos utilitzats són software.

Una oportunitat de millora seria canviar el mètode de desplaçament i fer servir el transport públic, però tenint en compte la localització del lloc de treball del desenvolupador resulta complicat.

Un altre element a considerar és el consum energètic durant la realització del projecte, que es pot considerar un consum baix ja que aprofita el consum ja existent a la casa del desenvolupador.

No es farà ús de paper en les fases del desenvolupament del projecte, i en el cas de la memòria, si es requereix d'imprimir-la es farà sempre amb paper reciclat.

En quant a la vida útil del projecte, s'espera que redueixi la quantitat de recursos mal aprofitats al permetre optimitzar les xarxes a través de les simulacions.

## 7.4 Sostenibilitat social

A nivell personal, aquest projecte m'aportarà experiència en desenvolupament de certes tecnologies en les quals tinc interès a nivell professional.

El projecte en un principi no està plantejat per ser una solució de software lliure, ja que depèn de la utilització que els directors vulguin donar, així que en aquest aspecte no serà col·laboratiu.

Una carència del treball és la poca accessibilitat. No s'ha tingut en compte de fer-ho accessible perquè el públic al que va dirigit no és un públic gaire obert, i també pel propi desconeixement del desenvolupador sobre la matèria.

En quant a la vida útil del projecte, quant més ràpida sigui l'adopció dels nous serveis de xarxa per part dels operadors, més qualitat de serveis tindran els usuaris, millorant la seva qualitat de vida.

L'impacte, però, serà més gran per a les empreses que per als propis clients.

# 8 Requisits

A continuació es detallen els requisits establerts per els directors sobre com ha de ser la solució a desenvolupar:

- L'usuari ha de poder crear, modificar, guardar i eliminar *boards*. El *board* tindrà la configuració de l'esquema de xarxa.
- L'usuari ha de poder importar i exportar les dades d'un *board*.
- Per cada *board*, s'han de poder afegir, modificar i eliminar nodes de forma gràfica i intuïtiva. Aquests nodes, a més a més, s'han de poder connectar entre si.

  Els diferents tipus de nodes (vèrtexs d'ara endavant, seguint la notació de grafs) que hi pot haver són els següents:

  - Generadors: simulen el tràfic generat per un grup d'usuaris que fan ús d'un conjunt de serveis.
  - Cues: permeten agregar, distribuir i limitar el tràfic.
  - *Sinks:* permeten observar el tràfic que arribaria a un cert punt de la xarxa.
- Els generadors són elements que generen fluxos de dades. Per això, han de tenir una sola sortida que es pot connectar a altres vèrtexs.
- Els generadors han de poder configurar la taxa de bits (*bitrate* d'ara endavant) màxima, que és la capacitat màxima de dades que poden generar.
- L'usuari ha de poder importar un *.csv* amb una corba que modeli la quantitat d'usuaris que simula un generador en el temps durant un període definit. A més a més, ha de poder configurar els següents paràmetres:

  - **Growth Rate (over X periods):** creixement del nombre d'usuaris (en %) respecte la corba original al cap de X períodes.

    En els punts que hi ha entremig de cada període el creixement es calcula mitjançant interpolació lineal.
  - **Factor:** multiplicador d'usuaris. La corba es defineix amb un seguit de punts amb valors de 0 a 1, i per obtenir el valor real s'han de multiplicar per aquest factor.
- L'usuari ha de poder configurar els serveis que produiran els fluxos de dades en un generador. Cada servei tindrà uns paràmetres de configuració, que expliquen com es comporten els paquets d'informació i els *bursts*, que són grups de paquets enviats consecutivament en un període de temps.

  Els paràmetres són els següents:

  - **Inter-Burst Rate:** La inversa del temps entre *bursts.*
  - **Burst Size:** Quantitat de bytes enviats en cada *burst.*

    - **Packet Size:** Quantitat de bytes enviats en cada paquet.
    - **Inter-Packet Time:** Temps en segons entre paquets

  Aquests paràmetres s'han de poder configurar amb distribucions estadístiques.
- Les cues han de tenir una entrada i una sortida, i accepten múltiples connexions a cadascuna.
- L'usuari ha de poder configurar els següents paràmetres de les cues:
    - **Capacity (k):** Capacitat de la cua en bytes, quantitat de bytes que pot tenir pendents de processar sense perdre informació.
    - **Bitrate (mu):** Velocitat en b/s amb la qual la cua processa els paquets.
- L'usuari ha de poder especificar quin percentatge de dades s'enviarà per cada connexió de sortida de les cues.
- Els *sinks* tindran només una entrada, que pugui acceptar múltiples connexions.
- L'usuari ha de poder guardar generadors i cues existents en una biblioteca de vèrtexs, de forma que es puguin reutilitzar.
- L'usuari ha de poder realitzar simulacions amb les dades d'un *board* configurat prèviament. Els paràmetres de configuració d'una simulació són els següents:
    - Data de referència d'inici. Aquesta data no serà necessàriament la data de començament de la simulació.
    - Data de començament de la simulació, posterior (o igual) a la data de referència d'inici.
    - Data de final de simulació.
    - Granularitat de la simulació (segons, minuts, hores, etc.).
- L'usuari ha de poder llançar l'execució d'una simulació, i parar-la en qualsevol moment.
- S'han de mostrar a l'usuari gràfiques de tràfic per cada element i en temps real.
- L'usuari ha de poder configurar una pla de canvis, que és un conjunt de canvis en valors de certs paràmetres que ocorren en moments planificats de la simulació.

  Els paràmetres que es poden canviar són els següents:
    - Multiplicador d'usuaris del generador.
    - *Bitrate* de la cua.
    - Percentatges de tràfic que reben les sortides d'una cua.
- Els resultats de la simulació s'han de poder exportar per poder ser tractats posteriorment per software extern.
- El sistema ha d'estar preparat per permetre simulacions simultànies.

# 9 Arquitectura

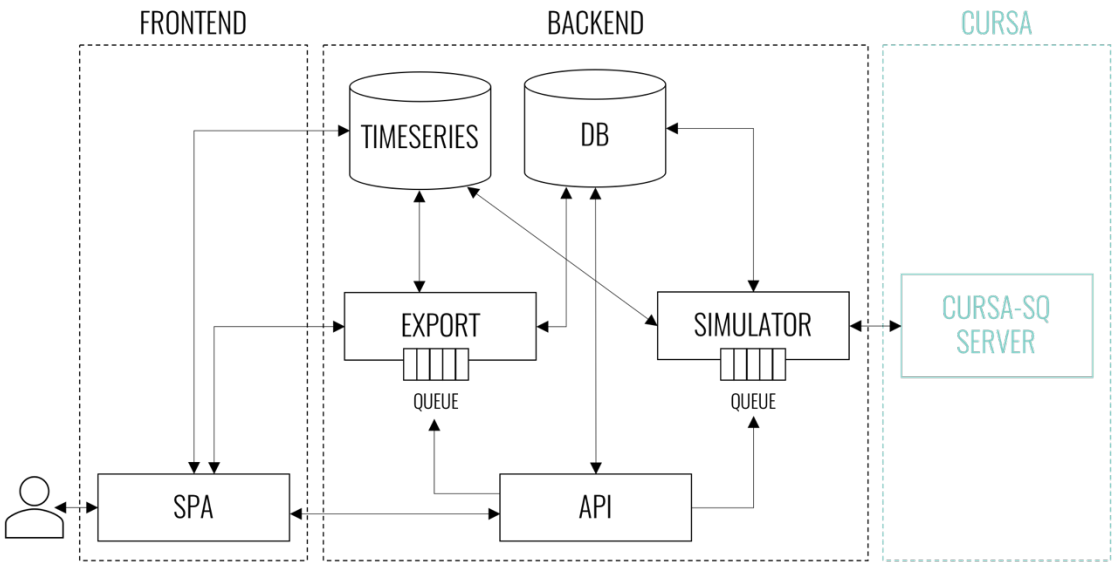


*Figura 7: Arquitectura del sistema*

L'arquitectura que té el sistema del treball està representada en la Figura 7. Està organitzada en tres blocs: un primer bloc de *frontend* que s'encarrega de controlar la interfície gràfica, un segon bloc de *backend* que gestionarà les dades i les execucions de les simulacions i un tercer bloc que ja està implementat i és el servidor de CURSA-SQ.

A continuació es detallen els diferents mòduls que hi ha a cada bloc.

## 9.1 Frontend

Com s'ha mencionat en apartats anteriors, el *frontend* és una *Single Page Application* desenvolupada amb *React*.

La Figura 8 mostra l'arquitectura que segueix la SPA. Es pot veure que *React* només s'encarrega de renderitzar les vistes mitjançant components, les demés funcionalitats s'han afegit amb llibreries externes.

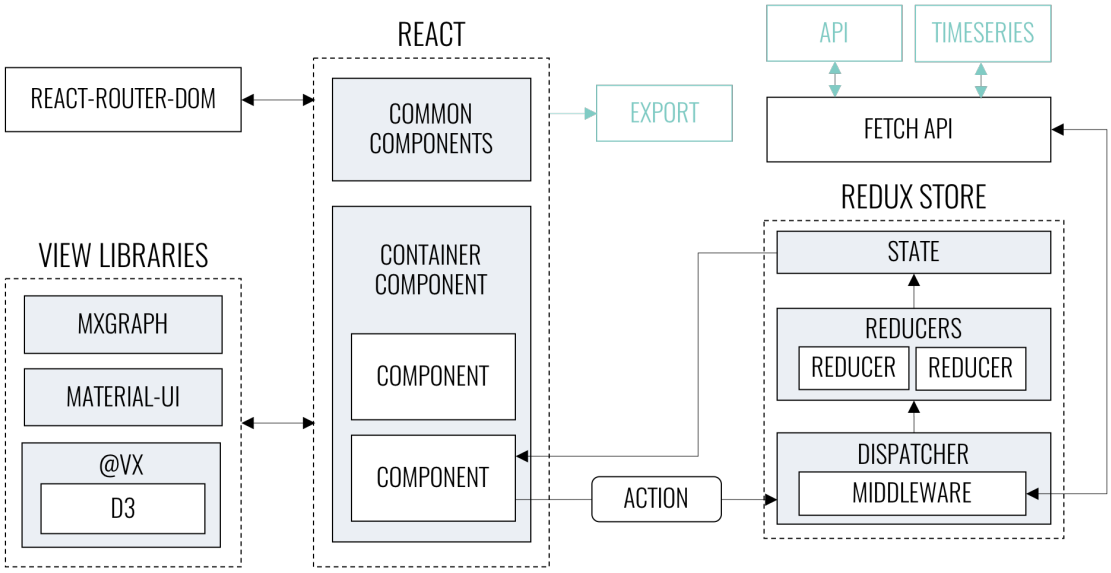


*Figura 8: Arquitectura de frontend*

### 9.1.1 Estat de l'aplicació

#### 9.1.1.1 Flux

L'estat global de l'aplicació s'ha construït seguint el patró *Flux*.

*Flux* és un patró arquitectònic per desenvolupar aplicacions *frontend* creat per Facebook. A diferència del patró MVC convencional, *Flux* proposa un flux de dades unidireccional.

Tal com mostra la Figura 9, els diferents components que formen part del patró són els següents:

- **Actions:** descriuen les accions que poden ocórrer a l'aplicació, és a dir, tots aquells esdeveniments susceptibles de generar canvis a l'estat.
- **Dispatcher:** mecanisme central que recull les *Actions* que es llancen, i avisa a cada *Store* per si han de reaccionar.
- **Store:** contenen el model que representa l'estat de l'aplicació, així com les lògiques per a reaccionar davant de les *Actions*.
- **View:** són les vistes les que demanen l'estat a les *Stores* i s'encarreguen de representar-les gràficament. Aquestes poden llançar *Actions* quan passin certs esdeveniments, com per exemple un clic a un botó.

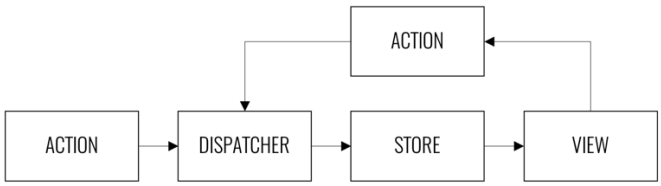


*Figura 9: Arquitectura Flux*

*Flux*, a més a més, proposa mecanismes per actualitzar cada *Store*: els *Reducers*. Un *Reducer* és una funció que rep com a paràmetre l'estat anterior i una *Action* que ha emès el *Dispatcher*, i retorna com queda l'estat després del canvi.

