Universitat de Girona
Escola Politècnica Superior

Grau en Enginyeria Informàtica

PROJECTE FINAL DE GRAU

Virtualització d'aplicacions amb accés via web

Autor:
Aniol Fernández Cano

Tutor:
Lluís Fàbrega Soler

MEMÒRIA

Convocatòria:
Juny 2023

Departament:
Arquitectura i Tecnologia de Computadors

Projecte: Projecte Final de Grau
Document: Memòria
Títol: Virtualització d'aplicacions amb accés via web
Autor: Aniol Fernández Cano
Data: Juny 2023

Estudi:
Grau en Enginyeria Informàtica
Universitat de Girona

Tutor:
Lluís Fàbrega Soler
Universitat de Girona
Email: lluis.fabrega@udg.edu

Índex

Índex de figures

Índex de taules

1. Introducció

1.1 Introducció i motivació

Tradicionalment, les aplicacions s'han desenvolupat de forma "nativa" per al sistema al qual l'aplicació va dirigida, de manera que el codi desenvolupat està lligat al *hardware* per al que s'havia desenvolupat. Aquest tipus d'aplicacions, tenen accés complet als dispositius i perifèrics i, per tant, poden obtenir tot el rendiment que aquests són capaços de donar, però és clar, estant lligat al sistema operatiu i maquinari concret per al qual ha estat dissenyat. Una aplicació "*web*", en canvi, no té aquest tipus de dependència ja que s'executa sobre un navegador web (present a gairebé qualsevol dispositiu). D'aquesta manera, es facilita molt més l'accessibilitat a aquest programari ja que només es requereix un dispositiu connectat a la xarxa per tal d'accedir-hi. Ara bé, contrastant amb les aplicacions "natives", les aplicacions "*web*" no disposen d'un accés complet als recursos del sistema i aquest fet en limita les seves capacitats.

D'altra banda, existeixen solucions per tal de fer una aplicació accessible i alhora eliminar la necessitat d'un *hardware* adequat per a l'aplicació, a aquestes solucions se les coneix com a aplicacions virtualitzades a través d'escriptori remot. El concepte és simple, una màquina servidora assolirà el rol d'estació de treball i els usuaris s'hi connectaran a través de la xarxa, establint una sessió remota per tal d'executar-hi programari. Fent ús d'aquesta metodologia de treball, l'usuari obté una experiència quasi "nativa" ja que fa ús dels recursos dels quals disposa la màquina servidora (aquesta estarà aprovisionada amb el maquinari més adequat per a l'aplicació) a canvi només de realitzar una connexió remota amb aquest servidor, fet que permet reduir al mínim els requisits de *hardware* d'aquests usuaris clients. El problema amb aquest enfocament però, recau en la necessitat d'haver d'instal·lar i configurar un programari que permeti la realització d'aquesta sessió remota, a més, obre un possible forat de seguretat al deixar entrar un usuari al sistema perquè hi executi una aplicació; per això el següent pas és desenvolupar aquest client en forma d'aplicació web.

El projecte pretén ser una solució integral que posi en conjunt gran part de les disciplines practicades en el grau. Per a poder oferir una solució viable i funcional, caldrà dissenyar una arquitectura de sistema distribuïda que permeti l'escalabilitat

horitzontal[1] del servei. Conjuntament amb el disseny de l'arquitectura, caldrà realitzar una implementació adient al sistema que permeti l'execució concurrent de múltiples sessions per usuari. El sistema serà gestionat a través d'un servei web que caldrà dissenyar i implementar conjuntament amb la seva la interfície web per l'usuari.

El sol desafiament de la posada en marxa d'aquest sistema representa una especial motivació personal per la gran quantitat de conceptes i peces a posar en consens. A més, suposa un interès afegit la possibilitat de desenvolupar una nova alternativa al desplegament d'aplicacions viable per al mercat.

1.2 Propòsit i objectius

El projecte busca oferir una nova alternativa al desplegament d'aplicacions, tot aplicant els millors aspectes de cada una de les solucions existents i eliminant-ne els punts febles per tal d'oferir una nova solució apta per al mercat. El principal propòsit d'aquest projecte és dissenyar i desenvolupar un servei i aplicació web que permeti accedir, a través d'un navegador, a aplicacions natives executant-se en màquines remotes i, d'aquesta manera, els usuaris finals podran fer servir aquestes aplicacions sense haver d'instal·lar-les en els seus dispositius, alliberant-los de requisits específics de *hardware*, S.O. o qualsevol altre *software* addicional més enllà d'un navegador. L'aplicació, a més, permetrà ser configurada per un usuari administrador, el qual podrà afegir noves aplicacions i limitar i controlar l'accés a aquestes. El sistema garantirà la integritat i confidencialitat de les dades dels usuaris i proporcionarà aïllament entre les execucions simultànies d'aplicacions de diversos usuaris.

Per a aconseguir-ho, es farà ús de tecnologies web modernes i es buscarà en tot moment garantir la seguretat tant de l'usuari com del sistema. En concret, amb aquest treball, es pretenen assolir els següents objectius:

1. Estudi de tecnologies i opcions d'implementació per al transport de les interaccions entre usuari i aplicació.

2. Estudi de tecnologies web i elecció de la més adient per al projecte.

3. Estudi de tecnologies de contenidors.

4. Estudi de requisits i latències per a obtenir una bona interactivitat en temps real.

5. Disseny i desenvolupament de l'arquitectura del sistema d'execució d'aplicacions.

[1] Augmentar els recursos del sistema afegit noves màquines sense canviar cap configuració.

6. Disseny i desenvolupament del servidor i API web.
7. Disseny i desenvolupament de la interfície per l'usuari.
8. Realització de proves de funcionament.
9. Documentació del servei per a usuaris administradors.

1.3 Estructura de la documentació

La continuació d'aquesta memòria es troba estructurada en els següents capítols:

1. **Estudi de viabilitat** - S'analitza quins son els aspectes que fan possible el desenvolupament del projecte.
2. **Metodologia** - És detalla la metodologia de treball seguida per al desenvolupament del projecte.
3. **Planificació** - Desglossament de les tasques a realitzar, el temps previst i la planificació de realització.
4. **Marc de treball i conceptes previs** - Es comenten les eines ja existents que permeten fer una feina semblant a l'objectiu del projecte i la distinció entre aplicació natives, web i l'execució remota.
5. **Requisits del sistema** - S'especifiquen quins son els requisits funcionals i no funcionals que ha de complir el sistema.
6. **Estudis i decisions** - S'analitzen les diferents tecnologies que fan possible el desenvolupament i s'argumenta la utilització de les escollides.
7. **Disseny de l'aplicació de virtualització** - S'especifica el disseny que tindrà l'arquitectura del sistema i la interfície d'usuari.
8. **Implementació i proves** - Es detalla el procés de desenvolupament de l'aplicació juntament amb les proves de funcionament realitzades.
9. **Resultats** - Es presenta i comenta el resultat final obtingut.
10. **Conclusions i treball futur** - Comentaris sobre el projecte, conclusions i treball futur a realitzar.
11. **Manual per a l'administrador** - Especificació d'instal·lació i configuració per a un usuari administrador que vulgui implementar el sistema.

2. Estudi de viabilitat

Per tal d'elaborar l'anàlisi de viabilitat del projecte, ens posarem en la tessitura d'una empresa de desenvolupament de programari que, analitzant el mercat, veu potencial en aquest nou servei i en vol realitzar el desenvolupament de forma autònoma. Donem per amortitzat qualsevol maquinari necessari per a dur a terme el desenvolupament i deixem a càrrec de qui vulgui utilitzar el servei qualsevol hardware necessari per a mantenir-lo en funcionament.

2.1 Viabilitat legal

Aquest projecte, per la seva naturalesa i objectiu (distribuir aplicacions a través de la web) pot ser objecte de polèmica a l'hora d'analitzar la seva legalitat, ja que pot ser usat per a la retransmissió d'aplicacions per les quals no es disposi del permís necessari.

Per tal de d'evitar aquest impediment, el projecte es centrarà només en el desenvolupament del servei que fa possible la distribució d'aplicacions mitjançant la web, sense incloure-hi cap producte o aplicació concreta que pugui veure vulnerada la seva propietat intel·lectual. En cas d'utilitzar-se alguna aplicació per a la demostració del funcionament, es procurarà utilitzar aquelles aplicacions que la seva tipologia de llicència en permeti la lliure distribució sense permís explícit.

2.2 Viabilitat econòmica i recursos humans

El projecte serà desenvolupat utilitzant eines, mòduls i llibreries de programari lliure o bé amb llicències permissives que en permetin el desenvolupament i distribució de l'aplicació de forma gratuïta.

El desenvolupament del projecte no tindrà cap cost econòmic afegit més enllà del temps de desenvolupament i anàlisi requerida que, en aquest cas, hauria d'aproximar-se a les 375 hores de feina si tenim en compte que el projecte s'ha de desenvolupar dins del marc de l'assignatura del TFG (que suposa un pes de 15 crèdits ECTS, representant un aproximat de 25 hores de feina per cada crèdit).

Suposarem que la implementació del projecte pot ser realitzada per un equip de tres persones, dissenyador gràfic, programador web i l'enginyer encarregat del sistema.

Així doncs, donarem un preu brut per hora simbòlic de cadascun dels integrants basant-nos en el sou mitjà d'aquestes posicions:

Integrant	**Sou brut per hora**
Enginyer informàtic	20€
Programador web	13€
Dissenyador gràfic	11€

TAULA 2.1: Sous bruts per hora i posició

Desglossant de forma general les hores implicades per cadascun dels integrants en el projecte, obtenim una visió general del cost total de desenvolupament;

Activitat de	**Hores d'implicació en el treball**	**Cost per hora**	**Cost**
Enginyer informàtic	200 h	*20€/h*	**4.000€**
Programador web	160 h	*13€/h*	**2.080€**
Dissenyador gràfic	15 h	*11€/h*	**165€**
		Cost total del projecte:	**6.245€**

TAULA 2.2: Desglossament del pressupost del projecte

Com veiem, el projecte suposa un cost total de 6.245€. Dins del pressupost hi contemplem tot el temps requerit per a realitzar la recerca i estudi de les possibilitats d'implementació del projecte així com la confecció de la documentació.

2.3 Viabilitat tecnològica

Per a dur a terme aquest projecte, es farà ús de tecnologies de contenidors de programari i retransmissió de vídeo per la web ja existents. El projecte consistirà en la correcta elecció d'aquestes tecnologies i la seva integració al sistema per al correcte funcionament d'aquest. Més endavant, durant els estudis i decisions (vegeu capítol 7) es farà menció a les tecnologies proposades i escollides per a la realització del projecte.

3. Metodologia

3.1 Metodologia de treball

Prèviament al desenvolupament de l'aplicació del projecte, caldrà realitzar un estudi de tecnologies candidates per a la implementació per tal d'escollir la més adient. Per tal de posar a prova aquestes tecnologies es faran estudis i petites proves de concepte per tal d'avaluar el seu ús. Un cop definides les tecnologies a utilitzar, el desenvolupament de l'aplicació seguirà una metodologia de treball àgil basada en iteracions incrementals. En concret, s'ha dividit tota la funcionalitat a implementar del projecte en tasques (vegeu capítol 4) on cada una de les tasques aporta una funcionalitat concreta al projecte. Cada una de les tasques a realitzar seguirà el següent desenvolupament:

1. Anàlisi de la tasca i disseny de la solució
2. Implementació de la funcionalitat
3. Proves de funcionament sobre la funcionalitat i la integració

Així doncs, el flux de treball complet del projecte serà el següent:

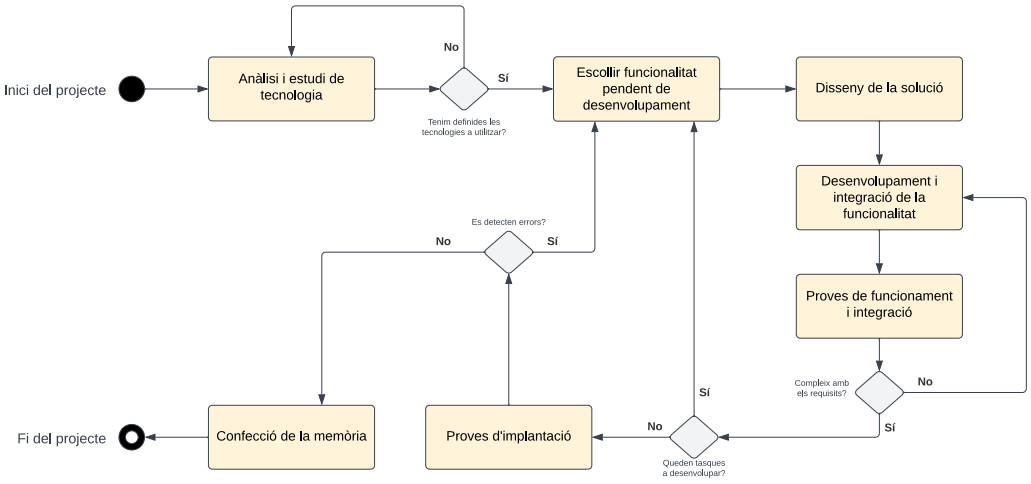

FIGURA 3.1: Flux de treball

4. Planificació

4.1 Planificació del projecte

Per tal de dur a terme el projecte, es realitzarà una planificació inicial de les tasques a realitzar amb una estimació temporal per cada una d'elles. A l'hora de portar el seguiment del projecte, es realitzaran reunions setmanals amb el tutor del treball per a validar les implementacions i decisions realitzades i seguir la continuïtat al projecte. A més, es farà ús d'eines externes com ara TRELLO[1] per a portar l'organització de les tasques.
El projecte té una duració aproximada de 14 setmanes, amb un pes de 375 hores[2] de feina, degut a això, es planificarà el projecte tenint en compte setmanes de 5 dies i una dedicació diària d'aproximadament 5.5 hores.