Fer-ho així permet que la programació de la lògica sigui totalment funcional, i, per tant, el resultat dels *reducers* sigui totalment determinista i molt fàcil de provar.

A més a més, obliga a que l'estat sigui immutable; és a dir, cada cop que es genera un canvi es crea un nou estat de zero, mai s'ha de modificar directament l'anterior. Això fa que els mecanismes de detecció de canvis de les vistes siguin molt més ràpids.

#### 9.1.1.2 Redux

La implementació concreta de *Flux* que s'ha fet servir és *Redux*, que és la més estesa.

*Redux* és una llibreria de codi obert inspirada en l'arquitectura *Flux* que permet gestionar l'estat d'aplicacions escrites en *Javascript*. Per a integrar-ho s'ha fet servir la llibreria *react-redux*, llibreria oficial de *React*.

Com a particularitats, *Redux* ofereix una única *Store* que es defineix mitjançant un *Reducer* i que pot ser resultat de combinar diferents *Reducers*.

A més a més, *Redux* permet configurar *Middlewares*, que són trossos de codi que s'executen en el moment en el que es llança una *Action* i poden fer accions amb ella abans que arribi al *Reducer*, com registrar informació, modificar l'acció o inclús llançar-ne de noves.

#### 9.1.1.3 Redux-Saga

El *Middleware* més important que s'ha fet servir és *Redux-Saga*. Una *saga* no és més que una funció que s'executa al rebre una *Action*, i que permet llançar altres *Actions* de forma asíncrona.

L'ús més important de les *Saga* és fer crides a la API, així com fer tasques periòdiques com el guardat automàtic.

### 9.1.2 react-router-dom

*React* en si no incorpora cap funcionalitat per gestionar l'enrutament de les aplicacions.

Per a fer-ho, s'ha fet servir la llibreria de react-router-dom, que permet definir les diferents rutes i quins contenidors es renderitzaran a cada ruta així com injectar als components mecanismes per canviar de ruta i saber els paràmetres de la ruta actual.

### 9.1.3 mxgraph

Com s'ha mencionat anteriorment, la llibreria escollida per als diagrames de xarxa és *mxgraph*. La llibreria no està plantejada per funcionar amb *React*, ja que funciona amb un estat intern.

El que s'ha fet és crear un component que guardi una instància de *mxgraph*, i establir una sincronització entre estats mitjançant dos mecanismes de detecció de canvis: un a nivell de propietats del component, i un altre a nivell de canvis interns del *mxgraph*.

El cicle de sincronització es pot observar a la Figura 10.

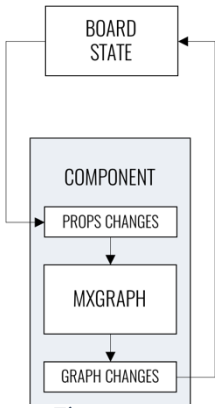


*Figura 10: Sincronització component i mxgraph*

### 9.1.4 Material-UI

Per no perdre gaire temps amb la implementació d'elements comuns en qualsevol UI s'ha fet ús de *Material-UI*, una llibreria que implementa els components definits per *Material Design* de Google fent servir *React*.

Els components que podem trobar en aquesta llibreria són botons, diàlegs, formularis, menús, taules, etc.

### 9.1.5 @VX i D3

Per a dibuixar les gràfiques de tràfic s'utilitza *D3.js*, una llibreria *Javascript* que permet la representació gràfica de dades en *HTML*, *SVG* i *CSS*. És una llibreria molt extensa i ofereix multitud d'eines per facilitar les tasques més comunes a l'hora de representar gràfics.

Per facilitar-ne el seu ús, s'ha fet us de *@VX*, una llibreria que integra *D3* en components de *React*.

## 9.2 Backend

El bloc de *Backend* es compon de diferents mòduls que compleixen diferents funcions. A continuació es detalla l'arquitectura de cada mòdul.

### 9.2.1 API

El mòdul d'API s'ha desenvolupat utilitzant el *NestJS*, un framework escrit amb *Typescript* que permet crear aplicacions de servidor. Internament utilitza *Express*, el servidor HTTP més comú en aplicacions de *Node.js*.

S'ha escollit aquest framework en comptes de fer servir pur *Express* perquè ofereix eines perquè el codi quedi ben ordenat, i per facilitar tasques comunes. A més a més, el fet que estigui escrit en *Typescript* per defecte facilita la configuració del projecte.

La Figura 11 mostra el funcionament del framework.

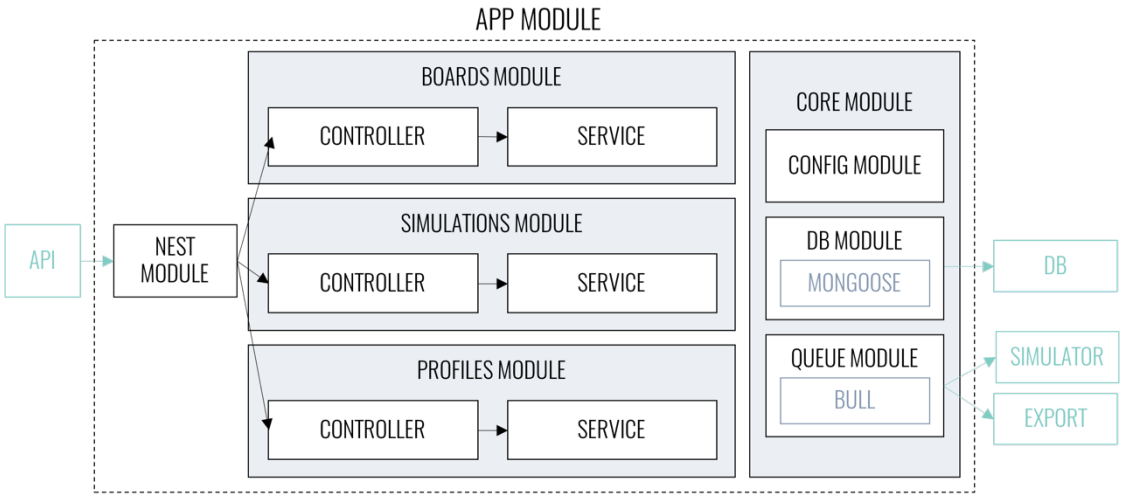


*Figura 11: Arquitectura del Backend*

L'arquitectura d'una aplicació *NestJS* es basa en mòduls. Tenim un mòdul global que inclou els demés mòduls (*AppModule*), i per cada funcionalitat es crea un altre mòdul.

A més a més, s'ha inclòs un mòdul de funcionalitats compartides (*CoreModule*) que inclou els següents mòduls:

- **ConfigModule**: permet obtenir la configuració de l'entorn
- **DBModule**: permet connectar amb la base de dades. Per dins fa ús de la llibreria *Mongoose*.
- **QueueModule**: permet enviar missatges a les cues. Per dins fa ús de la llibreria *Bull*.

Cada mòdul registra uns serveis que són utilitzats en altres parts mitjançant la injecció de dependències.

Els altres mòduls es creen seguint el model *REST*, on cada mòdul és un recurs, i el controlador és el que rep les peticions (*GET*, *POST*, *PUT* i *DELETE*) i fa les crides pertinents al servei, que conté la lògica.

### 9.2.2 Base de dades

El mòdul de base de dades conté tota la informació persistent de l'aplicació, com són els *boards*, les simulacions o els perfils.

Com que cadascun d'aquests elements es pot representar amb un model senzill de *JSON*, s'ha decidit de fer servir **MongoDB**, una base de dades no relacional i basada en documents. Actualment és una de les bases de dades de documents més populars, i ofereix llibreries per facilitar la integració en aplicacions *Javascript*.

### 9.2.3 Timeseries

Tot i que inicialment es va plantejar guardar els resultats de les simulacions en la mateixa base de dades dels *boards*, fent recerca vaig poder comprovar que aquest tipus de base de dades no estan pensades per aquest tipus d'operacions i dades.

Per això, s'ha fet servir una base de dades específicament pensada per dades que es produeixen en una línia de temps, el que es coneix com a **timeseries**.

L'opció escollida és **InfluxDB**, una base de dades relacional especialitzada en guardar dades de *timeseries* de forma eficient i escalable.

### 9.2.4 Simulador

El mòdul del simulador s'ha separat del mòdul d'API per dos motius:

1. És un mòdul suficientment complex a nivell de lògica com per voler separar-ho en un projecte de codi diferent.
2. El procés no ha de ser bloquejant, ja que mentre s'executa la simulació la API hauria de ser capaç de seguir responent a les peticions.

Per a fer-lo totalment independent i no bloquejant, s'ha definit una cua de peticions.

La cua està implementada amb **Bull**, una llibreria de codi obert que permet configurar cues ràpides i fiables en *Node.js* a partir d'una base de dades **Redis**.

L'arquitectura del simulador apareix descrita en la Figura 12. Mitjançant el processador de cues que utilitza *Bull* es crearia una tasca que s'executa de forma asíncrona.

Per dins, aquesta tasca faria us de connectors per les dependències externes: un client de *grpc* per a la connexió amb el servidor CURSA-SQ, *Mongoose* per a connectar-se amb la base de dades i el client d'*InfluxDB* per guardar els resultats.

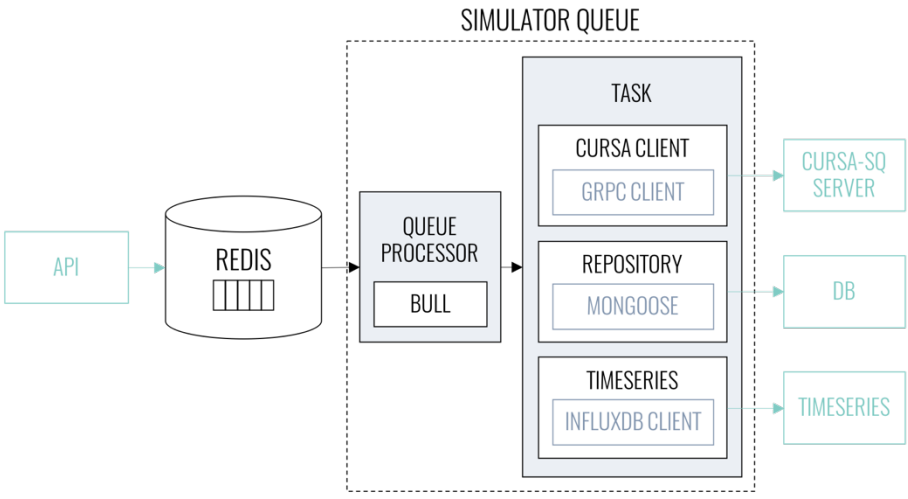


*Figura 12: Arquitectura del simulador*

### 9.2.5 Exportador

L'arquitectura de l'exportador és molt similar al simulador, com veiem a la Figura 13. La tasca d'exportar els resultats pot ser costosa, i per això s'ha implementat sobre una cua asíncrona.

Els elements diferenciadors són els següents:

- Es requereix d'un servidor de fitxers estàtics per proveir els fitxers comprimits. Aquest servidor és una instància senzilla d'*Express*.
- Per a comprimir els resultats, s'utilitza la dependència de **tar**.

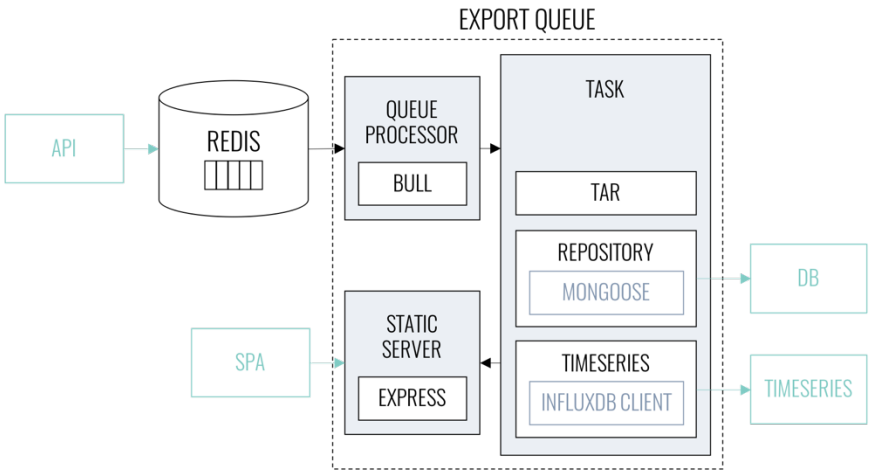


*Figura 13: Arquitectura de l'exportador*

## 9.3 Servidor CURSA-SQ

El servidor de CURSA-SQ que han preparat els directors és un servidor *gRPC* basat en *Python*.

S'anomena *RPC* als protocols que permeten a un programa executar un servei d'un altre programa en una altra màquina a través de la xarxa. *gRPC* és un framework que proveeix eines per a realitzar comunicacions *RPC* d'alt rendiment en qualsevol entorn.

Com veiem a la Figura 14, podem configurar el número de *threads* que volem que tingui el servidor per a les execucions de les simulacions.

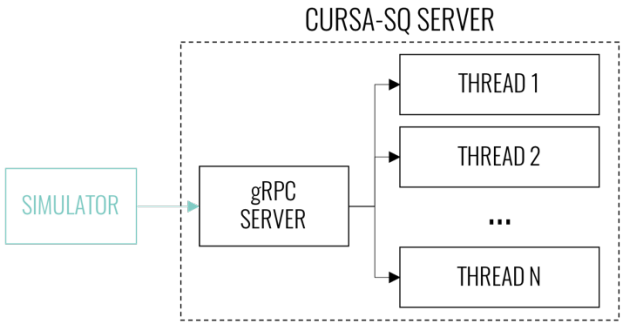


*Figura 14: Arquitectura del servidor CURSA-SQ*

## 9.4 Containers

L'arquitectura descrita en els apartats anteriors resulta força complexa ja que està formada per molts mòduls diferents, amb tecnologies diferents, i ens hem d'assegurar que funcioni en qualsevol màquina on instal·lem el projecte i que ho faci sempre de la mateixa forma.

Per a aconseguir-ho s'ha fet servir *Docker*.

### 9.4.1 Docker

**Docker** és un programa de virtualització basat en contenidors. Un contenidor és un espai virtual isolat que es diferencia de les màquines virtuals convencionals en el fet que fa servir un *kernel* compartit entre tots els contenidors, i, per tant, és molt més lleuger.

Els contenidors es creen executant imatges. Una imatge és un paquet executable que conté tot allò necessari per executar una aplicació.

Aquestes imatges les podem obtenir de dues maneres:

1. Per a les instàncies de programes existents, com són les bases de dades (*MongoDB*, *InfluxDB* i *redis*), es poden descarregar les imatges en la llibreria oficial d'imatges **Docker Hub**.
2. A cada repositori propi s'ha afegit un fitxer anomenat *Dockerfile* que conté les instruccions per construir la imatge i executar el contenidor.

   Aquestes imatges s'han compartit amb els directors utilitzant el repositori d'imatges privat *Treescale*.

### 9.4.2 Dockerfile

Els fitxers *Dockerfile* tenen dues parts: una primera part on s'especifica tot el que s'ha de fer per construir la imatge, i una segona part on es defineix què farà el contenidor un cop executat.

En els nostres repositoris, la primera part serà per preparar les dependències, copiar el codi i compilar-lo, així com per configurar les visibilitats, i la segona part executarà el nostre codi.

La Taula 10 mostra un exemple de *Dockerfile* per un repositori *Javascript*.

*Taula 10: Exemple de Dockerfile*

```
# Imatge base. Com és una aplicació Javascript, extenem de la
# imatge oficial de Node.js
#
# '8-alpine' és el tag de versió. Per mantenir la mida de l'imatge
# lo més petita possible, escollim el sistema operatiu Alpine.
FROM node:8-alpine

# Carpeta del contenidor on copiarem el codi
WORKDIR /app

# Copia els fitxers de dependències
COPY yarn.lock ./
COPY package.json ./

# Instal·la les dependències
RUN yarn install

# Copia la resta de fitxers, a excepció dels inclosos en el .dockerignore
COPY . .

# Instal·la les dependències
RUN yarn install

# Compila el codi
RUN yarn build

# El port 3000 serà visible des de fora del contenidor
EXPOSE 3000

# Comanda que s'executa al llançar el contenidor.
CMD [ "serve", "-s", "-l", "3000", "build" ]
```

### 9.4.3 Docker Compose

*Docker* ofereix un seguit de línies de comanda per descarregar imatges, executar/parar contenidors, i molt més. Donat el volum de contenidors que necessita el nostre projecte necessitem trobar una eina que ens faciliti aquestes tasques.

Per sort, *Docker* ja ofereix una eina amb aquest propòsit. **Docker Compose** és una eina que ens permet definir i executar aplicacions amb múltiples contenidors mitjançant un fitxer de configuració.

Aquest és un fitxer *YAML* i s'ha de dir *docker-compose.yml*. Aquest fitxer conté, per cada contenidor que volem llançar, els següents paràmetres:

- **image**: imatge del contenidor. Pot ser una URL a un repositori privat, o el nom de la imatge en el repositori de *Docker Hub*.
- **ports**: llistat de ports que es volen exposar.

  El format és: [port extern]:[port intern]
- **volumes**: enllaça carpetes externes amb carpetes internes al contenidor. D'aquesta manera, els continguts d'aquestes carpetes són persistents i no depenen del cicle de vida del contenidor.

  El format és: [carpeta externa]:[carpeta interna]
- **environment**: variables d'entorn.
- **depends_on**: llistat de contenidors als que s'ha d'esperar per començar a executar-se.

En la Taula 11 es pot veure un exemple de fitxer que s'ha fet servir com a "instal·lador" en aquest projecte.

*Taula 11: docker-compose.yml*

```
version: 3
services:
  mongodb:
    image: 'mongo'
    volumes:
      - ./db:/data/db
  influxdb:
    image: 'influxdb:alpine'
    ports:
      - 8086:8086
    volumes:
      - ./tsdb:/var/lib/influxdb
    environment: # Desactiva logs
      - INFLUXDB_META_LOGGING_ENABLED=false
      - INFLUXDB_DATA_QUERY_LOG_ENABLED=false
      - INFLUXDB_HTTP_LOG_ENABLED=false
  redis:
    image: 'redis:alpine'
    volumes:
      - ./redis:/data
```

```
cursa:
  image: 'repo.treescale.com/sferriz/tfg-cursa:latest'
simulator:
  image: 'repo.treescale.com/sferriz/tfg-simulator:latest'
  depends_on:
    - mongodb
    - redis
    - cursa
export:
  image: 'repo.treescale.com/sferriz/tfg-export:latest'
  ports:
    - 8080:8000
  depends_on:
    - mongodb
    - redis
backend:
  image: 'repo.treescale.com/sferriz/tfg-backend:latest'
  ports:
    - 8000:8000
  volumes:
    - ./logs:/app/logs
  depends_on:
    - mongodb
    - redis
    - simulator
frontend:
  image: 'repo.treescale.com/sferriz/tfg-frontend:latest'
  ports:
    - 3000:3000
```

# 10 Implementació

## 10.1 Frontend

En aquest apartat s'expliquen les diferents pantalles i funcionalitats que s'han desenvolupat en el *frontend*. L'idioma amb el que s'ha creat la interfície és l'anglès, per facilitar-ne l'ús a un públic més internacional.

### 10.1.1 Pantalla dashboard

La primera pantalla que se'ns presenta al accedir a l'aplicació és el *Dashboard*. Aquesta pantalla és el punt d'entrada que permet accedir a la resta de funcionalitats.

Com veiem a la Figura 15, actualment hi ha un espai en blanc a la dreta. Per limitacions temporals no s'ha inclòs res en aquesta part, però podria donar lloc al perfil de l'usuari, estadístiques d'ús, etc.

Al panell de l'esquerra hi veiem el llistat de *boards* que hi ha a la base de dades amb el nom i la data de modificació de cadascun. A més a més, tenim les opcions de crear, editar, importar i eliminar *boards*.

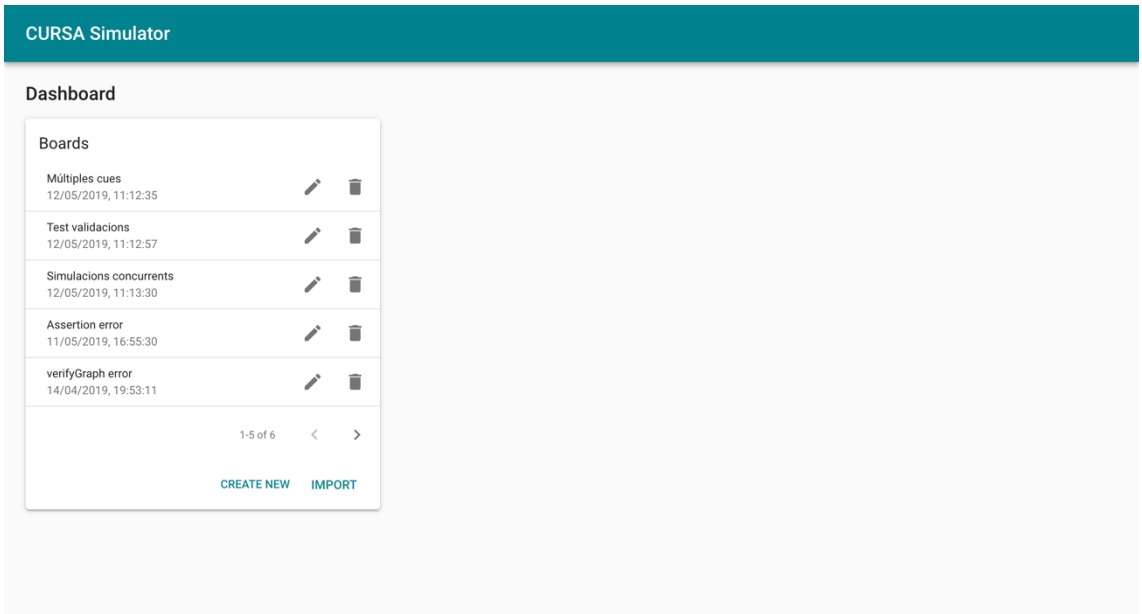


*Figura 15: Pantalla de Dashboard*

### 10.1.2 Pantalla d'edició de board

Al fer clic a editar o crear un nou *board*, s'accedeix a la pantalla d'edició del *board* (*veure* Figura 16).

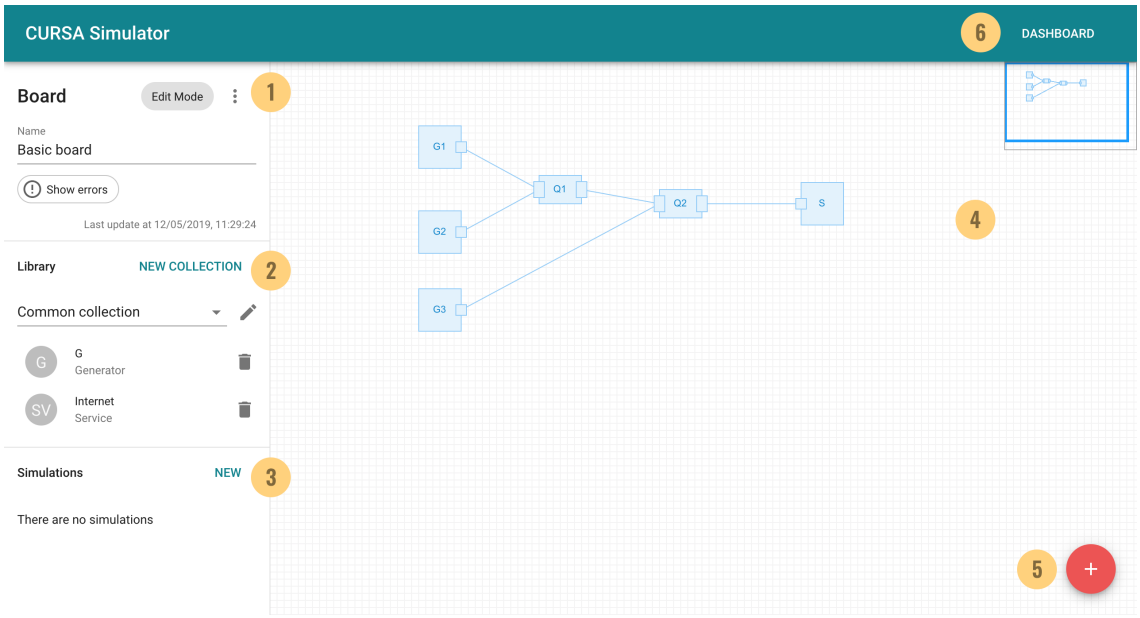


*Figura 16: Pantalla d'edició del board*

Els diferents apartats que apareixen són els següents:

### 1 Capçalera del *board*

Mostra les opcions bàsiques del *board*, com editar el nom, exportar o eliminar (aquestes últimes dues sota el menú desplegable). Ens mostra també l'última data de modificació com a comprovació de que el guardat automàtic funciona.