4.1.1 Classificació de les tasques a realitzar

A fi de portar el control i organització de les tasques a realitzar, aquestes s'han dividit en els següents blocs:

- Tasques prèvies al desenvolupament
  - Estudi de les possibilitats d'implementació
  - Disseny de l'arquitectura i sistema
- Tasques de desenvolupament
  - Desenvolupament del client web
  - Desenvolupament del servidor web
  - Desenvolupament del servidor d'aplicacions
  - Desenvolupament del contenidor d'aplicació
- Proves de funcionament i correcció d'errors
- Confecció de la documentació

[1] Aplicació web per al control i organització de tasques. https://trello.com/
[2] Pes de 15 ECTS x 25h/ECTS

4.1.2 Estimació temporal

Seguidament, es desglossa cadascun dels blocs a desenvolupar juntament amb les seves subtasques i estimació en dies de feina.

Tasques prèvies al desenvolupament	
Disseny de l'arquitectura i sistema	2
Estudi de possibilitats d'implementació	10
Desenvolupament del client web	
Accés al sistema de fitxers	2
Administració del lloc	2
Disseny i implementació IU	2
Inici de sessió i registre	1
Obrir una aplicació	4
Obtenció d'aplicacions disponibles	1
Obtenir retransmissió de vídeo	1
Registrar esdeveniments de teclat & ratolí	1
Desenvolupament del servidor web	
Catàleg d'aplicacions per usuari	1
Concessió d'accés a servidor d'aplicacions	2
Creació del servei	1
Gestió d'aplicacions	2
Gestió d'usuaris	2
Registre i control de servidor d'aplicacions	2
Desenvolupament del contenidor d'aplicacions	
Captura i retransmissió de vídeo	5
Execució d'aplicacions gràfiques	2
Simulació d'esdeveniments de teclat i ratolí	1
Desenvolupament del servidor d'aplicacions	
Inici d'imatges a petició	3
Concedir accés al sistema de fitxers	2
Proxy d'esdeveniments de teclat & ratolí	1
Sincronització d'imatges amb el catàleg	1
Subscriure's al servidor central	1
Vídeo en temps real de l'aplicació	3
Altres	
Confecció de la documentació	15
Proves de funcionament i correcció d'errors	5

TAULA 4.1: Estimació temporal en dies de feina

4.1.3 Planificació temporal

TASQUES	FEBRER		MARÇ					ABRIL				MAIG				
	20 - 24	27 - 28	1 - 3	6 - 10	13 - 17	20 - 24	27 - 31	3 - 7	10 - 14	17 - 21	24 - 28	1 - 5	8 - 12	15 - 19	22 - 26	29 - 31
Prèvies al desenvolupament																
Estudi de possibilitats d'implementació	5	2	3													
Disseny de l'arquitectura i sistema				2												
Contenidor d'aplicacions																
Execució d'aplicacions gràfiques				1	1											
Captura i retransmissió de vídeo					4											
Simulació d'esdeveniments de teclat i ratolí							1									
Client Web																
Disseny i implementació IU				2												
Obtenir retransmissió de vídeo						1										
Registrar esdeveniments de teclat & ratolí						1										
Accés al sistema de fitxers							1	1								
Inici de sessió i registre								1								
Administració del lloc									2							
Obtenció d'aplicacions disponibles										1						
Obrir una aplicació											3	1				
Servidor d'aplicacions																
Vídeo en temps real de l'aplicació						3										
Proxy d'esdeveniments de teclat & ratolí							1									
Concedir accés al sistema de fitxers							2									
Subscriure's al servidor central										1						
Inici d'imatges a petició										1	2					
Sincronització d'imatges amb el catàleg												1				
Servidor Web																
Creació del servei								1								
Gestió d'usuaris								2								
Gestió d'aplicacions									2							
Catàleg d'aplicacions per usuari									1							
Registre i control de servidor d'aplicacions										2						
Concessió d'accés a servidor d'aplicacions												2				
Altres																
Proves de funcionament i correcció d'errors												1	4			
Confecció de la documentació													1	5	5	4

FIGURA 4.1: Diagrama de Gantt - Planificació temporal de les tasques

5. Marc de treball i conceptes previs

Aquest capítol té com a objectiu situar al lector en el marc de treball del projecte. Per aquest motiu, es descriuen i comenten de forma generalitzada les diferents tipologies d'aplicacions i els components que les caracteritzen juntament amb altres conceptes necessaris per a la comprensió del document.

5.1 Tipus d'aplicacions

5.1.1 Aplicacions natives

Simplificacions: Existeixen diferents tipologies d'aplicacions natives. En aquest treball, reduirem el concepte d'aplicació nativa a aquelles aplicacions que resideixen i s'executen directament en la màquina de l'usuari i fan ús del seu *hardware*.

Les aplicacions natives [1] són un tipus de programari que requereix ser instal·lat i executat en una plataforma específica (per a la qual el programa ha estat dissenyat). Aquestes aplicacions es veuen lligades al sistema operatiu i arquitectura concreta per a les que han estat generades. Aquestes, tenen diversos avantatges respecte a altres tipus de *software*, a continuació en destaquem les més importants:

- **Millor rendiment**: les aplicacions natives estan compilades específicament per a una plataforma i arquitectura concreta, per la qual cosa, poden aprofitar al màxim els recursos de la màquina en què s'executen.
- **Accés als recursos del sistema**: Al poder interactuar directament amb el sistema operatiu, les aplicacions natives disposen d'accés complet als recursos del sistema i a les seves característiques. A causa d'això, aquestes poden realitzar funcions que amb altres tipologies d'aplicacions no s'és possible.

Per la naturalesa d'aquestes aplicacions, l'execució i poder de còmput recau completament en el *hardware* de la màquina amfitriona del programa i aquest fet, pot limitar als usuaris en poder o no executar una aplicació segons si disposen del maquinari i sistema adequat.

La següent taula resumeix les qualitats de les aplicacions natives interessants per a aquest projecte:

Configuració local	Requereixen ser instal·lades / executades localment i de la seva pertinent configuració
Accessible	Cal distribuir diferents versions dels executables segons els sistemes operatius i arquitectures a les quals distribuïm l'aplicació, fet que dificulta la distribució.
Sistema/hardware específic	Són dissenyades per un sistema i hardware específic.
Accés als recursos	Permeten accés complet als recursos del sistema.

TAULA 5.1: Punts d'interès sobre les aplicacions natives

5.1.2 Aplicacions web

Les aplicacions web [2] són aplicacions dissenyades per a ser accedides i executades utilitzant un navegador web. Les aplicacions web són pràctiques ja que no requereixen de cap instal·lació ni maquinari concret, només requereixen connexió amb el servidor que proporciona l'aplicació. Aquest tipus d'aplicacions es veuen limitades en les accions que poden realitzar perquè només disposen de l'accés que el navegador web proporciona sobre el sistema.

Per tal de dotar amb més capacitats a les aplicacions web, existeixen essencialment dos vessants:

- **Acceleració per *hardware***: El navegador web posa a disposició de les aplicacions web certes interfícies que permeten l'execució de blocs funcionals de *hardware*. Aquesta tecnologia fa possible, per exemple, l'ús de la GPU (*Graphics Processing Unit*) a través d'interfícies com `WebGL` [3].

- **Processat asíncron al servidor**: L'aplicació web realitza peticions a un servidor per tal que realitzi el processat de les dades. Aquesta opció permet alliberar a l'aplicació web de dur a terme l'esforç computacional del processament de les dades a canvi d'esperar que el servidor retorni una resposta.

Tot i disposar de mecanismes per a enriquir les aplicacions web i permetre la majoria d'usos que requereixen cert processament de dades, aquestes no tenen el mateix potencial que una aplicació nativa, i fa inviable el desenvolupament de programaris d'ús intensiu del *hardware*.

La següent taula resumeix les qualitats de les aplicacions web interessants per a aquest projecte:

Configuració local	No requereixen cap configuració en la màquina de l'usuari.
Accessible	Fàcilment distribuïbles. Una sola plana web accessible a través del navegador.
Sistema/hardware específic	No estan lligades a cap hardware ni sistema específic.
Accés als recursos	No poden accedir a tot el potencial del hardware del sistema.

TAULA 5.2: Punts d'interès sobre les aplicacions web

5.1.3 Aplicacions a través de sessions remotes

Les sessions remotes permeten a un usuari realitzar una connexió a través de la xarxa amb un servidor de treball. Aquesta sessió permet la comunicació entre l'usuari i el servidor de manera que l'usuari rep la imatge o sortida generada i hi envia esdeveniments de teclat i ratolí per tal de controlar el sistema. Típicament, l'establiment d'una sessió remota genera una sessió virtual per a l'usuari connectat i li permet accedir i controlar aquest sistema de forma remota.

L'ús de sessions remotes allibera a l'usuari de qualsevol requisit específic de maquinari ja que tot el processament es realitza en el servidor. Aquest fet, permet que la màquina de l'usuari disposi d'un *hardware* molt simple, només capaç d'establir i processar la connexió. A aquest tipus de màquines simples se'ls anomena *ThinClient* [4].

Per tal d'establir una sessió remota, cal prèviament haver configurat el servidor de treball per a acceptar la connexió de l'usuari i haver configurat la màquina del client amb el programari i dades de connexió necessàries. És a dir, es requereix un programari específic que implementi el protocol de la connexió remota i de la correcta configuració d'aquest. Aquest fet en limita el seu ús de forma flexible i sense cap configuració prèvia. A més, permetre la connexió d'un usuari a la màquina servidora obra un potencial forat de seguretat perquè l'execució de la sessió es troba en el sistema del servidor, limitada això sí, pels permisos que disposi aquest usuari.

La següent taula resumeix les qualitats de les aplicacions a través de sessió remota interessants per a aquest projecte:

Configuració local	Requereixen de la instal·lació i configuració del software que permet la connexió.
Accessible	Un cop configurat, l'usuari té accés complet al sistema i aplicacions disponibles en la sessió.
Sistema/hardware específic	L'usuari no està lligat a cap requisit específic.
Accés als recursos	Permeten accés complet als recursos del sistema remot.

TAULA 5.3: Punts d'interès sobre les aplicacions sobre sessió remota

Seguidament, es procedeix en comentar alguns protocols i eines que permeten establir una sessió remota.

- **Secure Shell**
  SSH (*Secure Shell*) [5] és un protocol i programa que permet l'accés i control remot d'un servidor. Està orientat a establir connexions a través de línia de comandes (l'usuari envia esdeveniments de teclat i rep la sortida de la consola) tot i que també pot ser usat per a aplicacions gràfiques utilitzant la redirecció de gràfics d'X11 (vegeu capítol 7 per a més detalls sobre X11) ara bé, requereix que el client disposi d'un client d'X11 per a poder visualitzar la sortida gràfica.

- **Remote Desktop Protocol**
  RDP (*Remote Desktop Protocol*) [6] és un protocol (propietari de Microsoft) que permet l'establiment de sessions remotes amb un "terminal server" (servidor que executa el servei *Remote Desktop Services*). L'establiment d'una sessió a través d'RDP requereix d'un usuari del sistema del servidor (o del domini) amb els permisos necessaris per a l'establiment de la connexió, si s'estableix la connexió, el servidor genera una nova sessió per a l'usuari creant-ne un escriptori virtual de Windows. Dins la sessió remota, l'usuari pot interactuar amb l'escriptori virtual com si es tractés del seu propi sistema. El protocol també ofereix la possibilitat de no crear un escriptori virtual i retransmetre només els gràfics d'una aplicació concreta.

- **Virtual Network Computing**
  VNC (*Virtual Network Computing*) és un programa de compartició de sistema que utilitza el protocol RFB [7] per a controlar un sistema remot. A diferència d'altres sistemes, VNC no genera una sessió virtual per a la connexió remota sinó que cedeix el control d'una sessió ja existent a l'usuari remot.