Ens mostra també l'estat del *board*. Existeixen dos estats:

- **Edit mode**: en aquest mode podem modificar el *board* sense cap restricció
- **Simulation mode**: s'entra en aquest mode al crear una simulació, i bloqueja l'edició del *board*. D'aquesta manera evitem haver d'invalidar els resultats d'una simulació al modificar el *board*.

Per últim, el botó de *Show errors* ens permet mostrar en temps real les validacions que es realitzen sobre el *board* prèvies a llançar una simulació.

### 2 Biblioteca de perfils

Permet navegar per les diferents col·leccions de perfils.

El primer selector permet seleccionar la col·lecció, que podem editar o eliminar mitjançant el botó del llapis.

Un cop seleccionada una col·lecció apareixen a sota tots els perfils que té. Si un perfil és de tipus generador o cua, fent clic s'afegirà automàticament al *board*. També podem eliminar un perfil d'una col·lecció amb el botó d'eliminar.

### 3 Simulacions

Mostra el llistat de simulacions que té configurades el *board*. Per cada simulació es mostra el nom, l'última data de modificació i un botó per eliminar-la. Fent clic a la simulació accedim a la pantalla de simulació.

### 4 Board

Interfície gràfica per editar els diferents elements del *board*. Cada vèrtex és una capsa amb un nom que es pot moure, redimensionar i connectar entre si mitjançant els ports. Fent clic sobre cada vèrtex s'entra en el mode d'edició del vèrtex.

En la part superior dreta hi ha un mini mapa del *board*, que permet moure el zoom en el cas que el *board* sigui més gran que la nostra pantalla.

En cas que el botó de *Show errors* estigui actiu podem veure els vèrtex que tenen errors en vermell, tal com mostra la Figura 17.

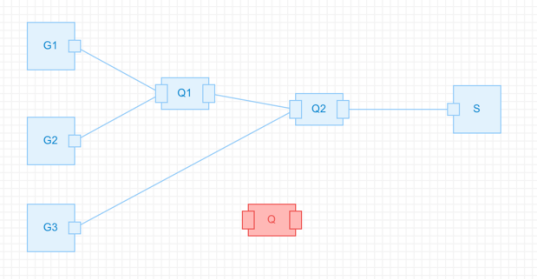


*Figura 17: Errors del board*

### 5 Afegir vèrtex

El botó d'afegir vèrtex permet afegir nous elements al *board* sense cap configuració prèvia.

Mostra el llistat de simulacions que té configurades el *board*. Per cada simulació es mostra el nom, l'última data de modificació i un botó per eliminar-la. Fent clic a la simulació accedim a la pantalla de simulació.

### 6 Menú

En el menú superior només ens apareix un enllaç per tornar al *Dashboard*.

### 10.1.3 Panell d'opcions dels generadors

S'accedeix fent clic sobre un generador existent en el *board*, o afegint-ne un de nou. Al ser el tipus de vèrtex amb la configuració més complicada algunes configuracions s'han hagut d'incloure en diàlegs.

Els diferents elements que apareixen en la Figura 18 són els següents:

#### 1 Capçalera del generador

Mostra les opcions bàsiques del generador. El menú té les següents opcions:

- Afegir a la biblioteca de perfils: ja sigui en una col·lecció existent o en una nova.
- Duplicar el generador
- Eliminar el generador

També podem modificar el nom, i el *bitrate* màxim.

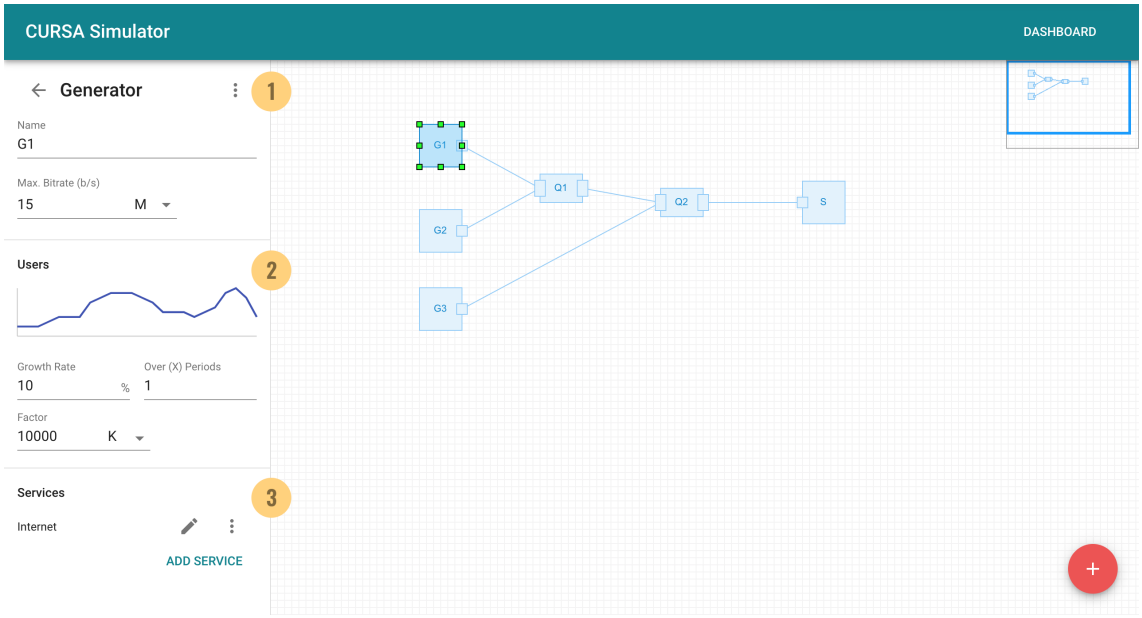


*Figura 18: Panell d'opcions dels generadors*

#### 2 Corba d'usuaris

Permet configurar la corba d'usuaris, que definirà el número d'usuaris que faran ús dels serveis d'un generador en cada unitat de temps.

El primer pas és configurar la corba mitjançant el diàleg de la Figura 19. En aquest, se'ns permet importar un *CSV* amb el següent format i sense capçalera:

<data>;<valor [0-1]>

Un cop importat el *CSV*, es poden fer modificacions sobre els valors directament en el diàleg, o exportant el fitxer i tornant-lo a importat modificar

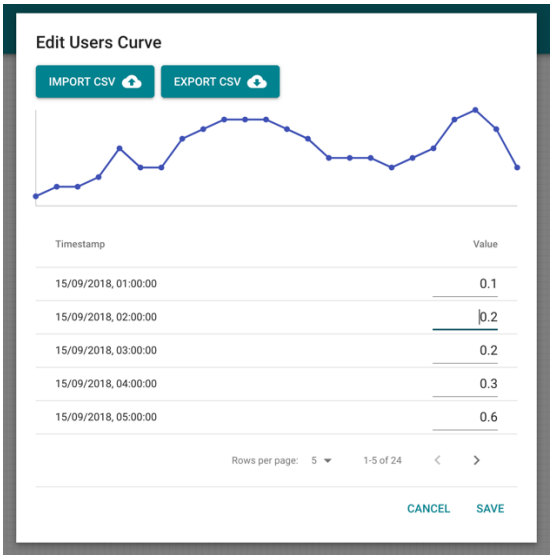


*Figura 19: Diàleg de corba d'usuaris*

Les dates que s'utilitzin en el *CSV* no són rellevants, el que importa és el temps que passa entre data i data, que és el que es prendrà com a referència a l'hora de simular els fluxos.

Un cop donada d'alta la corba podrem modificar els camps de *growth rate* i *factor*, mencionats anteriorment a l'apartat de requisits.

## 3 Serveis

Llistat de serveis configurats pel generador. Per cada servei tenim les opcions de modificar les característiques, afegir a la biblioteca de perfils o eliminar.

A l'afegir un nou servei s'obre un diàleg, tal com mostra la Figura 20.

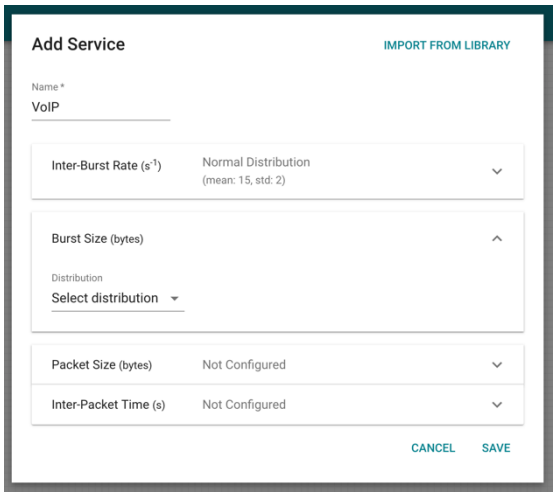


*Figura 20: Diàleg per crear un servei*

En aquest diàleg podem importar la configuració d'un servei existent de la biblioteca o editar directament les diferents configuracions. Per cada paràmetre de configuració s'ha de seleccionar una distribució estadística i configurar-ne els valors.

### 10.1.4 Panell d'opcions de les cues

Com es pot veure a la Figura 21, el panell d'opcions de les cues és força més senzill que el dels generadors.

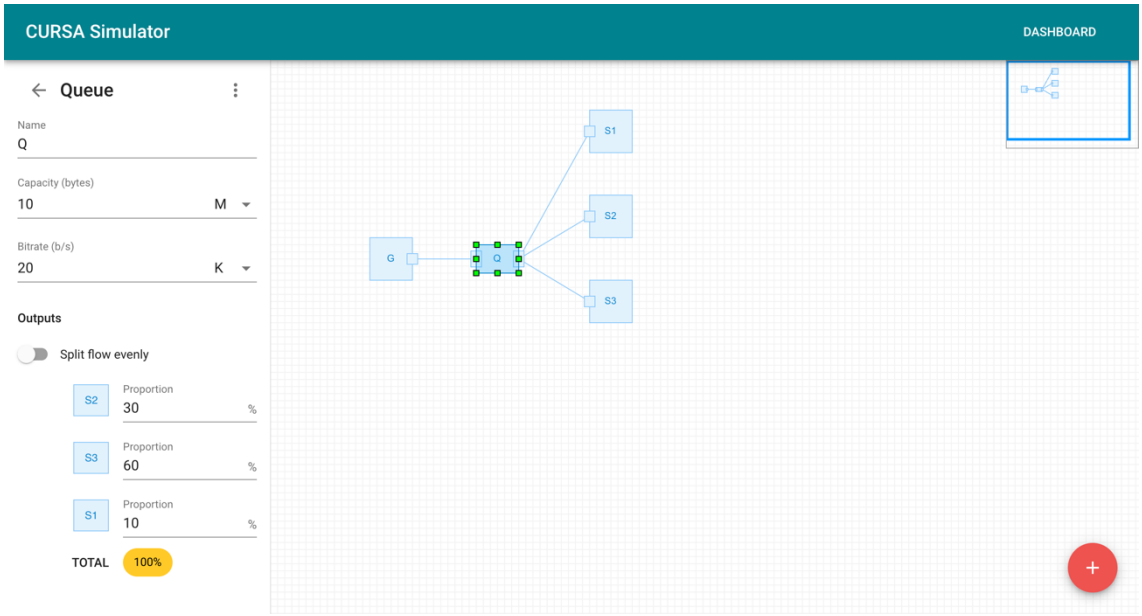


*Figura 21: Panell d'opcions de les cues*

La part superior és la part comuna dels vèrtexs que ens permet modificar el nom, afegir a la biblioteca de perfils, duplicar la cua o eliminar-la.

A continuació podem modificar els camps de capacitat de la cua i el *bitrate*.

Per últim, ens apareix un llistat de les connexions de sortida de la cua on podem modificar el percentatge de tràfic que anirà dirigit a cada sortida. Si volem que el percentatge sigui equitatiu per totes les sortides, es pot marcar el *checkbox*.

### 10.1.5 Panell d'opcions dels *sinks*

Per últim, el panell d'opcions dels *sinks* només permeten modificar el nom, duplicar o eliminar; no té més configuracions.

### 10.1.6 Pantalla de simulacions

S'accedeix fent clic sobre una simulació dins d'un *board*.