5.1.4 Citrix Virtual Apps

Citrix Virtual Apps and Desktops, comunament conegut com a XenApps, és una aplicació de virtualització d'aplicacions desenvolupada per l'empresa Citrix. Aquest programari permet la virtualització d'aplicacions individuals i d'escriptoris Windows i la seva retransmissió cap a un client amb *Citrix Workspace* [8]. L'aplicació és una eina comercial i de pagament, que permet als usuaris executar aplicacions i escriptoris de Windows independentment del sistema operatiu que estigui utilitzant. Per a fer-ho, es basen en una implementació pròpia del protocol RDP adaptant-lo i encapsulant-lo sota un protocol propi desenvolupat per Citrix anomenat HDX [9]. Per tal d'accedir a un sistema amb XenApps, l'usuari pot optar en usar l'accés via web o bé a través de l'aplicació d'escriptori. Ara bé, en qualsevol dels casos, l'usuari requereix prèviament tenir instal·lat el programari *Citrix Workspace* i estar autenticat al sistema remot.

La següent taula resumeix les qualitats de la virtualització d'aplicacions mitjançant Citrix Virtual Apps:

Configuració local	Requereixen la instal·lació i configuració del *Citrix Workspace*.
Accessible	Un cop configurat, l'usuari té accés complet al sistema i aplicacions disponibles en la sessió.
Sistema/hardware específic	L'usuari no està lligat a cap requisit específic.
Accés als recursos	Permeten accés complet als recursos del sistema remot.

TAULA 5.4: Punts d'interès sobre la virtualització amb Citrix Virtual Apps

5.1.5 Comparació entre tipus d'aplicacions

La següent taula permet comparar les característiques analitzades de cada tipus d'aplicació i a més s'hi inclou l'aplicació proposta a desenvolupar per aquest projecte i que pretén donar solució a totes les característiques analitzades. Amb la taula, es pretén comparar i valorar els diferents pros i contres dels conceptes i tecnologies esmentats en les seccions anteriors envers l'aplicació proposta a desenvolupament per a aquest projecte.

	Sense configuració local [1]	***Fàcilment accessible*** [2]	***Sense sistema específic*** [3]	***Accés als recursos del sistema*** [4]
Aplicació natives	✗	✗	✗	✓
Aplicació web	✓	✓	✓	✗
Aplicació a través de sessió remota	✗	✓	✓	✓
Aplicació a través de XenApps	✗	✓	✓	✓
Proposta a desenvolupar	✓	✓	✓	✓

TAULA 5.5: Comparativa entre tipus d'aplicacions i la proposta a desenvolupar

[1] Configuració local: Instal·lar o configurar algun element a la màquina de l'usuari.
[2] Fàcilment accessible: Referent a la dificultat de distribució de l'aplicació o element a executar.
[3] Sistema específic: Aplicació lligada a un sistema operatiu o arquitectura concreta que l'usuari ha de disposar.
[4] Accés als recursos del sistema: Referent a la capacitat de, si es necessita, poder accedir a qualsevol recurs disponible del sistema en que s'executa.

5.2 Temps de resposta i impacte en la interacció amb aplicacions

5.2.1 Impacte del temps de resposta en l'experiència d'usuari

El temps de resposta en les interaccions amb les aplicacions juga un paper clau en com és d'usable una aplicació. Per tal de sintetitzar aquest concepte, imaginem un sistema molt simple on l'usuari disposa una pantalla de color vermell on aquesta canvia de color a blau quan rep un clic del ratolí de l'usuari (veure figura 5.1). Aquesta interacció, tot i ser molt ràpida, no té efecte instantani i té una gran importància en la percepció i usabilitat de l'aplicació. A l'interval de temps succeït entre la realització de l'acció de l'usuari i mostrar-se efectivament el canvi en la pantalla l'anomenarem **temps de resposta**. De forma generalitzada, els humans disposen d'alta sensibilitat al temps de resposta [10] ocasionant que retards superiors als *110ms* siguin perceptibles i ocasionin errors humans si s'utilitzen sistemes d'interacció indirecta; ratolí, teclat, etc.

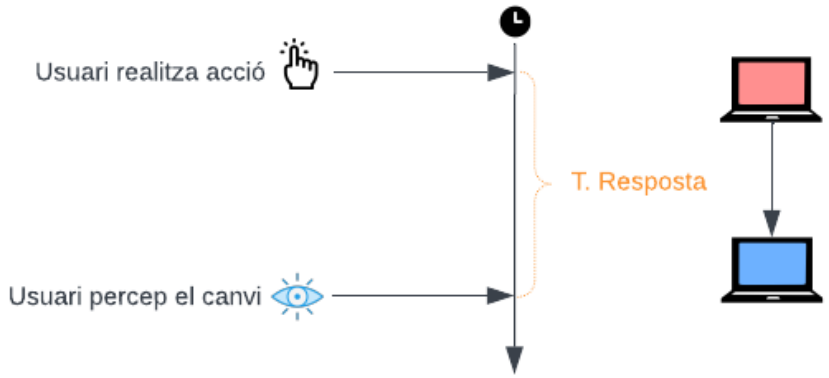

FIGURA 5.1: Percepció del temps de resposta en una aplicació d'escriptori

5.2.2 Temps de resposta en una aplicació en xarxa

El temps de resposta en les aplicacions d'escriptori, sol ser mínim, ja que qualsevol acció té efecte pràcticament immediat i només depèn del temps d'execució i renderització dels gràfics. En una aplicació en xarxa, però, aquest temps es veu incrementat pel temps de transmissió i enviament de les dades. Suposem la mateixa aplicació que en l'exemple anterior però aquest cop essent executada sobre una sessió en un equip remot a través de la xarxa. El temps de resposta efectiu que experimentarà l'usuari serà causat pel temps d'enviament de l'acció a realitzar cap a la màquina remota, el processat i renderitzat del resultat i finalment l'enviament del resultat i renderitzat per part de l'usuari. Així doncs, de forma general i simplificada[1], direm

[1]No es tenen en compte temps d'anada i tornada, control d'errors, etc. Suposem un protocol de transmissió d'un sol sentit.

que el temps de resposta en una aplicació executada en una sessió remota vindrà donat pel següent càlcul (vegeu figura 5.2):

$$\text{T. Resposta} = \mathbf{t1} + \mathbf{t2} + \mathbf{t3} + \mathbf{t4} + \mathbf{rp_1} + \mathbf{rp_2}$$

On:
$t1$ = *temps de processament i transmissió de l'esdeveniment (captura, codificació, compressió i enviament del/s paquet/s)*
$t2$ = *temps de recepció de l'esdeveniment (recepció, descompressió, descodificació i processament del/s paquet/s) i processament de les accions a realitzar (si és un clic, fer-lo i renderitzar el resultat). i enviament dels canvis (recepció, descompressió, descodificació, processat, codificació, compressió i enviament)*
$t3$ = *temps de processament i transmissió dels gràfics (captura, codificació, compressió i enviament del/s paquet/s)*
$t4$ = *temps de recepció i renderitzat del canvi (recepció, descompressió, descodificació, processat i renderitzat de la imatge)*
rp_1 = *retard del/s paquet/s des de client fins servidor*
rp_2 = *retard del/s paquet/s des de servidor fins client*

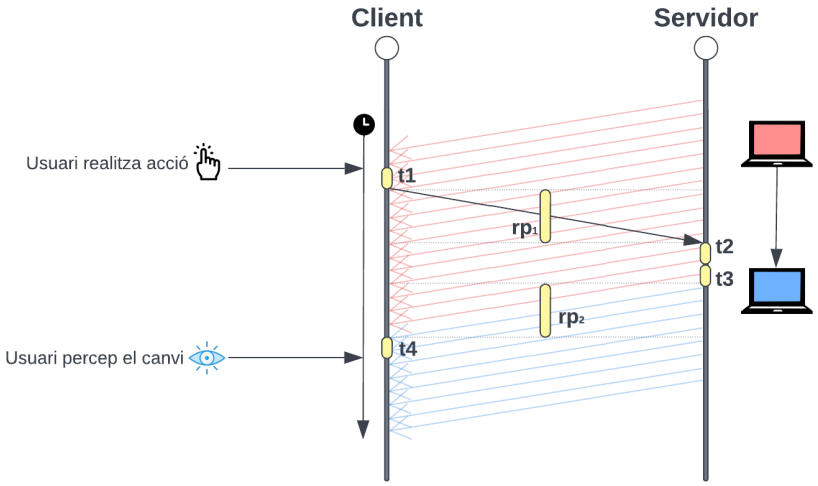

FIGURA 5.2: Percepció de latència en una aplicació d'escriptori a través d'una sessió remota

Notis com en aquest esquema, podem fixar els temps $t1$, $t2$, $t3$ i $t4$ pràcticament com a constant ja que l'element que variarà més i serà crític per a l'experiència de l'usuari serà el retard dels paquets entre el client i el servidor.

Així doncs, caldrà ser curosos i analitzar quin és el retard màxim que hauran d'experimentar els paquets enviats pels nostres usuaris. El retard entre client i servidor vindrà donat per molts factors que no podrem controlar; nombre d'enllaços, capacitat i ocupació de les cues dels enllaços i distància entre enllaços. Com que el nombre

d'enllaços i l'ocupació de les cues que tindrà la connexió client-servidor no és previsible en el món real, l'àmbit que podem analitzar és la distància de la connexió.

En el món real, les connexions no són directes, establir una connexió amb un servidor a distància requereix travessar diversos enllaços i retencions degudes al tràfic de xarxa d'aquell precís instant. A més, com més enllaços s'hagin d'utilitzar, més probabilitat d'error hi haurà en la connexió, causant un retard encara més gran. A causa d'això, és impossible estimar quin serà el temps de resposta efectiu entre dos servidors a la distància.

El que si podem fer però, és realitzar una estimació estadística del temps d'anada i tornada de paquets entre servidors situats en diferents ciutats. En aquest cas, fent servir les estadístiques proveïdes per WonderNetwork [11] podem analitzar els temps de resposta mitjans entre Barcelona i diferents ciutats del món. Notar que aquestes estimacions s'han realitzat mitjançant *pings*[2].

Paris (850 km)	**London (1.150 km)**	**Moscow (3.000 km)**	**Cairo (2.900 km)**	**Washington (6.500 km)**	**Toronto (6.400 km)**	**Caracas (7.500 km)**	**Cape Town (8.500 km)**	**Tokyo (10.400 km))**	**Auckland (19.200 km)**
23ms	28ms	71ms	75ms	96ms	110ms	147ms	171ms	213ms	274ms

TAULA 5.6: Temps de resposta mig entre Barcelona i ciutats del món

Nota: Distàncies terrestres aproximades entre Barcelona i destí.

Com veiem, les ciutats properes tenen un menor temps de resposta en comparació a les ciutats llunyanes. Tenint això en compte, si volguéssim donar servei mundial (o en diferents països), caldria distribuir diversos servidors d'aplicacions en funció del temps de resposta que experimentaran els usuaris, tenint present que aquest, comença a ser perjudicial per a l'experiència de l'usuari a partir dels 110ms.

[2]Paquets del protocol ICMP, avaluen el retard mig entre dues estacions.

6. Requisits del sistema

6.1 Requisits de maquinari/programari pel desenvolupament

Els requisits de maquinari i programari s'han extret a partir de les necessitats específiques del projecte i dels requisits dels components més demandants:

- Sistema operatiu Linux amb client d'X11 instal·lat
- CPU amb arquitectura AMD64 i suport per virtualització KVM.
- Com a mínim 4 GB de RAM disponible
- 10 GB de memòria lliure

6.2 Requisits no funcionals

L'aplicació a desenvolupar haurà de permetre a l'usuari final accedir a les aplicacions disponibles en el sistema implementat de forma dinàmica i sense haver d'instal·lar ni configurar cap mena de programari o extensió en el seu equip. L'ús de l'aplicatiu desenvolupat només requerirà la disposició d'un navegador web i connectivitat amb el sistema.

El sistema dissenyat haurà de ser segur; tant per l'usuari com pel mateix sistema. Les connexions hauran de seguir els estàndards actuals quant al xifratge de les comunicacions i el sistema haurà de garantir l'aïllament dels usuaris de manera que les seves accions es trobin en un marc controlat.

Caldrà proporcionar eines per a permetre que el servei escalí de manera horitzontal[1]. Aquest fet serà necessari a causa de dos factors:

1. **Capacitat de servei**: Una màquina servidora tindrà una capacitat limitada per a servir les peticions dels usuaris. Quan aquesta capacitat es vegi sobrepassada serà necessari assignar als nous usuaris en una màquina diferent en la qual es disposi de capacitat.

[1] Escalabilitat horitzontal: Permetre augmentar els recursos del sistema afegint noves instàncies que ofereixin el servei.

2. **Temps de resposta**[2]: La qualitat del servei es veurà afectada pel temps de resposta en la realització d'accions. Per tal de disminuir el temps de resposta, caldrà poder situar màquines servidores més a prop de l'usuari i això implica disposar de diverses instàncies de servidor (vegeu capítol 5.2.2).