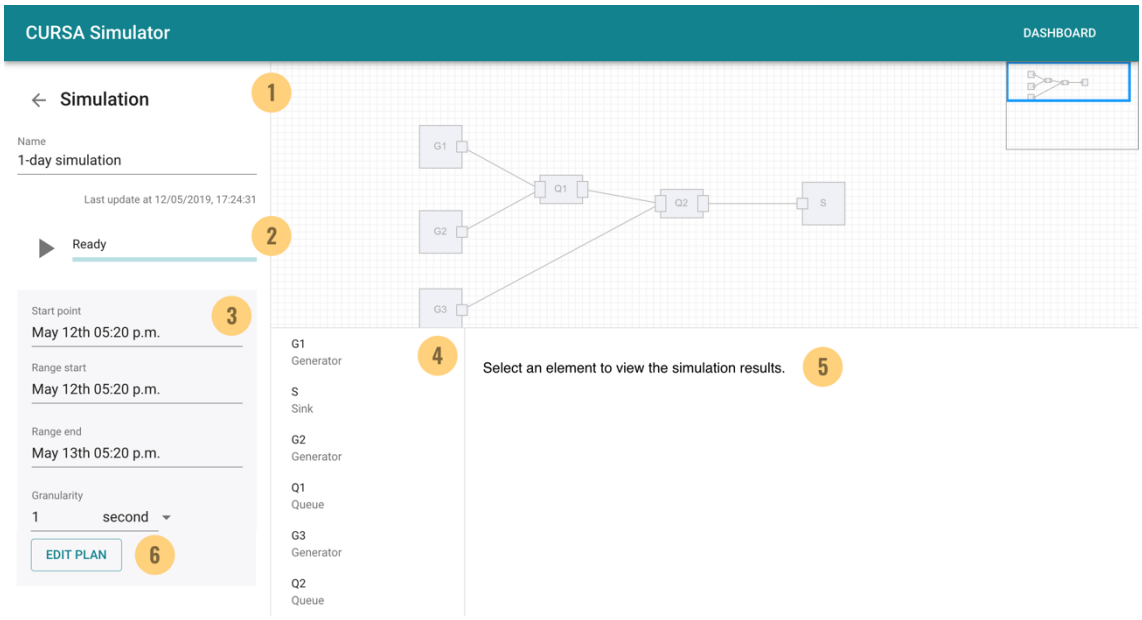


*Figura 22: Pantalla de simulacions*

Els apartats que es mostren en la Figura 22 són els següents:

**1 Capçalera de la simulació**

Mostra el nom de la simulació i l'última data de modificació. El botó de tornar enrere aniria a la pantalla del *board* al que pertany aquesta simulació.

**2 Controls d'execució**

Permeten controlar l'execució de la simulació. Els diferents estats que pot tenir aquesta secció són els següents:

- **Ready:** Llest per començar la simulació.
- **Running:** En execució, ens mostra el total completat (en %) i l'opció de parar la simulació.
- **Completed:** Simulació completada. Podem tornar-la a executar, o exportar-ne els resultats en un comprimit.
- **Error:** Ha ocorregut un error en la simulació i podem tornar a intentar executar-la.

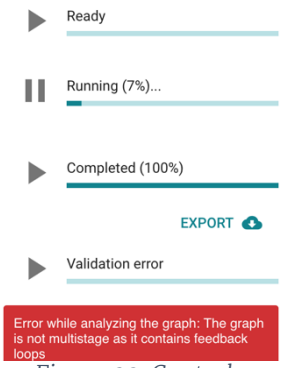


*Figura 23: Controls d'execució*

### 4 Llistat de vèrtexs

Es mostra un llistat de vèrtexs per a facilitar la navegació dels resultats de la simulació. Al fer clic sobre un vèrtex es marcarà l'equivalent en el *board*, que està desactivat. També podem fer clic directament sobre els vèrtexs del *board*.

### 5 Gràfiques de resultats

Un cop s'hagi seleccionat un vèrtex i la simulació hagi començat, es poden consultar en temps real els resultats de la simulació per a cada connexió del vèrtex.

La Figura 24 mostra un exemple de gràfica.

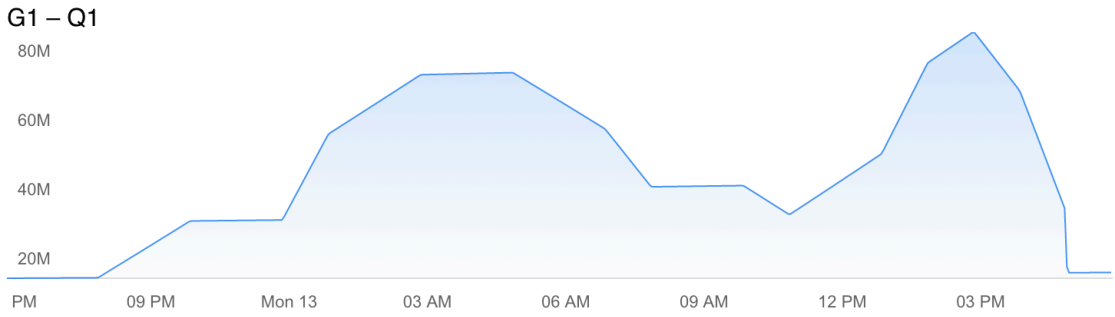


*Figura 24: Gràfica de resultats*

La part superior mostra el nom de la connexió, i al passar el cursor per sobre de la gràfica se'ns mostrarà el valor en cada instant de temps en un quadre de text.

Si fem clic en un instant de temps i arrosseguem el ratolí podem fer zoom en una zona concreta.

## 6 Pla de canvis

A l'apartat de pla de canvis podem planificar canvis en certs paràmetres del *board* perquè ocorrin en certs moments. D'aquesta manera podem simular una xarxa que es va transformant en el temps.

Per el moment, els paràmetres que es poden modificar són els següents:

- El multiplicador d'usuaris dels generadors.
- El *bitrate* de les cues.
- Les proporcions de sortida de les cues.

Al fer clic al botó es mostrarà el diàleg de la Figura 25, que ens permet anar afegint nous punts temporals on introduir aquests canvis.

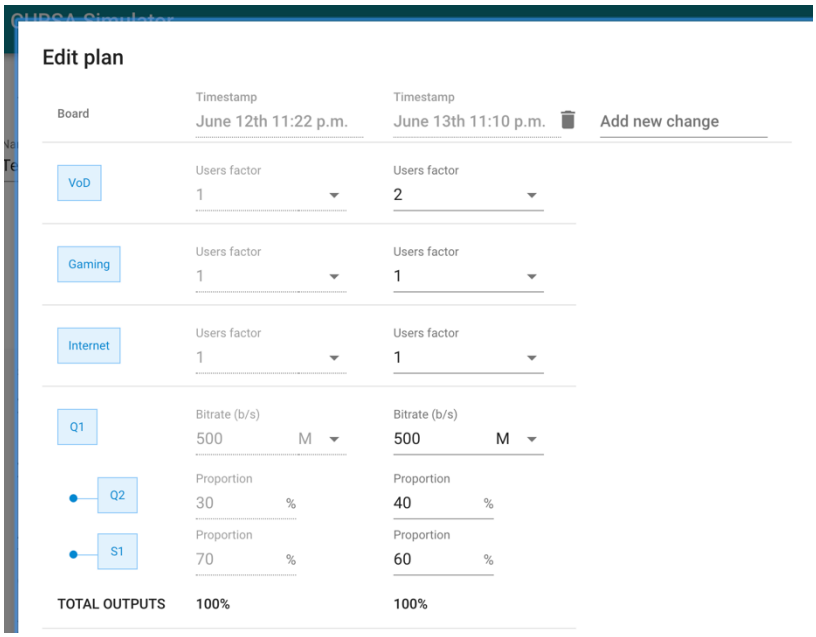


*Figura 25: Pla de canvis de simulació*

## 10.2 Backend

### 10.2.1 API

S'ha intentat seguir els principis REST a l'hora d'estructurar l'API. Les crides que ofereix l'API estan detallades en la següent taula.

**GET** /boards

Retorna tots els *boards* del sistema.

**Resposta**
Board[]

**GET** /boards/{ id }

Retorna un *board*.

| Resposta | Paràmetres | |
|---|---|---|
| Board | id | Id del board |

**POST** /boards

Crea un nou *board* i el retorna.

| Resposta | Body |
|---|---|
| Board | Board |

**PUT** /boards/{ id }

Actualitza les dades del *board* i el retorna actualitzat.

| Resposta | Body | Paràmetres | |
|---|---|---|---|
| Board | Board | id | Id del board |

**DELETE** /boards/{ id }

Elimina un *board*.

| Resposta | Paràmetres | |
|---|---|---|
| - | id | Id del board |

**GET** /simulations

Retorna totes les simulacions, amb l'opció de filtrar per *board*.

| Resposta | Paràmetres GET | |
|---|---|---|
| Simulation[] | boardId | [*opcional*] Id del board |

**GET** /simulations/{ id }

Retorna un simulació.

| Resposta | Paràmetres | |
|---|---|---|
| Simulation | id | Id de la simulació |

**POST** /simulations

Crea una nova simulació per a un *board* concret

| Resposta | Body | |
|---|---|---|
| Simulation | boardId | Id del board |

**PUT** /simulations/{ id }

Actualitza les dades de la simulació i la retorna actualitzada

| Resposta | Body | Paràmetres | |
|---|---|---|---|
| Simulation | Simulation | id | Id de la simulació |

**PUT** /simulations/run/{ id }

Actualitza les dades de la simulació i comença l'execució

| Resposta | Body | Paràmetres | |
|---|---|---|---|
| Simulation | Simulation | id | Id de la simulació |

**GET** /simulations/stop/{ id }

Para l'execució d'una simulació.

| Resposta | Paràmetres | |
|---|---|---|
| Simulation | id | Id de la simulació |

**GET** /simulations/export/{ id }

Comença el procés d'exportació de la simulació

| Resposta | Paràmetres | |
|---|---|---|
| Simulation | id | Id de la simulació |

**DELETE** /simulations/{ id }

Elimina una simulació.

| Resposta | Paràmetres | |
|---|---|---|
| - | id | Id de la simulació |

**GET** /profiles/collections

Retorna totes les col·leccions de perfils del sistema.

**Resposta**
ProfileCollection[]

**GET** /profiles/collections/{ id }

Retorna una col·lecció de perfils

| Resposta | Paràmetres | |
|---|---|---|
| ProfileCollection | id | Id de la col·lecció |

**POST** /profiles/collections

Crea un nova col·lecció de perfils i la retorna.

| Resposta | Body | |
|---|---|---|
| ProfileCollection | name | Nom de la col·lecció |

**PUT** /profiles/collections/{ id }

Actualitza una col·lecció de perfils i la retorna actualitzada.

| Resposta | Body | Paràmetres | |
|---|---|---|---|
| ProfileCollection | ProfileCollection | id | Id de la col·lecció |

**DELETE** /profiles/collections/{ id }

Elimina una col·lecció de perfils.

| Resposta | Paràmetres | |
|---|---|---|
| - | id | Id de la col·lecció |

**POST** /profiles/{ id }

Crea un nou perfil en una col·lecció nova o existent i retorna la col·lecció actualitzada.

| Resposta | Paràmetres | |
|---|---|---|
| ProfileCollection | collectionId | [*opcional*] Id de la col·lecció existent |
| | collectionName | [*opcional*] Nom de la nova col·lecció |
| | profile | Dades del perfil |

**DELETE** /profiles/{ id }

Elimina un perfil de la seva col·lecció i retorna la col·lecció actualitzada.

| Resposta | Paràmetres | | Body | |
|---|---|---|---|---|
| ProfileCollection | id | Id del perfil | collectionId | Id de la col·lecció |

### 10.2.2 Esquema de la base de dades

Com s'ha mencionat anteriorment, *MongoDB* és una base de dades basada en documents. S'han creat tres col·leccions de documents per aquest projecte: *Boards*, *Simulations* i *ProfileCollections*.

#### 10.2.2.1 Col·lecció de Boards

En la Taula 12 es detalla el model de la primera col·lecció, els *Boards*.

*Taula 12: Model de Board*

```
{
  _id: string, // Id de MongoDB
  name: string, // Nom
  width: number, // Amplada
  height: number, // Altura
  vertices: Vertex[], // Vector de vèrtexs
  edges: Edge[], // Vector d'enllaços
  createdAt: Date, // Data de creació
  updatedAt: Date, // Data d'actualització
}
```

Entenem el *Board* com un graf on els diferents elements són vèrtexs connectats entre si. En la nostra base de dades, aquests elements els anomenem *Vertex* (Taula 13) i les connexions les anomenem *Edge* (Taula 14).

*Taula 13: Model de Vertex*

```
{
  id: string, // Id
  type: string, // 'Generator' | 'Sink' | 'Queue' | 'Aggregator'
  name: string, // Nom
  position: {
    x: number, // Coordenada x
    y: number, // Coordenada y
  },
  width: number, // Amplada
  height: number, // Altura
  ports: Port[], // Vector de ports
}
```

*Taula 14: Model d'Edge*

```
{
  id: string, // Id
  from: string, // Id del port de sortida
  to: string, // Id del port d'arribada
```

```
}
```

A més a més, els vèrtexs poden tenir diferents models en funció del valor del camp *type*. Alguns, com el *Sink* o els *Aggregators*, no necessiten dades addicionals a les que venen amb el model de *Vertex*. Els models de *Generator* (Taula 15) i *Queue* (Taula 16), en canvi, afegeixen dades pròpies.

*Taula 15: Model de Generator*

```
{
  ..., // Herència de Vertex
  maxBitrate: ValueMultiplier, // Bitrate màxim
  services: Service[], // Vector de serveis
  users: { // Configuració d'usuaris
    factor: ValueMultiplier, // Multiplicador d'usuaris
    growthRate: number, // Ratio de creixement
    growthOverPeriods: number, // Període de creixement (en núm. períodes)
    curve: {
      points: [
        {
          value: number, // Valor proporcional d'usuaris (0-1)...
          timestamp: Date, // ... en un moment donat
        }
      ]
    }
  },
}
```

*Taula 16: Model de Queue*

```
{
  ..., // Herència de Vertex
  k: ValueMultiplier, // Capacitat (bytes)
  mu: ValueMultiplier, // Bitrate (b/s)
}
```

Per guardar valors que tenen un rang de valors molt gran, s'ha creat el model de *ValueMultiplier* (Taula 17), que consta d'un valor i d'un multiplicador. El valor s'obtindrà en multiplicar el valor *n* pel multiplicador en funció de la base escollida (veure llistat d'equivalències a la Taula 18).

*Taula 17: Model de ValueMultiplier*

```
{
  value: number, // Valor
  multiplier: string, // null | 'k' | 'm' | 'g' | 't'
}
```

*Taula 18: Equivalència de multiplicadors*

| Multiplicador | Valor base 2 | Valor base 10 |
|---|---|---|
| null | $n$ | $n$ |
| k | $n \cdot 2^{10}$ | $n \cdot 10^{3}$ |
| m | $n \cdot 2^{20}$ | $n \cdot 10^{6}$ |
| g | $n \cdot 2^{30}$ | $n \cdot 10^{9}$ |
| t | $n \cdot 2^{40}$ | $n \cdot 10^{12}$ |

Cada generador té un llistat de *Services* que permeten parametritzar la generació de fluxos de dades. El model de *Service* es pot veure a la Taula 19.

*Taula 19: Model de Service*

```
{
  id: string, // Id
  name: string, // Nom
  interBurstRate: Distribution, // Inter-Burst Rate (s⁻¹)
  burstSize: Distribution, // Burst Size (bytes)
  packetSize: Distribution, // Packet Size (bytes)
  interPacketTime: Distribution, // Inter-Packet Time (s)
}
```

Cada paràmetre de configuració del *Service* obté els valors a partir d'una distribució estadística. Aquestes distribucions es representen amb el model de *Distribution* (Taula 20).

*Taula 20: Model de Distribution*

```
{
  dist: string, // 'normal' | 'constant' | 'weibull' | 'exponential'
  params: {
    mean: number, // Mitjana (distribucions 'normal' i 'constant')
```

```
    std: number, // Desviació típica (distribució 'normal')
    shape: number, // Forma (distribució 'weibull')
    scale: number, // Escala (distribució 'weibull')
    lambda: number, // Lambda (distribució 'exponencial')
  }
}
```

#### 10.2.2.2 Col·lecció de Simulations

En aquesta col·lecció es guarda la configuració de la simulació i el progrés dels processos asíncrons, com l'execució de la simulació o l'exportació. No es guarden, però, els resultats de la simulació, ja que aquests van en una base de dades especialitzada.

El model que conté aquesta simulació està definit a la Taula 21.

*Taula 21: Model de Simulation*

```
{
  _id: string, // Id de MongoDB
  name: string, // Nom
  boardId: string, // Id del Board al que pertany
  startTimestamp: number, // Data de començament de la simulació
  rangeStartTimestamp: number, // Data a partir de la qual es simulen dades
  rangeEndTimestamp: number, // Data fins la qual es simulen dades
  granularityUnit: string, // 'second' | 'minute' | 'day' | 'hour'
  granularityValue: number, // Valor de granularitat
  state: string, // 'ready' | 'invalid' | 'running' | 'completed' | 'failed'
  progress: number, // Percentatge de simulació completat
  jobId: string, // Id de la tasca de simulació a la cua
  error: string, // Error de simulació
  export: {
    url: string, // Url per descarregar el fitxer exportat
    progress: number, // Percentatge d'exportació completat
  },
  createdAt: Date, // Data de creació
  updatedAt: Date, // Data d'actualització
}
```

#### 10.2.2.3 Col·lecció de ProfileCollections

Finalment, la col·lecció de *ProfileCollections* conté biblioteques d'elements reutilitzables. Per tant, els models que s'utilitzen són els mateixos que hem vist en els apartats anteriors.

Cada col·lecció conté un conjunt de perfils, tal com veiem a la Taula 22.

*Taula 22: Model de ProfileCollection*

```
{
  _id: string, // Id de MongoDB
  name: string, // Nom
  profiles: Profile[], // Vector de perfils
  createdAt: Date, // Data de creació
  updatedAt: Date, // Data d'actualització
}
```

El camp *type* de la Taula 23 especifica quina mena d'objecte representa aquell perfil, ja sigui un *Generator* o una *Queue*.

*Taula 23: Model de Profile*

```
{
  id: string, // Id
  type: string, // 'Generator' | 'Queue'
  data: object, // Dades del perfil
}
```

### 10.2.3 Timeseries

Com s'ha mencionat anteriorment, *InfluxDB* és una base de dades relacional especialitzada en guardar informació en format *timeseries*.

Cada taula que es genera dins d'una base de dades *InfluxDB* s'anomena *measurement*. Aquesta taula tindrà una fila per cada mesura en un cert instant de temps, i pot tenir tres tipus de columnes:

- **Timestamp**: moment en que s'ha pres la mesura.
- **Fields**: representen allò que es vol mesurar. Els valors seran numèrics.
- **Tags**: informació addicional a la mesura. A diferència dels *fields*, aquesta informació s'indexa, i per tant fan que les consultes siguin molt més ràpides.

Per el moment s'ha creat measurement (**simulation**) amb un sol *field*: la quantitat de tràfic que s'observa en un instant de temps (**flow**).

En quant a tags, se n'han creat dos: un per identificar la simulació a la qual pertany (**simulation**), i un altre per identificar l'*edge* sobre el qual estem prenent la mesura (**edge**).

Així, per obtenir les dades de sortida d'un cua haurem de fer una consulta a la base de dades amb l'id de simulació i l'id de l'*edge* que connecta la sortida de la cua amb un altre *vertex*.

A més a més, per limitar el nombre de dades que rebem, *InfluxDB* ofereix eines per agrupar resultats en grups de temps, així com per limitar l'interval de resultats.

En el nostre cas, per cada gràfic que es vol generar s'estableix l'interval de resultats, pren una freqüència de mostra i es demana la mitjana de cada grup de mesures. Per acabar, s'utilitza interpolació lineal per donar sensació de continuïtat a la gràfica.

La consulta resultant es mostra en la Taula 24.

*Taula 24: Consulta d'InfluxDB per al tràfic observat en un Edge en un intèrval de temps*

```
SELECT MEAN(flow) as flow FROM simulation
WHERE simulation = [SIMULATION_ID] AND edge = [EDGE_ID]
AND time >= [INTERVAL_START]ms AND time >= [INTERVAL_END]ms
GROUP BY time([FREQUENCY]ms) fill(linear)
```

### 10.2.4 Simulador

La tasca que executa el simulador quan rep una nova petició a la cua té dues fases: una de validació i una altra d'execució.

La validació es fa mitjançant la crida al servidor CURSA-SQ de *validateGraph*, que a partir d'un llistat de nodes ens retorna un llistat de *stages* o un error si el graf està mal format.

Cada *stage* ens indica els nodes que es poden simular a la vegada, és a dir, que no tenen dependència entre ells. A més a més, el llistat de *stages* està ordenat de forma que un *stage* necessita els resultats dels anteriors per a poder-se simular.

La Figura 26 mostra mitjançant un exemple com es formen els *stages* d'un *board*.

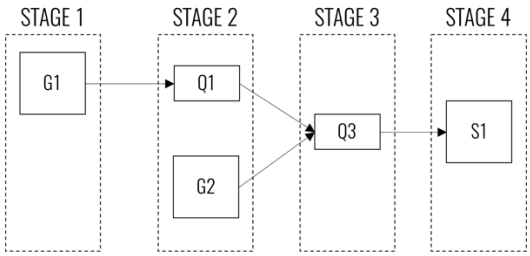


*Figura 26: Exemple de stages en un board*

Un cop tenim els *stages*, comença el procés de simulació.

Per evitar excessives crides al servidor CURSA-SQ, aquest permet simular múltiples vegades en una sola crida enviant grups d'inputs per cada node. Aquest grup d'inputs és el que anomenem *chunk*. Aquest valor és un valor fixe definit després de múltiples proves de rendiment sobre el servidor CURSA-SQ, i és de 30 valors per *chunk*.

Per a enviar dades en *chunk*, el primer pas que fa l'algoritme és dividir el període de simulació en intervals de temps que anomenem *steps*. La mida de l'*step* ve donada per la següent equació:

$$step = chunkSize * granularity$$

A cada *step* es simulen tots els *stages* enviant al servidor CURSA-SQ un *chunk* d'entrades per cada *stage* i rebent un *chunk* de sortides. Aquest *chunk* de sortides serveix com a grup de valors d'entrada per al següent *stage*.

Un cop simulat un *step*, es guarden les dades a *InfluxDB* i s'actualitza el progrés de la simulació a *MongoDB*.

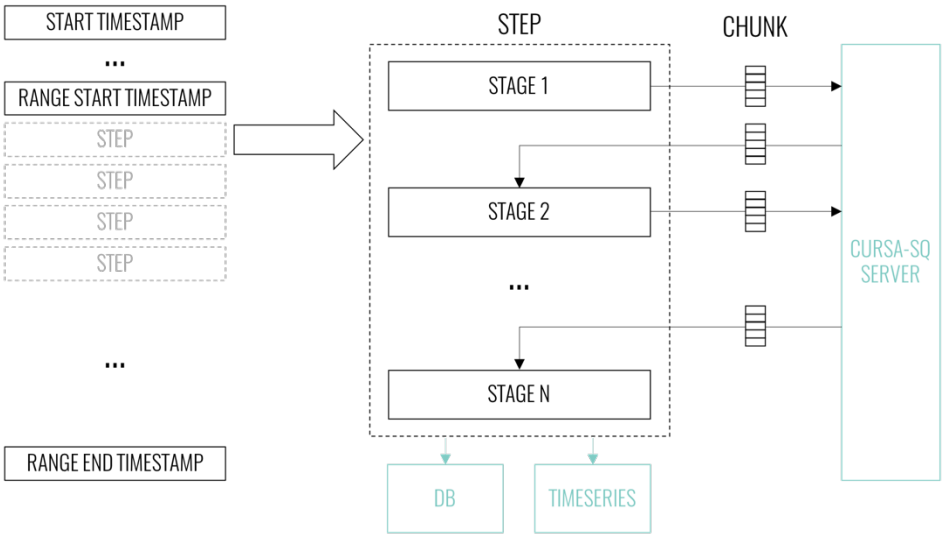


*Figura 27: Algoritme del simulador*

Per parar l'execució d'una simulació l'algoritme va realitzant comprovacions periòdiques de l'estat de la simulació a la base de dades.

### 10.2.5 Exportador

L'exportador genera un fitxer comprimit *.tar.gz* on els continguts són un fitxer *.csv* amb els resultats exportats d'*InfluxDB* per cada *edge*.

Ja que la API HTTP per obtenir dades d'*InfluxDB* falla quan les consultes són excessivament grans, s'ha limitat el nombre de resultats per consulta a 100.000.

A mesura que es van generant els fitxers *.csv* es va actualitzant el progrés a *MongoDB*, i, un cop acabat, es guarda la *URL* del fitxer comprimit perquè el *frontend* hi pugui accedir.

## 10.3 Servidor CURSA-SQ

El servidor CURSA-SQ ofereix tres crides *RPC*:

| Authenticate |
| --- |
| Donades les credencials d'un usuari, retorna un *token* que permet realitzar la resta de crides del servidor. |
| **VerifyGraph** |
| Verifica que el graf de xarxa està ben format, i retorna els grups de nodes que es poden simular a la vegada (*stages*). |
| **SimulateStage** |
| Donat un llistat de nodes (*stage*) amb les seves configuracions i estats retorna els fluxos de sortida de cada node, així com la quantitat de bits a la cua o descartats. |

Aquestes són les crides que s'han d'executar per ordre per a simular una xarxa, tal i com s'ha mencionat en la implementació del simulador.