Així doncs, podem resumir els requisits no funcionals del sistema com:

- L'aplicació no requerirà configuració per part de l'usuari.
- El sistema haurà de ser segur, tant per l'usuari com pel mateix sistema.
- Caldrà proporcionar eines per a l'escalabilitat del servei.

6.3 Requisits funcionals

- **Distinció entre usuaris**
  L'aplicació desenvolupada distingirà entre tres tipus d'usuaris; visitant, registrat i administrador. Tant usuaris visitants com registrats disposaran de les mateixes funcionalitats dins l'aplicació però els registrats podran disposar d'accés a les aplicacions que se'ls ha concedit accés específic. L'usuari administrador disposarà de funcionalitats addicionals per tal d'administrar el servei.

- **Funcionalitats de l'aplicació**
  Tots els usuaris podran accedir a la plana de virtualització d'aplicacions, en aquesta, els usuaris podran visualitzar les aplicacions que tenen disponibles per a ser executades i les que actualment està executant. L'usuari podrà sol·licitar l'execució d'una aplicació realitzant un clic sobre aquesta i un cop oberta podrà interactuar amb ella a partir d'esdeveniments de teclat i ratolí. En tot moment s'hauran de diferenciar l'aplicació actualment activa en primer pla, les actives però en segon pla i finalment les inactives.

- **Funcionalitats addicionals durant l'execució d'una aplicació**
  Durant l'execució d'una aplicació, l'usuari podrà fer ús de la pantalla completa per al control de l'aplicació. Es posarà a disposició de l'usuari un accés al sistema de fitxers del sistema on s'està executant l'aplicació per tal que es puguin pujar i descarregar fitxers entre la màquina de l'usuari i la instància de l'aplicació en execució.

[2]Interval de temps entre que l'usuari realitza una acció i aquesta es percep com a realitzada

- **Administració del servei**
  L'usuari administrador disposarà un apartat especial de configuració i administració del servei des del qual podrà assignar diferents rols als usuaris registrats, gestionar les aplicacions disponibles i visualitzar les màquines d'execució disponibles.

- **Gestió d'aplicacions**
  La gestió d'aplicacions des de la plataforma permetrà a l'usuari administrador canviar la configuració bàsica de l'aplicació i controlar quins usuaris disposen d'accés a l'aplicació, permetent limitar l'accés a qualsevol usuari, usuaris només registrats o bé a aquells que compleixin un rol.

Així doncs, podem resumir els requisits funcionals del sistema com:

- Hi haurà tres tipus d'usuaris; visitant, registrat i administrador.
- Tots podran realitzar les mateixes accions a excepció de l'administrador que a més, disposarà d'un panell de configuració.
- Qualsevol usuari podrà veure les aplicacions que té disponibles i executar-les.
- L'usuari podrà interactuar amb una aplicació en execució, de manera que:
  - Podrà controlar-la enviant esdeveniments de teclat i ratolí
  - Visualitzarà els gràfics de l'aplicació
  - Podrà accedir a un sistema de fitxers compartit on l'aplicació tindrà accés.
- L'usuari podrà identificar en tot moment quines aplicacions té disponibles, en execució i en primer pla.
- Durant l'execució d'una aplicació, aquesta es podrà controlar en pantalla completa.
- L'administrador del sistema podrà accedir al panell d'administració on podrà realitzar les següents accions:
  - Gestionar els rols disponibles i assignar-los a usuaris.
  - Gestionar el permís d'accés a les aplicacions (segons tipus d'usuari i rol).
  - Visualitzar l'estat del sistema.

6.4 Cas d'ús general

Basat en els requisits funcionals del sistema, a continuació es presenta el cas d'ús general per tal de descriure què ha de complir l'aplicació a desenvolupar.

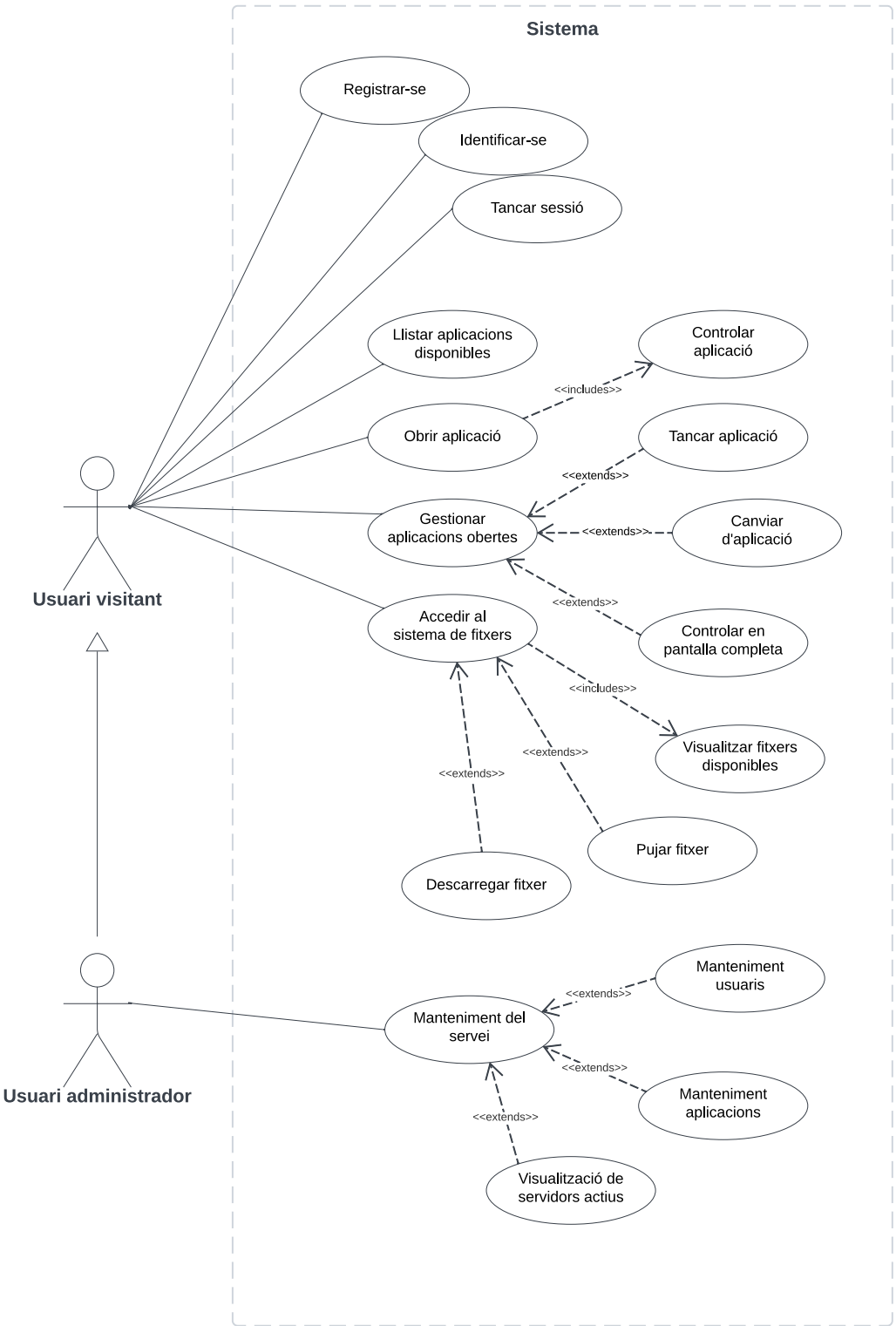

FIGURA 6.1: Cas d'ús general

6.4.1 Fitxes dels casos d'ús

6.4.1.1 Registrar-se

Cas d'ús	Registrar-se
Actor	Usuari visitant
Objectiu	Permetre a l'usuari registrar-se en el sistema.
Postcondició	L'usuari queda registrat al sistema i automàticament s'identifica.
Escenari principal	El sistema presenta a l'usuari un formulari sol·licitant el seu nom d'usuari, correu i contrasenya. L'usuari introdueix les dades al formulari i prem el botó per registrar-se. El sistema comprova que les dades tinguin un format correcte i que el nom d'usuari no existeixi ja al sistema. Al finalitzar, el sistema porta a l'usuari a la plana principal.
Escenari alternatiu	En cas d'error de validació de dades, es mostra un missatge a l'usuari indicant l'error.

TAULA 6.1: Fitxa del cas d'ús: Registrar-se

6.4.1.2 Identificar-se

Cas d'ús	Identificar-se
Actor	Usuari visitant
Objectiu	Permetre a l'usuari identificar-se en el sistema.
Postcondició	L'usuari queda identificat i autenticat al sistema.
Escenari principal	El sistema presenta a l'usuari un formulari sol·licitant el seu nom d'usuari i contrasenya. L'usuari introdueix les dades al formulari i prem el botó per identificar-se. El sistema comprova que les dades siguin correctes. Al finalitzar, el sistema porta a l'usuari a la plana principal.
Escenari alternatiu	En cas de no existir l'usuari o introduir dades d'accés errònies es mostrarà un error indicant-ho.

TAULA 6.2: Fitxa del cas d'ús: Identificar-se

6.4.1.3 Tancar sessió

Cas d'ús	Tancar sessió
Actor	Usuari visitant
Objectiu	Permetre a l'usuari tancar la seva sessió
Precondició	L'usuari es troba identificat al sistema.
Postcondició	Es tanca la sessió de l'usuari.
Escenari principal	El sistema mostra a l'usuari que disposa de la sessió iniciada i li habilita un botó per a poder tancar la sessió. Al prémer el botó la sessió queda tancada.

TAULA 6.3: Fitxa del cas d'ús: Tancar sessió

6.4.1.4 Llistar aplicacions disponibles

Cas d'ús	Llistar aplicacions disponibles
Actor	Usuari visitant
Objectiu	Visualitzar les aplicacions disponibles a executar per l'usuari
Postcondició	Es mostra el llistat d'aplicacions disponibles.
Escenari principal	El sistema mostra a l'usuari un llistat amb les aplicacions disponibles. L'usuari en podrà veure el nom i el logo si aquest està definit.
Escenari alternatiu	Si el sistema no té aplicacions o els permisos de l'usuari ocasionen que no hi hagi cap aplicació disponible per a ell, es mostrarà un missatge indicant la falta d'aplicacions.

TAULA 6.4: Fitxa del cas d'ús: Llistar aplicacions disponibles

6.4.1.5 Obrir aplicació

Cas d'ús	Obrir aplicació
Actor	Usuari visitant
Objectiu	Iniciar l'execució d'una aplicació.
Postcondició	S'inicia l'execució d'una aplicació.
Escenari principal	L'usuari escull una aplicació de la llista de disponibles i el sistema n'inicia l'execució. Mentre el sistema assigna els recursos i estableix la connexió es mostrarà una animació de càrrega.
Escenari alternatiu	Si l'aplicació ja es troba en execució i es troba en primer pla no es realitzarà cap acció. En cas de trobar-se en execució però no en primer pla aquesta passarà a ser-ho.

TAULA 6.5: Fitxa del cas d'ús: Obrir aplicació

6.4.1.6 Controlar aplicació

Cas d'ús	Controlar aplicació
Actor	Usuari visitant
Objectiu	Establir interacció usuari<>aplicació.
Precondició	L'usuari disposa d'una aplicació oberta i activa.
Postcondició	L'usuari interactua amb l'aplicació.
Escenari principal	L'usuari disposa d'una aplicació oberta i activa de la qual pot visualitzar la seva sortida gràfica. L'aplicació enregistrarà els esdeveniments de teclat i retolí de l'usuari sobre l'aplicació i el sistema respondrà a ells.

TAULA 6.6: Fitxa del cas d'ús: Controlar aplicació

6.4.1.7 Gestionar aplicacions obertes

Cas d'ús	Gestionar aplicacions obertes
Actor	Usuari visitant
Objectiu	Notificar a l'usuari de l'estat de les seves aplicacions.
Precondició	L'usuari disposa de com a mínim, una aplicació oberta.
Postcondició	L'usuari gestiona les aplicacions obertes.
Escenari principal	L'usuari disposa de com a mínim una aplicació oberta, en tot moment pot identificar quines són les aplicacions que es troben en execució i quina és la que es troba en primer pla.

TAULA 6.7: Fitxa del cas d'ús: Gestionar aplicacions obertes

6.4.1.8 Canviar d'aplicació

Cas d'ús	Canviar d'aplicació
Actor	Usuari visitant
Objectiu	Canvia d'aplicació activa en primer pla.
Precondició	L'usuari disposa de més d'una aplicació oberta.
Postcondició	L'usuari canvia d'aplicació en primer pla.
Escenari principal	L'usuari selecciona una de les aplicacions que té en execució. El sistema posa en primer pla a l'aplicació seleccionada sense tancar l'actual.

TAULA 6.8: Fitxa del cas d'ús: Canviar d'aplicació

6.4.1.9 Tancar aplicació

Cas d'ús	Tancar aplicació
Actor	Usuari visitant
Objectiu	Canvia d'aplicació activa en primer pla.
Precondició	L'usuari disposa de, com a mínim, una aplicació oberta.
Postcondició	L'usuari finalitza l'execució d'una aplicació.
Escenari principal	L'usuari decideix finalitzar l'execució d'una aplicació. El sistema finalitza l'execució de l'aplicació i allibera qualsevol recurs utilitzat. En cas de disposar d'altres aplicacions en execució es seleccionarà una de les aplicacions en execució com a activa.

TAULA 6.9: Fitxa del cas d'ús: Tancar aplicació

6.4.1.10 Controlar en pantalla completa

Cas d'ús	Controlar en pantalla completa
Actor	Usuari visitant
Objectiu	Controlar l'aplicació utilitzant la vista en pantalla completa.
Precondició	L'usuari disposa de, com a mínim, una aplicació oberta.
Postcondició	L'usuari interactua amb l'aplicació en pantalla completa.
Escenari principal	L'usuari disposarà d'un botó per a activar la vista en pantalla completa. En prémer el botó, l'aplicació en primar pla esdevindrà en vista de pantalla completa.

TAULA 6.10: Fitxa del cas d'ús: Controlar en pantalla completa

6.4.1.11 Accedir al sistema de fitxers

Cas d'ús	Accedir al sistema de fitxers
Actor	Usuari visitant
Objectiu	Proporcionar a l'usuari accés al sistema de fitxers per a l'aplicació
Precondició	L'usuari disposa de, com a mínim, una aplicació oberta.
Postcondició	L'usuari interactua amb el sistema de fitxers on també hi té accés l'aplicació
Escenari principal	L'usuari disposarà d'un botó per a accedir al sistema de fitxers compartit amb l'aplicació. En prémer el botó, es mostrarà el contingut del sistema de fitxers compartit amb l'aplicació en primar.

TAULA 6.11: Fitxa del cas d'ús: Accedir al sistema de fitxers

6.4.1.12 Visualitzar fitxers disponibles

Cas d'ús	Visualitzar fitxers disponibles
Actor	Usuari visitant
Objectiu	Visualitzar els fitxers i directoris disponibles entre usuari i aplicació.
Precondició	L'usuari disposa de, com a mínim, una aplicació oberta i ha accedit a la vista de fitxers.
Postcondició	L'usuari visualitza el llistat de fitxers i directoris compartits entre usuari i aplicació
Escenari principal	L'usuari ha accedit a la vista del sistema de fitxers. El sistema presenta a l'usuari un llistat de fitxers i directoris disponibles per a l'aplicació. L'usuari pot navegar entre els fitxers i directoris.

TAULA 6.12: Fitxa del cas d'ús: Visualitzar fitxers disponibles

6.4.1.13 Descarregar fitxer

Cas d'ús	Descarregar fitxer
Actor	Usuari visitant
Objectiu	Descarregar un fitxer dels disponibles entre usuari i aplicació.
Precondició	L'usuari disposa de, com a mínim, una aplicació oberta i ha accedit a la vista de fitxers.
Postcondició	L'usuari inicia la descàrrega un fitxer.
Escenari principal	L'usuari ha accedit a la vista del sistema de fitxers i prem un dels fitxers disponibles, aleshores el sistema inicia la descàrrega del fitxer seleccionat.

TAULA 6.13: Fitxa del cas d'ús: Descarregar fitxer

6.4.1.14 Pujar fitxer

Cas d'ús	Pujar fitxer
Actor	Usuari visitant
Objectiu	Pujar un fitxer del sistema de l'usuari cap als fitxer disponibles per a l'aplicació.
Precondició	L'usuari disposa de, com a mínim, una aplicació oberta i ha accedit a la vista de fitxers.
Postcondició	L'usuari puja un fitxer.
Escenari principal	L'usuari ha accedit a la vista del sistema de fitxers i prem el botó de càrrega de fitxers, el sistema presenta a l'usuari un diàleg per a seleccionar un fitxer del seu equip, en fer-ho el es puja el fitxer i es fa disponible per a l'aplicació.

TAULA 6.14: Fitxa del cas d'ús: Pujar fitxer

6.4.1.15 Manteniment del servei

Cas d'ús	Manteniment del servei
Actor	Usuari administrador
Objectiu	Permetre a l'usuari administrador gestionar el servei.
Precondició	L'usuari ha estat identificat i és administrador.
Postcondició	L'usuari accedeix al panell de gestió
Escenari principal	L'usuari ha estat identificat i és administrador, el sistema li presenta l'opció d'accedir al panell de gestió. Al prémer el botó l'usuari és mogut al panell de gestió sense perdre cap de les aplicacions que possiblement tenia obertes.

TAULA 6.15: Fitxa del cas d'ús: Manteniment del servei

6.4.1.16 Manteniment d'usuaris

Cas d'ús	Manteniment d'usuaris
Actor	Usuari administrador
Objectiu	Permetre a l'usuari administrador gestionar els rols dels usuaris.
Precondició	L'usuari ha estat identificat, és administrador i es troba al panell de gestió.
Postcondició	L'usuari gestiona els rols dels usuaris.
Escenari principal	L'usuari administrador es troba al panell de gestió i pot visualitzar de forma paginada els usuaris registrats al sistema. Pot administrar els rols disponibles en el sistema i assignar-los als usuaris.

TAULA 6.16: Fitxa del cas d'ús: Manteniment d'usuaris

6.4.1.17 Manteniment d'aplicacions

Cas d'ús	Manteniment d'aplicacions
Actor	Usuari administrador
Objectiu	Permetre a l'usuari administrador gestionar l'accés i dades de les aplicacions.
Precondició	L'usuari ha estat identificat, és administrador i es troba al panell de gestió.
Postcondició	L'usuari gestiona les aplicacions del sistema.
Escenari principal	L'usuari administrador es troba al panell de gestió i pot visualitzar totes les aplicacions disponibles en el sistema. Des de la vista, pot gestionar el nom i logo visible de les aplicacions i configurar quin tipus d'accés tenen els usuaris sobre l'aplicació. En concret, es permetrà configurar una aplicació per tres nivells de seguretat: 1. Accessible per a tothom. 2. Accessible només per a usuaris identificats. 3. Accessible només per a usuaris que compleixin un rol.

TAULA 6.17: Fitxa del cas d'ús: Manteniment d'aplicacions

6.4.1.18 Visualització de servidors actius

Cas d'ús	Visualització de servidors actius
Actor	Usuari administrador
Objectiu	Permetre a l'usuari administrador visualitzar l'estat del sistema.
Precondició	L'usuari ha estat identificat, és administrador i es troba al panell de gestió.
Postcondició	L'usuari visualitza l'estat del sistema.
Escenari principal	L'usuari administrador es troba al panell de gestió i pot visualitzar un llistat amb tots els servidors d'aplicacions actius en aquell precís moment. El llistat mostrarà l'adreça del servidor i el seu ús actual en percentatge d'ocupació dels seus recursos.

TAULA 6.18: Fitxa del cas d'ús: Visualització de servidors actius

7. Estudis i decisions

7.1 Estudi i elecció de tecnologies per a la virtualització d'aplicacions

Per tal de dur a terme el projecte seguint els requisits definits en el capítol anterior, requerim una tecnologia per a executar aplicacions de forma simultània en un mateix servidor sense que aquestes aplicacions puguin tenir cap afecte sobre el sistema operatiu en què s'allotgen ni entre les execucions simultànies de diferents usuaris. Així doncs, per tal d'aconseguir aquest objectiu, s'analitzen dues possibles implementacions.

- Ús d'execució en "sandbox"
- Ús de contenidors de programari

7.1.1 Execució d'aplicacions en una "sandbox"

7.1.1.1 Característiques de l'execució en una "sandbox"

L'execució de processos dins una "sandbox" [12] proporciona aïllament entre processos, sistema i entorn, de manera que l'aplicació s'executa dins d'un entorn controlat i qualsevol acció realitzada per l'aplicació no té afecte permanent sobre el sistema. L'ús d'aquesta tecnologia permetria disposar d'un nivell de protecció addicional al limitar l'accés i recursos disponibles per a l'aplicació que s'està executant en el seu interior.

Els aspectes clau sobre l'execució dins d'una "sandbox" són:

1. **Aïllament**: Al crear un entorn aïllat de la resta del sistema operatiu i aplicacions, assegurem que qualsevol activitat o canvi realitzat no tindrà cap efecte sobre el sistema operatiu real ni altres processos.
2. **Restricció de recursos**: Ens permet limitar els recursos disponibles per a l'aplicació a executar, aquest fet ens proporciona control sobre que podrà fer l'aplicació i limitació del seu possible impacte.

7.1.1.2 Anàlisi d'integració amb el projecte

Tot i que l'execució d'aplicacions a través de "sandbox" oferirà la seguretat necessària per al sistema, aquesta obre dos problemes a solucionar:

- **Distribució d'aplicacions**
  Recordem que el sistema haurà de disposar de diverses instàncies de servidors d'aplicacions, i que per tant, totes requeriran disposar instal·lades en el sistema les aplicacions a executar. A l'utilitari aquesta tecnologia per a l'aïllament, caldria dissenyar una solució per a la distribució de les aplicacions a executar entre els diversos servidors juntament amb les seves dependències i possibles configuracions.

- **Captura dels gràfics**
  Capturar els gràfics d'una aplicació executada a través d'una "sandbox" complicarà el sistema ja que haurem de dissenyar un sistema capaç de separar els gràfics de cada una de les aplicacions en execució simultània.

7.1.2 Execució d'aplicacions en contenidor de programari

7.1.2.1 Característiques de l'execució en un contenidor de programari

Els contenidors de programari [13] són una forma de virtualització a nivell de sistema operatiu que permeten executar aplicacions de forma aïllada en un entorn contingut. En lloc de virtualitzar tot un sistema operatiu (com fan les màquines virtuals[1]), els contenidors es basen a dividir i compartir els recursos del sistema amfitrió, és a dir, utilitzant el mateix sistema operatiu amfitrió i el seu kernel, es genera un espai virtualitzat on s'hi executen els contenidors.

Els contenidors són segurs per disseny, igual que amb l'execució en "sandbox" els contenidors s'executen de forma aïllada, tant del sistema operatiu amfitrió com d'altres contenidors. A més, la tecnologia de contenidors permet distribuir i replicar imatges de contenidors de forma fàcil.

[1] Una màquina virtual virtualitza el *hardware* i hi executa a sobre un sistema operatiu i les seves aplicacions, oferint un sistema virtual aïllat al real

7.1.2.2 Conceptes i definicions dels contenidor de programari

Per tal de comprendre els conceptes necessaris per a l'execució en contenidors, cal diferenciar els següents conceptes:

1. **Contenidor**: És la tecnologia que fa possible la virtualització d'aplicacions. Aprovisiona el sistema d'encapsulat i l'entorn d'execució de les aplicacions sobre el sistema operatiu amfitrió.

2. **Imatge**: Una imatge és un fitxer de només lectura que emmagatzema la configuració inicial d'un contenidor. La imatge conté les instruccions, programari i llibreries necessàries a executar.

3. **Instància de contenidor**: Una instància de contenidor és una imatge en execució. És a dir, partint d'una imatge inicial, aquesta s'ha muntat i dut a execució. Una imatge pot tenir múltiples instàncies simultànies en execució.

7.1.2.3 Anàlisi d'integració amb el projecte

Els contenidors, per la seva naturalesa, proporcionen tot allò que és necessari per a oferir la capa de seguretat i aïllament del sistema i l'execució d'aplicacions. A més, faciliten la distribució d'aplicacions entre múltiples instàncies de servidors ja que podem empaquetar les diferents aplicacions a executar en imatges de contenidors.

Ara bé, l'ús dels contenidors com a tecnologia de virtualització representa un altre gran repte; l'execució d'aplicacions gràfiques. Per defecte, els contenidors són dissenyats per a contenir el mínim indispensable d'elements, processos i llibreries per a dotar al contenidor de capacitat d'execució, aquest fet es tradueix en la mancança d'un entorn gràfic on poder renderitzar les aplicacions a executar.

7.1.3 Elecció de la tecnologia de virtualització

Després d'analitzar les dues possibilitats d'implementació es veu més potencial a l'ús de contenidors com a sistema de virtualització i aïllament perquè proporcionen les mateixes qualitats que l'ús de "sandbox" però a més ens faciliten la distribució de les aplicacions entre les diferents instàncies de servidors d'aplicacions.

Així doncs, per a realitzar el desenvolupament s'utilitzarà la **virtualització per contenidors**.

Existeixen diverses tecnologies de virtualització per contenidors. En aquest cas, es farà ús de **Docker** com a plataforma de virtualització a causa de la seva senzilla integració a través de l'SDK[2] aprovisionat per la mateixa plataforma. Aquesta ens permet l'orquestració i control dels diferents contenidors a través de codi.

[2]Software Development Kit: Eines i llibreries que faciliten l'ús i modificació del software.