# 11 Validació

La primera fase de validació s'ha fet a nivell d'usabilitat. Aquesta fase ha estat present en tot el procés de desenvolupament. Després de cada *sprint*, els directors del projecte han anat validant les funcionalitats desenvolupades en les tasques de l'*sprint*, i aquestes s'han anat comentant a la següent reunió.

Un cop acabat el projecte s'ha dedicat un període de proves per revisar que la integració de totes les funcionalitats és la correcta.

Les següents fases tenen com a objectiu validar que els resultats obtinguts en les simulacions mitjançant la nostra interfície són correctes i s'executen en el temps esperat.

Per validar els resultats obtinguts es comparen les dades generades per la nostra interfície amb una implementació existent de CURSA-SQ en *MATLAB*. Aquestes proves s'han anat realitzant durant tot el procés de desenvolupament per provar els diferents elements (generadors, cues i *sinks*). En els següents apartats s'inclou una petita mostra.

Les proves s'han realitzat amb una màquina amb les següents prestacions:

- Processador Intel i7-4790K
- 16 GB de RAM
- Disc dur Crucial MX500 amb velocitat de lectures/escriptures seqüencials 560/510 MB/s i aleatòries 95/90 K IOPS.
- Sistema operatiu Ubuntu 16.04.3 LTS

## 11.1 Generadors

El primer escenari de validació (Figura 28) consta de tres generadors, representant els tres serveis de dades que l'article [1] pren com a referència per definir el seu model d'anàlisi: *VoD*, *Gaming* i *Internet*.

Les característiques d'aquest escenari són les següents:

- Interval de simulació d'1 dia amb granularitat d'1 segon.
- Cada generador està connectat a un *sink* a través d'una cua.
- Cada generador té una corba d'usuaris constant, de manera que el tràfic de cada generador és en mitjana un 85% de la capacitat de la cua.

El que es pretén comprovar en aquest escenari és que els fluxos de dades que surten dels generadors són correctes, així com comprovar que a causa de la variabilitat del tràfic generat pels generadors les cues es van omplint ocasionalment.

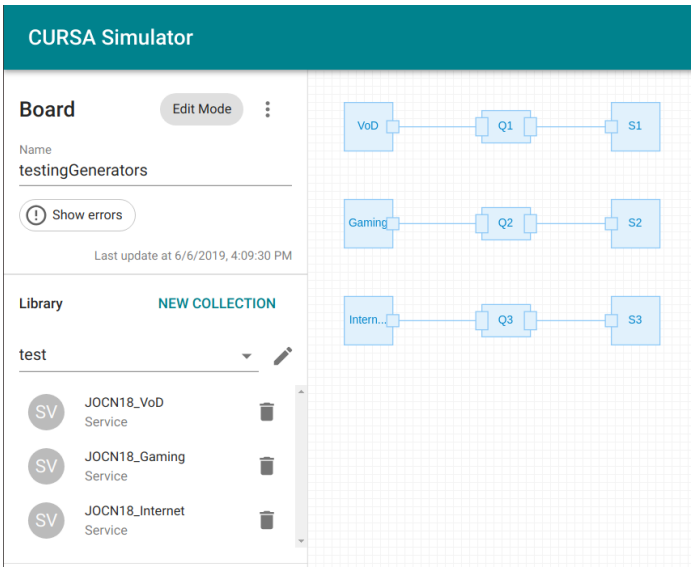


*Figura 28: Configuració de la prova de generadors*

Els resultats de la simulació es mostren en les següents figures. Aquestes mostren dos resultats:

- L'entrada de dades de cada cua observada en la simulació (*input_measured*) comparat amb l'entrada de dades esperada (*input_expected*).
- La sortida de dades de cada cua observada en la simulació (*output_measured*) comparat amb la sortida de dades esperada (*output_expected*).

Com que la generació de fluxos es calcula en funció de distribucions estadístiques i hi ha un component aleatori, el que s'ha fet és comparar la freqüència (*frequency*) amb la qual s'observa una certa taxa de bits (*bitrate*).

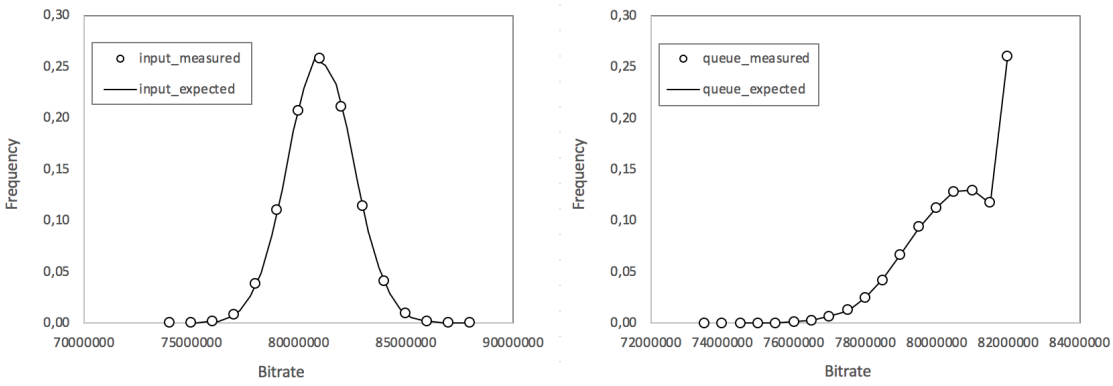


*Figura 29: Comparativa d'entrades i sortides de la cua de VoD*

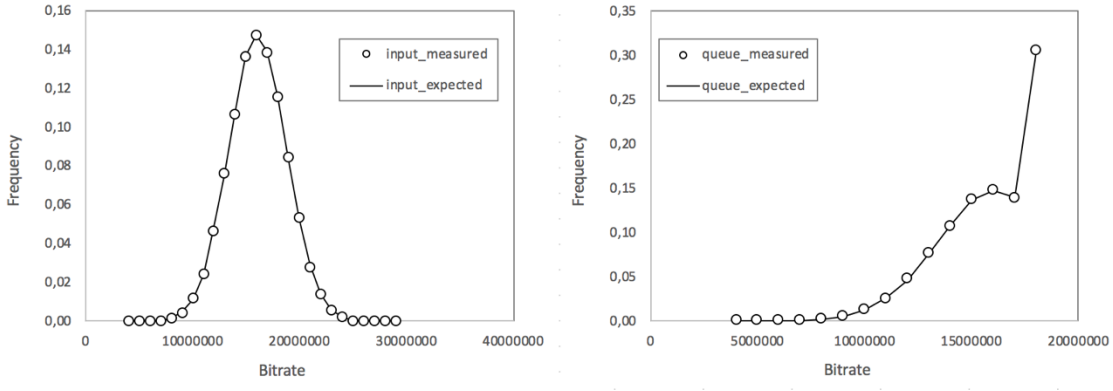


*Figura 30: Comparativa d'entrades i sortides de la cua de Gaming*

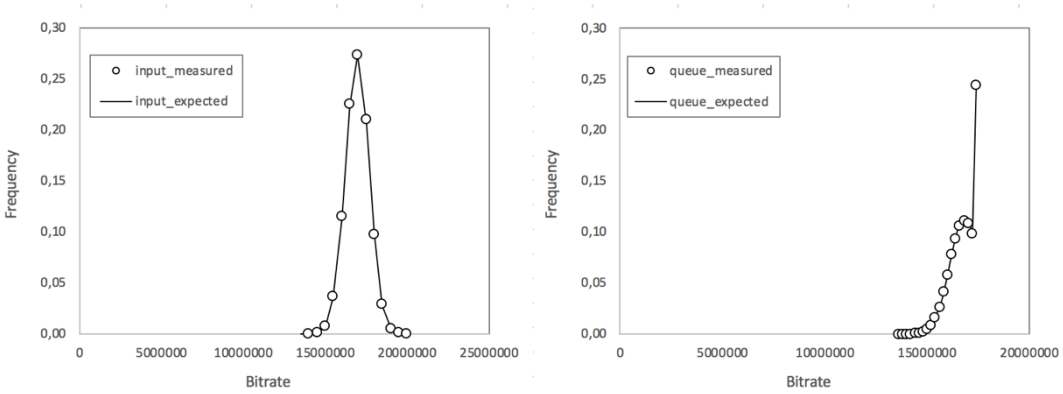


*Figura 31: Comparativa d'entrades i sortides de la cua d'Internet*

Com veiem en les gràfiques anteriors, en les tres cues el flux d'entrada i de sortida generat a través de la nostra interfície compleix amb els valors esperats.

A més a més, veiem que en les sortides de les cues hi ha sempre un pic de freqüència a partir d'un cert *bitrate* que es correspon als valors que emeten les cues quan estan saturades. Amb això es pot comprovar que la variabilitat de les sortides dels generadors provoquen que les cues se saturin.

## 11.2 Cues

Per aquesta prova s'ha definit un nou escenari, que es mostra en la Figura 32. Aquest consta dels mateixos tres generadors de l'escenari anterior però aquest cop connectats amb diferents cues.

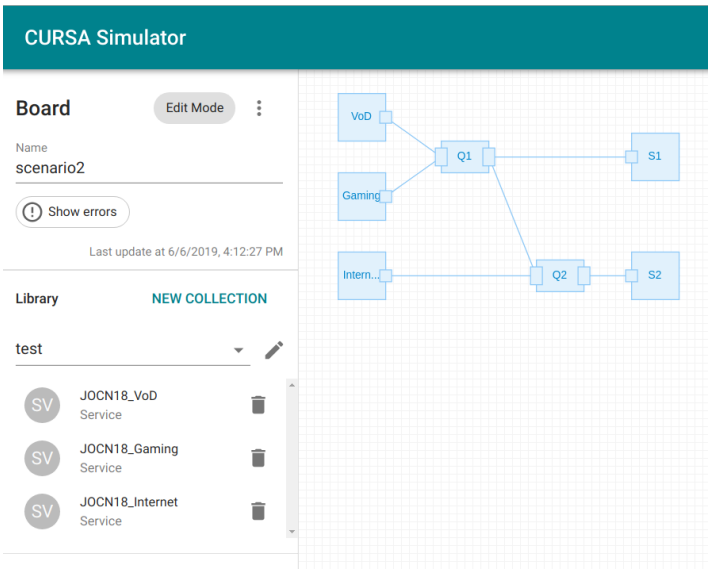


*Figura 32: Configuració de l'escenari 2*

Amb aquesta disposició es posen a prova les següents característiques de les cues:

- La capacitat d'agregació d'entrades. La cua *Q1* agregarà les entrades dels generadors *VoD* i *Gaming*.
- La saturació de les cues mitjançant un creixement diari d'usuaris del 20% als generadors. Amb aquestes dades la previsió indica que la cua *Q1* s'hauria de saturar sobre el dia 20-25 i que la cua Q2 s'hauria de saturar sobre el dia 35-40.
- La capacitat de separar el flux de dades de sortida en diferents connexions, cadascuna amb una proporció de dades. La cua *Q1* separa el tràfic en un 70% per al *S1* i un 30% per a Q2.

La simulació té una durada de 40 dies amb una granularitat d'1h.

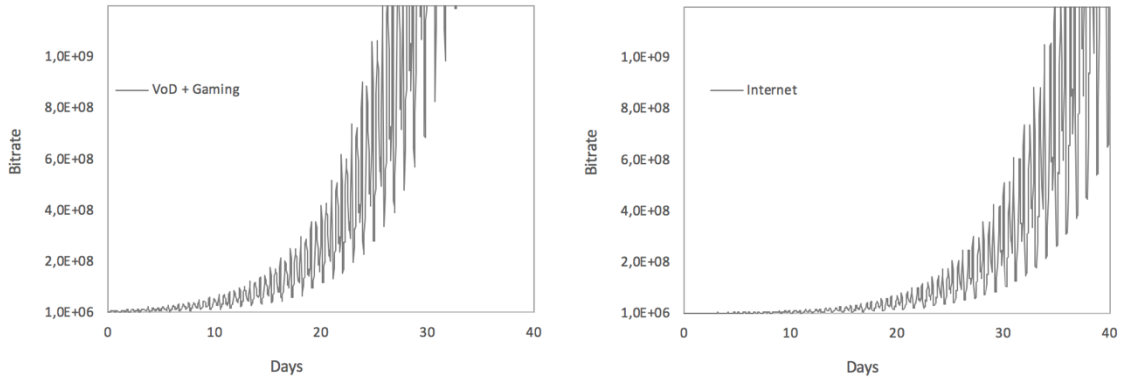


*Figura 33: Sortides dels generadors de l'escenari 2*

En les gràfiques de la Figura 33 veiem com el creixement d'usuaris configurat fa que el flux de dades vagi creixent, de manera que en algun punt les cues a les que estan connectats se saturaran. S'ha validat que el flux d'entrada observat a la *Q1* sigui la suma de les entrades de *VoD* i *Gaming*.