7.2 Estudi i elecció de tecnologies per a la retransmissió i control de l'aplicació

Aquest apartat és, amb diferència, el més important en referència a l'estudi realitzat, degut a les implicacions en la implementació i els avantatges i inconvenients que suposa. Per això es descriuen i comenten les tecnologies estudiades per a dur a terme la retransmissió i control de l'aplicació.

Cal recordar que per tal que alguna d'aquestes tecnologies sigui viable per al projecte, cal que es pugui implementar **sense haver d'instal·lar ni configurar cap programari** addicional a l'equip de l'usuari. Així doncs, les tecnologies subjectes a anàlisi hauran de complir amb aquest requisit.

7.2.1 Retransmissió a través d'HTTP Live Streaming

7.2.1.1 Característiques de la retransmissió amb HLS

HLS (HTTP Live Streaming) [14] és un protocol de transmissió de vídeo desenvolupat per Apple. El seu objectiu principal és permetre el lliurament de contingut multimèdia, com ara vídeos, a través d'internet, de manera eficient i adaptable.

La idea bàsica es basa en dividir el vídeo en fragments. Aquests fragments contenen una durada fixa de contingut (per exemple; uns pocs mil·lisegons). Cada fragment s'envia de forma individual i ordenada.

Quan un client vol reproduir un vídeo utilitzant HLS, fa una sol·licitud HTTP al servidor per obtenir un fitxer de reproducció amb les dades necessàries perquè el client reprodueixi el vídeo de manera correcta.

A mesura que el vídeo es reprodueix, el client continua descarregant i reproduint els fragments conseqüents en ordre. Això crea una experiència de reproducció suau i adaptativa, ja que els fragments tenen una durada fixa i es poden descarregar i reproduir de manera independent.

7.2.1.2 Anàlisi d'integració amb el projecte

El protocol HLS és un estàndard senzill d'implementar, la majoria dels servidors web disponibles l'implementen i aconseguir capturar vídeo i reenviar-lo a través d'aquest servidor és una tasca pràcticament feta, només requereix de la seva correcte configuració i implementació. El problema recau però, en la mateixa naturalesa del protocol, el fet d'haver de fragmentar el contingut en múltiples fragments ocasiona un petit retard en el servidor, ja que aquest, s'ha d'encarregar de rebre uns quants fragments de marge per part de l'origen del contingut ("buffering") per tal de retransmetre'l amb posterioritat.

De forma resumida, el protocol és excel·lent per a la retransmissió de vídeo, tan en diferit com en directe, sempre i quan aquesta retransmissió no requereixi una interacció directa. En el nostre cas, requerim una interacció directa i a més el temps de resposta ha de ser mínim (vegeu secció 5.2.2) i per aquest motiu, no ens és una opció viable. D'altra banda, el protocol és només unidireccional (només envia del servidor al client) hauríem de dissenyar un canal a part per tal de realitzar el control de l'aplicació.

Observi's a continuació una taula resum amb les característiques interessants per a la integració amb el projecte.

Fàcilment integrable al projecte	✓
Bitrate adaptatiu i gestió de qualitat	✓
Baix temps de resposta	✗
Control del servidor integrat	✗

TAULA 7.1: Característiques de l'ús del protocol HLS per a la retransmissió del vídeo

7.2.2 Retransmissió i control a través de l'adaptació de VNC

7.2.2.1 Descripció de la solució

Existeix la possibilitat d'adaptar una tecnologia de control de sessió ja existent com ara VNC però en format web. Recordem que per tal d'utilitzar una sessió amb VNC l'usuari ha de disposar d'un software instal·lat i configurat per a realitzar la connexió i aquest tipus de solució no és vàlida. D'altra banda, existeix la possibilitat d'adaptar i transformar el protocol a través d'un proxy/gateway per tal de realitzar un punt de traducció web. És a dir, afegir una capa més al protocol VNC que ens permeti transformar-lo a un protocol apte per a la comunicació bidireccional a través de la web com són els **WebSockets**.

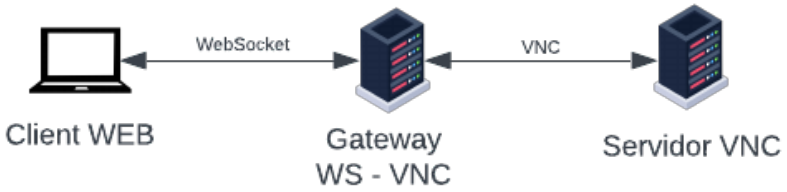

FIGURA 7.1: Connexió Client Web amb Servidor VNC a través de pont WS<>VNC

El problema d'aquesta implementació però, recau en la gran quantitat de feina a realitzar per tal d'implementar aquesta solució ja que la pròpia adaptació podria ser un projecte a part. Per sort, existeix una implementació ja realitzada d'aquest concepte.

El projecte **noVNC** [15] ofereix exactament una solució d'aquest tipus i ens permetria realitzar una connexió VNC utilitzant com a client una aplicació web.

7.2.2.2 Anàlisi d'integració amb el projecte

Al disposar ja d'un paquet que ens permet realitzar la funció de client receptor de vídeo i alhora control del servidor com és noVNC, la integració amb el projecte és bastant senzilla perquè només constarà de la integració del paquet i el disseny del sistema per tal que el suporti. El problema d'aquesta integració però, recau en el mateix protocol utilitzat per a la retransmissió del contingut, en aquest cas RFB [7].

El protocol RFB (usat per VNC) fa possible la retransmissió i control d'un servidor a través d'una connexió TCP/IP. En aquesta connexió, el servidor s'encarrega de capturar imatges de l'estat dels gràfics del servidor i enviar només aquella part que ha canviat respecte al frame anterior. Aquest concepte el fa un protocol lleuger a l'hora de controlar aplicacions amb poc moviment de píxels i pesat a l'hora de canvis sobtats de conjunts grans de píxels.

Al ser un protocol basat en TCP per la transmissió del contingut, aquest és molt susceptible a experimentar retards importants en el mostreig del resultat quan la connexió es realitza a través de xarxes amb un elevat temps de resposta [16].

Així doncs, si volem desenvolupar una solució apta per a qualsevol mena de xarxa i distància, hem de ser conscients que la distància i temps de resposta tindrà un gran impacte negatiu en la velocitat de la retransmissió. D'altra banda, el protocol disposa de compressió i xifrat de la connexió però no de control adaptatiu de la velocitat de connexió.

Observi's a continuació una taula resum amb les característiques interessants per a la integració amb el projecte.

Fàcilment integrable al projecte	✓
Bitrate adaptatiu i gestió de qualitat	✗
Baix temps de resposta	✗
Control del servidor integrat	✓

TAULA 7.2: Característiques de l'ús de VNC per a la retransmissió i control

7.2.3 Retransmissió a través de WebRTC

7.2.3.1 Característiques de la retransmissió amb WebRTC

WebRTC (Web Real-Time Communication) [17] és una tecnologia web disponible en tots els navegadors moderns que fa possible la retransmissió en temps real d'àudio, vídeo i dades, sense la necessitat de complements o programari addicional.

Aquesta tecnologia permet la transmissió de contingut a través del protocol SRTP (sobre UDP, tot i que també es pot utilitzar sobre TCP). Aquest fet ens permet obtenir una baixa latència a l'hora de realitzar connexions en directe. A més, WebRTC proporciona eines per al control adaptatiu de la connexió, la compressió i el xifrat de les dades.

Per tal d'establir una connexió WebRTC és necessària una connexió prèvia amb un servidor de senyalització encarregat d'establir la connexió inicial entre els dos *peers* (client - client o client - servidor). Aquesta connexió es sol realitzar mitjançant **WebSockets** tot i que també hi ha altres alternatives, en aquest cas però, l'ús de WebSockets és adient.

El problema més gran en aquesta tecnologia és la seva orientació al navegador. WebRTC és una tecnologia pensada per a ser usada a través dels navegadors en aplicacions d'usuari i no com a aplicacions de servidor. Per tal de poder usar aquesta tecnologia com a font de contingut per a la retransmissió del vídeo de l'aplicació caldria adaptar la tecnologia per tal de ser usable des d'una aplicació d'escriptori, i això suposa la implementació de tota la tecnologia a través de la seva API per a navegadors. Per sort, existeixen ja paquets que ens permeten realitzar aquesta tasca, un d'ells és el projecte **Pion** [18]. Pion és una implementació completa de l'API de WebRTC utilitzant el llenguatge de programació GO.

7.2.3.2 Anàlisi d'integració amb el projecte

Al disposar ja d'una llibreria que ens permet la integració de WebRTC a través de codi i no des del navegador, les úniques dificultats que presenta aquesta solució és la seva possible complexitat d'integració i ús del protocol. WebRTC requereix l'establiment de la connexió a través d'un servidor intermediari i l'ús del protocol ICE (Interactive Connectivity Establishment) [19] per tal d'obtenir els candidats de connexió possible.

Al ser una tecnologia per a transmissions en temps real, el problema del temps de resposta el tenim pràcticament resolt, a més, la connexió es realitza xifrada, comprimida i amb bitrate adaptatiu.

Observi's a continuació una taula resum amb les característiques interessants per a la integració amb el projecte.

Fàcilment integrable al projecte	-
Bitrate adaptatiu i gestió de qualitat	✓
Baix temps de resposta	✓
Control del servidor integrat	✗

TAULA 7.3: Característiques de l'ús de WebRTC per a la retransmissió

7.2.4 Control de l'aplicació

En cas d'utilitzar un canal que només ens permeti la retransmissió i no el control (com són HLS i WebRTC) caldrà implementar a part un mecanisme de control de l'aplicació (simulació d'esdeveniments de teclat i ratolí). Per a fer-ho caldrà realitzar un mòdul de control del sistema que rebi de l'usuari els esdeveniments de teclat i ratolí i aquests se simulin en el sistema. Per a fer-ho caldrà usar un protocol de transmissió sobre TLS/TCP perquè és necessari que els esdeveniments arribin en ordre i es conservi la seva integritat i confidencialitat. La implementació del protocol haurà de ser manual però per a la simulació de teclat i ratolí podem fer servir les mateixes llibreries del sistema d'X11 [20] o bé usar una llibrària que les implementi (vegeu 7.4.2.3).

7.2.5 Elecció de la tecnologia de retransmissió

Vistes les opcions a implementació, s'escull usar **WebRTC** com a mitjà de transport del vídeo de l'aplicació pels següents motius:

- Ús de transmissió del contingut a través de canal de baixa latència (UDP)
- La tecnologia proporciona un canal xifrat i segur
- Es pot integrar al projecte sense gaire dificultat
- Permet l'adaptació del bitrate segons la qualitat de la connexió

L'única mancança d'aquesta tecnologia és haver d'implementar a part el control del sistema. Tot i això, es decideix dur a terme la implementació del mòdul a part.

7.3 Elecció del sistema de seguretat i autenticació

7.3.1 Seguretat en les connexions

Per tal de mantenir la integritat i confidencialitat de les connexions dels usuaris, caldrà xifrar adientment les connexions. Per a fer-ho es farà ús del protocol HTTPS per a les connexions client-servidor i l'us de Secure Web Socket per a les connexions client-servidor d'aplicacions. En aquest cas, tant HTTPS com Secure Web Socket són la implementació del protocol ordinari sobre la capa de seguretat TLS, aquesta ens garanteix la confidencialitat, integritat i autenticitat de les dades mitjançant certificats.

Per a la retransmissió del vídeo, la mateixa tecnologia de WebRTC s'implementa sobre protocol segur (SRTP). Aquest és la integració del protocol RTP sobre DTLS.

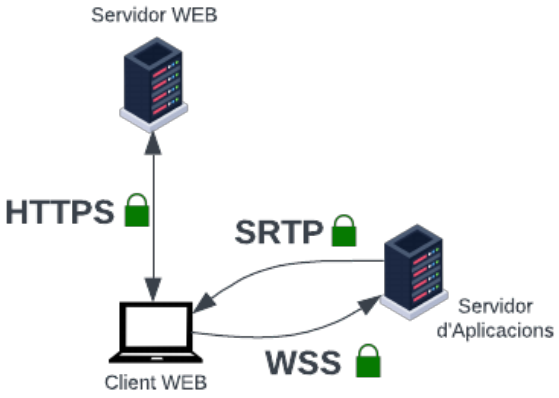

FIGURA 7.2: Xifrat de les connexions

7.3.2 Autenticació i autorització

Per tal d'autenticar als usuaris i a les sol·licituds que aquests puguin realitzar, farem ús dels JSON Web Tokens [21] en la seva versió sense estat. Els JSON Web Token permeten verificar que un contingut ha estat generat per una entitat amb autorització. En la seva modalitat sense estat, aquests es basen en el coneixement d'una clau secreta per les parts autoritzades per tal de crear i validar els tokens. Això resulta en el fet que només les persones o sistemes que tenen aquesta clau secreta poden generar i comprovar l'autenticitat dels tokens.

Quan un usuari s'autentica al sistema, se li assigna un JWT que conté informació sobre aquest usuari, com ara el seu identificador o els permisos que té. Aquest JWT és com una targeta d'identitat que pot ser presentada posteriorment per l'usuari per accedir a determinades funcionalitats o recursos.

7.4 Elecció de tecnologies, llenguatges i llibreries

Seguidament s'enumeraran les tecnologies, llenguatges i llibreries usades per al desenvolupament del projecte. Per a fer-ho, es dividirà aquest apartat en cada una de les parts requerides per l'arquitectura (vegeu capítol 8)

7.4.1 Servidor d'aplicacions

7.4.1.1 Situació

El servidor d'aplicacions és l'encarregat d'orquestrar les instàncies dels contenidors d'aplicació a executar. Un servidor executa un sol servidor d'aplicacions i aquest s'encarrega de gestionar totes les connexions entrants.

7.4.1.2 Llenguatges

Tota la programació del servei s'ha realitzat a través del llenguatge de programació **GO**. Go és un llenguatge de programació concurrent, eficient i ofereix un sistema de gestió de memòria automàtic i un rendiment òptim. Go és ideal per a programes paral·lels i sistemes distribuïts, a més, és una bona opció d'integració amb sistemes d'orquestració de docker ja que es disposa de la pròpia API en aquest llenguatge.

7.4.1.3 Llibreries

- Pròpies del llenguatge:
    1. **fmt**: Implementació i format d'I/O anàloga a la implementació en C.
    2. **log**: Permet mostrar missatges en forma de log.
    3. **strings**: Manipulació d'strings.
    4. **strconv**: Conversió de tipus bàsics a string.
    5. **bufio**: Implementació buffered I/O.
    6. **bytes**: Manipulació de bytes i slices.
    7. **encoding/base64**: Codificació en format base64.
    8. **encoding/json**: Codificació en format JSON.
    9. **crypto/sha256**: Implementació de l'algorisme de hash SHA256.
    10. **path/filepath**: Manipulació de paths del sistema.
    11. **os**: Capa d'abstracció per al control del sistema operatiu subjacent.

12. **os/exec**: Execució de comandes externes.
13. **net**: Implementació de sockets TCP i UDP.
14. **net/http**: Implementació de client i servidor HTTP.

- Externes:
  1. **go-ini**: Implementació i manipulació de fitxer .ini per a la configuració. `https://github.com/go-ini/ini`
  2. **docker**: API de Docker per al control de l'ecosistema de docker. `https://github.com/docker/docker`
  3. **golang-jwt**: Implementació en GO de JSON Web Token. `https://github.com/dgrijalva/jwt-go`
  4. **websocket**: Implementació en GO de WebSockets. `https://github.com/gorilla/websocket`
  5. **gopsutil/cpu**: Implementació en GO de psutil (Process and system utilities). `https://github.com/shirou/gopsutil/cpu`

7.4.1.4 Tecnologies / Aplicacions

1. **Docker**: S'usa Docker com a motor de virtualització i contenció de les aplicacions a executar.

7.4.2 Virtualitzador d'aplicació

7.4.2.1 Situació

Es descriu el programari i tecnologies usades dins la instància base de docker per a l'execució d'una aplicació. Dins la instància s'ha de realitzar l'execució de l'aplicació, control i retransmissió d'aquesta.

7.4.2.2 Llenguatges

D'igual manera que amb el servidor d'aplicacions, tota la programació s'ha realitzat a través del llenguatge de programació **GO**. Aquest ens permet generar un binari natiu lleuger i reduir la mida inicial de la imatge del contenidor.

7.4.2.3 Llibreries

- Pròpies del llenguatge: S'usen les mateixes llibreries que les enumerades en l'apartat 7.4.1.3
- Externes:
  1. **robotgo**: Implementació i de les llibreries d'X11 per al control de teclat i ratolí a través de GO.
     `https://github.com/go-vgo/robotgo`
  2. **pion/webrtc**: Implementació de l'API de WebRTC a través de GO.
     `https://github.com/pion/webrtc`
  3. **pion/rtp**: Implementació del protocol RTP a través de GO.
     `https://github.com/pion/rtp`

7.4.2.4 Tecnologies / Aplicacions

1. **X Virtual Frame Buffer**: Servidor d'X virtualitzat i en memòria [22]. Permet el renderitzat de gràfics completament en memòria sense la necessitat de cap pantalla. Amb ell aconseguim afegir una pantalla virtual a la instància de docker per tal d'aconseguir el renderitzat de gràfics.
2. **I3**: Gestor de finestres lleuger en mode rajoles (*tiled* en anglès) [23]. Ens permet afegir a la pantalla virtual un gestor de finestres que controli la posició de les aplicacions gràfiques.
3. **FFMPEG:** Software que, entre d'altres utilitats, ens permet la captura i retransmissió en temps real dels gràfics d'una pantalla a través d'RTP [24].

7.4.3 Servidor web i de control

7.4.3.1 Situació

Es descriu el programari i tecnologies usades per part del servidor central. Aquest és l'encarregat de servir la web, proporcionar l'API i la connexió inicial amb els servidors d'aplicacions.

7.4.3.2 Llenguatges

El desenvolupament d'aquest servei es realitza mitjançant **JavaScript** i **NodeJS**. NodeJS és en entorn d'execució de codi javascript en temps real, el servei és asíncron i amb E/S orientada a esdeveniments.

7.4.3.3 Llibreries

1. **express**: Framework web minimalista i eficient que facilita la creació d'APIs i aplicacions web.
2. **body-parser**: Utilitzat per analitzar el cos de les sol·licituds HTTP.
3. **cors**: Proporciona un mecanisme de seguretat en el navegador per permetre sol·licituds HTTP des de diferents dominis.
4. **dotenv**: Usat per a carregar variables d'entorn de l'aplicació.
5. **jsonwebtoken**: Implementació en NodeJS dels JSON Web Token.
6. **mysql2**: Utilitzat per la connexió i control de la base de dades MySQL.

7.4.3.4 Tecnologies / Aplicacions

S'usarà Docker per a la implantació del servei.

7.4.4 Base de Dades

S'usarà una base de dades **MySQL** per tal de mantenir les dades del servei.

7.4.5 Registre d'imatges

S'usarà **Docker Registry** per a mantenir les imatges de les aplicacions disponibles.

7.4.6 Client WEB

El desenvolupament del client web serà completament nadiu amb el navegador. No s'usarà cap llibreria ni *plugin* extern. Qualsevol funcionalitat serà implementada a través de la mateixa API de JavaScript.

Per tal de facilitar el desenvolupament, s'usarà **Angular** com a framework de front-end i l'ús de la llibreria de components **Angular Material UI** per a facilitar el desenvolupament d'elements d'interacció amb l'usuari.

8. Disseny de l'aplicació de virtualització

8.1 Disseny de l'arquitectura del sistema

Per tal de satisfer les necessitats del sistema, es dissenya una arquitectura orientada a serveis completament modular i independent. A continuació es detalla el disseny de cadascuna d'aquestes parts.

8.1.1 Disseny de l'entorn del virtualitzador

8.1.1.1 Descripció

L'entorn del virtualitzador seran les eines i programes utilitzats i preparats per a establir l'entorn bàsic per a la virtualització d'una aplicació. Aquest entorn es trobarà dins d'un contenidor de Docker i haurà de satisfer els següents requisits:

- Permetre l'execució d'una aplicació gràfica per a un usuari remot.
- Simular esdeveniments de teclat i ratolí per tal de ser controlada.
- Capturar els gràfics i enviar-los a l'usuari.
- Permetre a l'administrador distribuir noves aplicacions de forma senzilla.

8.1.1.2 Disseny

A continuació es mostra el disseny de processos dins del contenidor de docker per tal de complir amb els requisits (vegeu figura 8.1). Cadascun dels blocs mostrats representa un procés en execució simultània per tal d'acomplir els resultats.

1. **Simulador d'inputs**: Procés encarregat de rebre missatges amb les accions a realitzar. El procés rebrà a través d'una connexió per WebSocket, diferents esdeveniments de teclat i ratolí que simularà en el sistema per tal de controlar l'aplicació.

2. **Aplicació**: Bloc que representa l'aplicació en execució d'aquesta instància. La imatge base del contenidor no contindrà cap aplicació instal·lada però si tota l'estructura de processos muntada i configurada per tal que l'administrador pugui instal·lar-hi aplicacions amb facilitat.

3. **XVFB**: Serà el procés encarregat de crear la pantalla virtual per tal que l'aplicació hi pugui renderitzar els gràfics i posteriorment capturar-los.

4. **i3wm**: Serà el procés gestor de finestres, encarregat de gestionar la forma i posició de les finestres que pugui obrir l'aplicació.

5. **FFMPEG**: Procés encarregat de capturar en temps real els gràfics renderitzats en la pantalla virtual i generar un flux de vídeo per RTP per tal de ser consumit.

6. **Gateway RTP-WebRTC**: Procés encarregat de consumir el flux de vídeo RTP i transformar-lo en un flux apte per a la connexió WebRTC amb l'usuari.

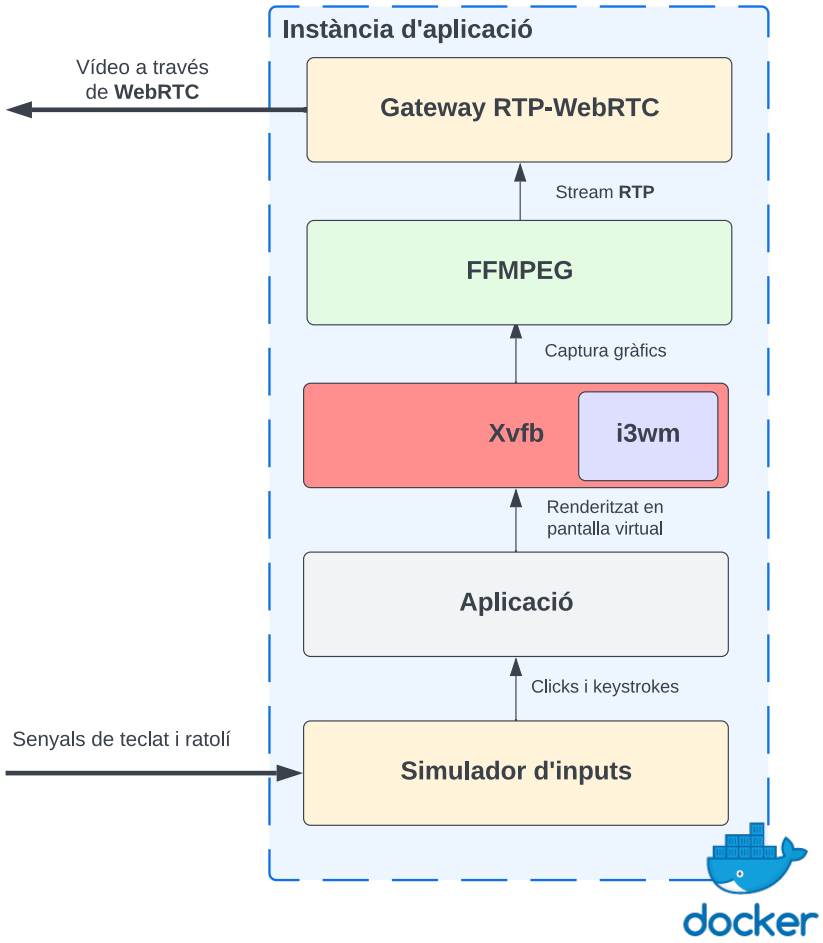

FIGURA 8.1: Disseny de l'entorn del virtualitzador (contenidor de docker)

8.1.2 Disseny del servidor d'aplicacions

8.1.2.1 Descripció

El servidor d'aplicacions serà el servei encarregat d'orquestrar les múltiples instàncies d'aplicacions actives en el sistema (instàncies de docker). El servei permetrà als usuaris autoritzats iniciar una instància d'aplicació i accedir als seus recursos. Així doncs, els requisits que haurà de complir aquest servei són:

- Permetre l'inici d'instàncies de docker si l'usuari està autoritzat a fer-ho.
- Dotar d'accés al sistema de fitxers a l'usuari i a la instància d'aplicació.
- Realitzar de *proxy* entre usuari i instància d'aplicació per al reenviament d'esdeveniments.
- Realitzar la multiplexació de connexions entre usuaris i instàncies.
- Orquestrar les diferents instàncies de docker actives.
- Mantenir les imatges dels contenidors actualitzades.

8.1.2.2 Disseny

A continuació es mostra el disseny de l'arquitectura del servidor d'aplicacions (vegeu figura 8.2).

1. **Servidor d'aplicacions**: Serà el procés encarregat d'orquestrar les instàncies de docker, així com arrancar-les i dotar a l'usuari de l'accés a elles mitjançant una connexió inversa. També muntarà sistemes de fitxers virtuals per a aquestes instàncies per tal que els usuaris hi puguin accedir. D'altra banda, portarà la gestió del repositori local d'imatges de contenidors disponibles per tal d'estar actualitzada en tot moment amb el servidor central.
2. **Recuros**: Representació dels volums muntats per a l'accés tant d'usuaris com d'instàncies de docker. Cada recurs virtual serà accessible només per l'usuari que l'ha generat i només per la instància d'aplicació que li pertoca.