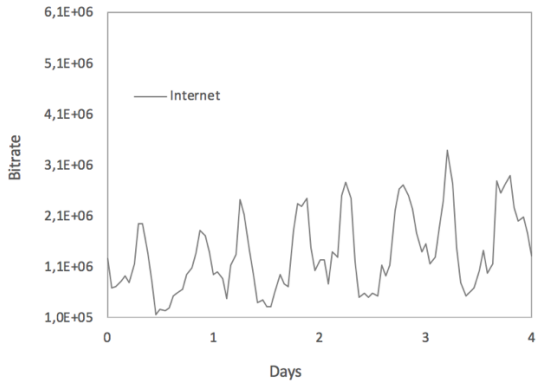


*Figura 34: Sortida del generador d'Internet durant els primers dies*

Ampliant una mica la gràfica de sortida d'un generador es pot observar el perfil diari d'usuaris. En la Figura 34 es pot veure com durant els primers 4 dies es reprodueixen els mateixos pics de dades donats per la corba d'usuaris, i també com els pics són cada dia més alts degut al creixement diari del 20%.

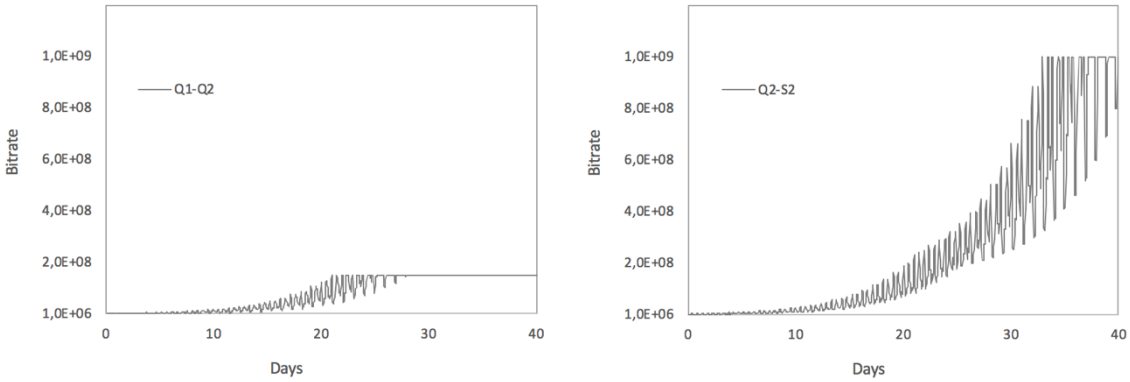


*Figura 35: Sortides de les cues de l'escenari 2*

En les gràfiques de la Figura 35 veiem la saturació de les cues. Es pot observar que els intervals estimats de saturació es compleixen en les dades obtingudes.

A més a més, es pot comprovar que la quantitat de dades que surt per l'enllaç *Q1-Q2* un cop saturada la cua (150 mb/s) és efectivament un 30% del *bitrate* de la *Q1* (500 mb/s), validant així les proporcions de sortida de les cues.

## 11.3 Rendiment

Per aquesta prova s'ha aprofitat l'escenari de la prova anterior, ja que és un escenari molt semblant al que es pren com a referència a l'article [1].

En una primera prova s'ha definit un interval de simulació de 24 hores i s'ha provat amb tres granularitats diferents. Els resultats obtinguts es mostren en la Taula 25.

*Taula 25: Resultats de la prova de granularitat*

| Granularitat | Mostres | Duració | Rati |
|---|---|---|---|
| Hora | 24 | 2,64 s | $3{,}1 \cdot 10^{-5}$ |
| Minut | 1.440 | 3,50 s | $4{,}1 \cdot 10^{-5}$ |
| Segon | 86.400 | 24,45 s | $2{,}8 \cdot 10^{-4}$ |

Per cada granularitat, es mostra el nombre de mostres que prendrà la simulació, la duració de la simulació i el rati de duració de simulació per temps simulat. Aquest rati podem veure que creix de forma lineal en funció del nombre de mostres, i, a més a més, va alineat amb els ratis mencionats a l'article [1].

La següent prova és la inversa de la prova anterior, és a dir, mantenir la granularitat a 1 segon i variar l'interval de simulació.

*Taula 26: Resultats de la prova d'intervals*

| Interval | Duració | Rati |
|---|---|---|
| 1 dia | 24,45 s | $2{,}83 \cdot 10^{-4}$ |
| 2 dies | 44,53 s | $2{,}58 \cdot 10^{-5}$ |
| 10 dies | 3 m 15 s | $2{,}26 \cdot 10^{-4}$ |
| 30 dies | 10 m 27 s | $2{,}42 \cdot 10^{-4}$ |

Els resultats es poden veure a la Taula 26. Hi ha una mica de fluctuació en els resultats del rati de duració, ja que no es tracta d'un entorn aïllat i són simulacions curtes. No obstant, veiem que aquests ratis es mantenen en l'ordre de magnitud esmentats a l'article [1].

# 12 Conclusions

## 12.1 Primer objectiu

El primer objectiu del treball era la implementació d'una interfície gràfica que permetés dissenyar xarxes basades en models de cues i definir els serveis a partir dels quals es generaria el tràfic seguint la metodologia CURSA-SQ.

Aquest objectiu venia amb uns requeriments molt marcats, a partir dels quals ha anat sorgint la planificació en tasques del projecte. Aquestes tasques s'han anat validant durant el seu desenvolupament pels directors del projecte, tant d'usabilitat com de funcionament.

Algunes tasques com el pla de canvis de les simulacions s'han hagut d'anar afegint durant el treball, ja que, al ser un projecte nou sense cap referència externa, algunes necessitats no s'han pogut valorar fins a tenir algunes parts ja desenvolupades per poder provar-les.

Les tasques que quedaven fora del *MVP* finalment no s'han pogut desenvolupar per manca de temps.

La part d'interfície gràfica s'ha decidit de fer amb *React*, degut als meus coneixements previs de *Javascript* i al meu interès personal.

Una part complicada del desenvolupament ha estat integrar *mxgraph* amb *React*. Alguns dels problemes que m'he trobat són els següents:

- No disposa de cap mena de tipus, així que apareixen molts problemes al no poder disposar de la comprovació de tipus en temps de transpilació. Tampoc he pogut disposar d'ajudes de l'editor de codi.
- La documentació és molt extensa i la corba d'aprenentatge és força elevada, ja que introdueix molts conceptes propis.
- Cada instància de *mxgraph* té el seu propi estat intern, i, per tant, s'ha de sincronitzar amb l'estat del component de *React*.

La part més desafiant ha estat pensar en el model *Redux* a l'hora de dissenyar la lògica de l'aplicació. Un gran al·licient ha estat trobar la llibreria *redux-saga*, que amplia les possibilitats del que es pot fer en *Redux* mitjançant una nomenclatura molt neta. Així i tot, considero que vaig trobar tard aquesta llibreria, i hi ha algunes funcionalitats que es podrien haver implementat amb *redux-saga* i no s'han pogut migrar a temps.

## 12.2 Segon objectiu

El segon objectiu era implementar una infraestructura que permetés connectar la interfície amb el servidor existent de CURSA-SQ, de forma que l'usuari pogués executar simulacions i veure'n els resultats.

Aquest objectiu era bastant més lliure a nivell de requisits. Per a fer-ho de forma consistent es va decidir implementar amb *Javascript* també.

La decisió de fer-ho de forma modular m'ha permès tenir el codi separat i no augmentar excessivament la complexitat a causa de dependències. Un cop definits els diferents serveis, *Docker* m'ha permès abstreure'm de les característiques de la màquina on s'executin, a més a més d'oferir un repositori de contenidors des d'on poder compartir el codi.

Després d'executar un conjunt de provés de rendiment hem pogut validar que la interfície no suposa un afegit considerable a nivell de temps d'execució, ja que les simulacions tenen el mateix ordre de magnitud que quan s'executen directament sobre el servidor de CURSA-SQ existent.

Com a limitació, cal comentar que les proves s'han fet sempre a petita i mitjana escala. Degut a una falta de temps i recursos no s'han pogut realitzar proves amb escenaris molt complexes o amb necessitats de duració molt exigents. En aquests casos s'hauria d'estudiar com dimensionar els recursos de les bases de dades. No obstant, tant *mongoDB* com *influxDB* són dues bases de dades que estan preparades per ser escalables, així que no hi hauria gaire problema.

## 12.3 Valoració personal

Personalment, emprendre un projecte d'aquesta envergadura ha estat tot un repte i una oportunitat per consolidar els coneixements assolits al Grau, concretament a l'especialitat de Tecnologies de la Informació, així com per adquirir-ne de nous.

Alguns dels coneixements assolits al Grau que he pogut aplicar són els conceptes de virtualització a l'hora d'estructurar l'arquitectura de l'aplicació amb *Docker*, el funcionament dels servidors i les bases de dades no relacionals.

Per altra banda, he pogut ampliar els meus coneixements sobre *Javascript* i les seves possibilitats, així com aprendre el funcionament d'un conjunt de llibreries com *React* o *redux* que em serviran en el meu futur professional.

I tot i que no acabi treballant amb aquestes tecnologies he pogut aprendre el patró de disseny que incentiva *redux*, cosa que podré aplicar en altres àmbits.

Tot i que no he arribat a aprofundir molt a l'hora d'implementar *Docker*, els coneixements bàsics adquirits em seran suficients per poder aplicar-los en el futur.

Per últim, ha estat un repte afrontar un projecte amb uns requisits ben marcats però amb llibertat a l'hora d'implementar-los. A la meva feina actual acostumo a tenir molt acotat com han de ser les funcionalitats, però en aquest projecte he hagut d'anar definint-les i modificant-les sobre la marxa, sota el meu criteri i amb l'aprovació dels directors del projecte.

En aquest cas, haver escollit *Scrum* com a metodologia de treball ha estat una decisió molt encertada, ja que ens ha donat la flexibilitat que necessitàvem.

# 13 Bibliografia

[1] M. Ruiz, F. Coltraro, and L. Velasco, "CURSA-SQ: A Methodology for Service-Centric Traffic Flow Analysis," *J. Opt. Commun. Netw.*, vol. 10, no. 9, p. 773, Sep. 2018.

[2] D. G. Kendall, "Stochastic Processes Occurring in the Theory of Queues and their Analysis by the Method of the Imbedded Markov Chain," *Ann. Math. Stat.*, vol. 24, no. 3, pp. 338–354, Sep. 1953.

[3] F. Xue and S. J Ben Yoo, *On the generation and shaping self-similar traffic in optical packet-switched networks*. 2002.

[4] F. Morales *et al.*, "Dynamic Core VNT Adaptability Based on Predictive Metro-Flow Traffic Models," *J. Opt. Commun. Netw.*, vol. 9, no. 12, p. 1202, Dec. 2017.

[5] R. M. Fujimoto, K. Perumalla, A. Park, H. Wu, M. H. Ammar, and G. F. Riley, "Large-scale network simulation: how big? how fast?," in *11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems, 2003. MASCOTS 2003.*, pp. 116–123.

[6] "OMNeT++ Discrete Event Simulator." [Online]. Available: https://omnetpp.org/. [Accessed: 03-Apr-2019].

[7] L. Drzewiecki and M. Antoniak-Lewandowska, "Flow Simulator - a flow-based network simulator," in *EUROCON 2007 - The International Conference on "Computer as a Tool,"* 2007, pp. 2132–2136.

[8] Google, "React, AngularJS, Vue.js - Google Trends." [Online]. Available: https://trends.google.com/trends/explore?date=2015-01-01 2019-02-22&q=%2Fm%2F012l1vxv,%2Fm%2F0j45p7w,%2Fg%2F11c0vmgx5d. [Accessed: 23-Feb-2019].

[9] Stack Overflow, "Developer Survey 2018." [Online]. Available: https://insights.stackoverflow.com/survey/2018#technology. [Accessed: 23-Feb-2019].

[10] R. Cimperman, *UAT Defined: A Guide To Practical User Acceptance Testing*. ADDISON-WESLEY PROFESSION, 2006.