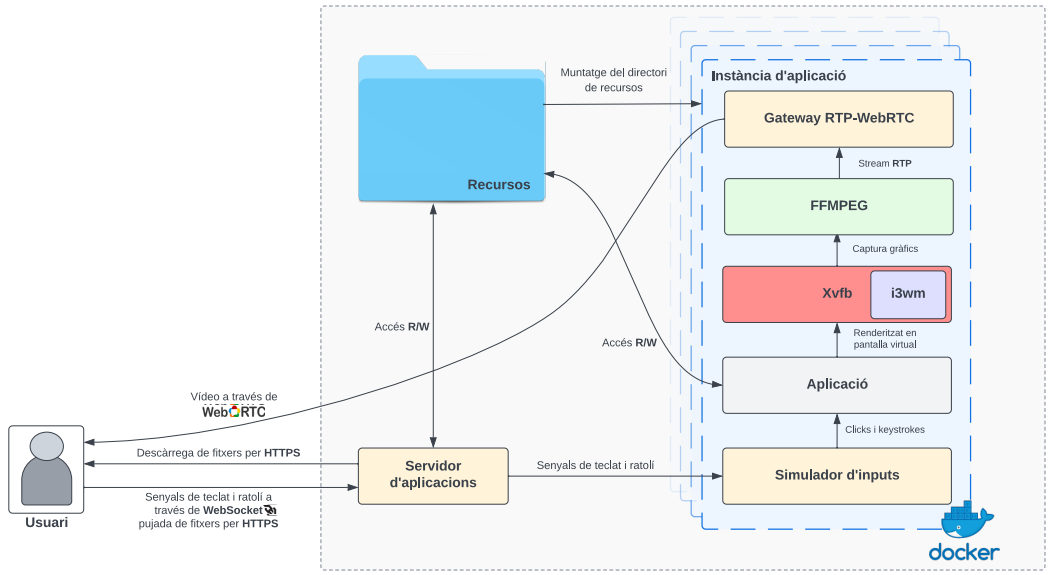

FIGURA 8.2: Disseny de l'arquitectura del servidor d'aplicacions

8.1.3 Disseny del servidor central i persistència de dades

8.1.3.1 Descripció

El servidor central serà l'encarregat d'actuar com a servidor web i servidor de control entre usuaris i les múltiples instàncies de servidor d'aplicacions. Aquest servei permetrà als usuaris accedir a l'aplicació web desenvolupada i en durà el seu control. D'altra banda, el servei també durà a terme la gestió i control de les diverses instàncies dels servidors d'aplicacions de manera que serà l'encarregat de gestionarne el seu accés per part dels usuaris. Així doncs, els requisits que haurà de complir aquest servei són:

- Servir l'aplicació web a l'usuari.
- Autenticar i autoritzar els usuaris tant a l'aplicació com als diversos servidors d'aplicacions.
- Gestionar la configuració del servei i aplicacions instal·lades.

8.1.3.2 Disseny

A continuació es mostra el disseny de l'arquitectura del servidor central (vegeu figura 8.3).

1. **Servidor central**: Serà el servei encarregat de servir la web i dur a terme la gestió del servei. Per dur a terme la seva comesa, farà ús d'autenticació i autorització mitjançant de tokens sense estat, permetent així, disposar de múltiples instàncies de servidor central.

2. **BDD**: Representa la instància de base de dades del sistema. Contindrà les dades referents a usuaris, aplicacions i configuració del servei.

3. **Registre d'imatges**: Serà el servei encarregat de contenir les múltiples imatges de contenidors disponibles per al sistema. Cada imatge representarà una aplicació disponible per als servidors d'aplicacions i subjectes a configuració pel servei.

Per tal de facilitar el desenvolupament i posteriorment la implantació. Cadascun dels serveis necessaris serà desenvolupat com a un contenidor de docker. Permetent d'aquesta manera la lliure implantació d'aquests, tant en una sola màquina com en diverses.

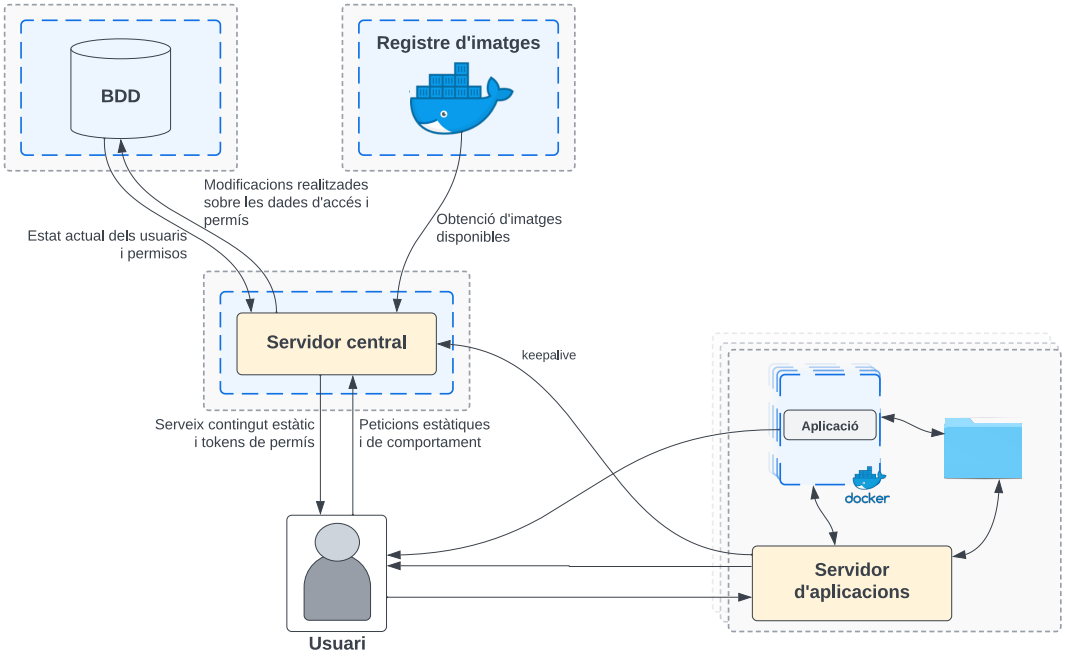

FIGURA 8.3: Disseny de l'arquitectura del servidor central

8.1.4 Arquitectura del sistema complet

Tal com s'ha comentat en les seccions anteriors, cadascun dels serveis dissenyats és completament modular i independent de la resta de manera que la seva implantació podrà ser monolítica (en una sola màquina) o bé completament distribuïda. El sistema està pensat de manera que cadascun dels elements pugui ser replicat N vegades. Així doncs, podem tenir una sola màquina física que integri tots els serveis o bé N màquines integrant el servidors centrals, per M servidors d'aplicacions on cadascun d'aquests servidors executa T instàncies d'aplicacions. De la mateixa manera, els servidors de dades també poden ser replicats ja que els serveis que els implementen també ho permeten.

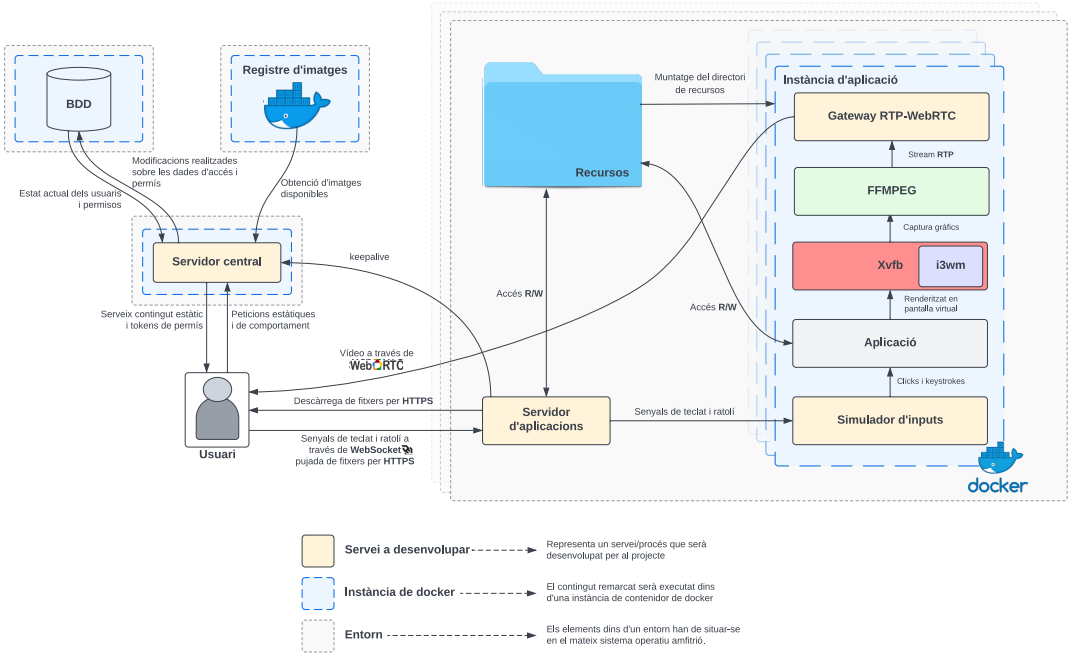

FIGURA 8.4: Disseny de l'arquitectura del sistema

8.2 Disseny del protocol/seqüència de connexió i control

Al ser una arquitectura distribuïda, cal plantejar un protocol de connexió i interacció amb el sistema que permeti a l'usuari establir la connexió i execució d'una aplicació i alhora controlar-la. Per a fer-ho, es dissenya un protocol/seqüència d'accions a realitzar que es dividirà en tres etapes.

1. Inicialització
2. Inici d'execució
3. Control de l'aplicació

La següent figura mostra una simplificació de la seqüència d'accions a realitzar. En ella es representen les connexions realitzades entre client i servidor de control, client i les múltiples instàncies de servidor d'aplicació i client i instància concreta d'aplicació.

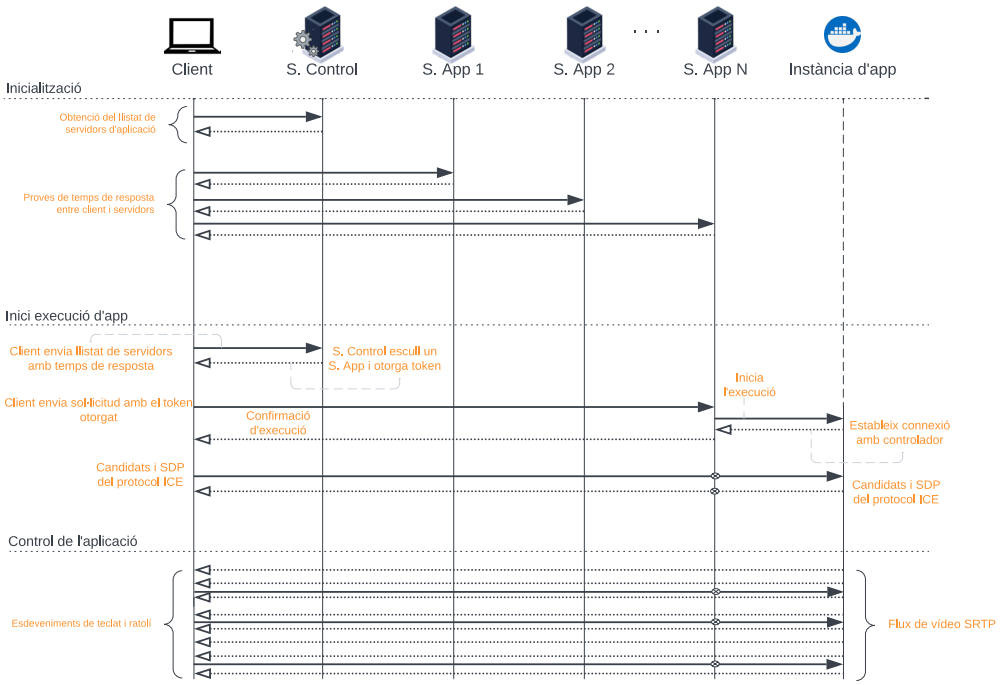

FIGURA 8.5: Seqüència de connexió i control

8.2.1 Inicialització

L'etapa d'inicialització succeeix en el primer instant que l'usuari accedeix a l'aplicació web i es realitza de forma asíncrona. El seu procediment és el següent:

1. El client sol·licita al servidor de control el llistat de servidors d'aplicacions disponibles.

2. Servidor de control respon amb el llistat de servidors d'aplicacions.

3. El client, realitza de forma asíncrona un test de connexió amb cadascun dels servidors d'aplicacions per tal d'anotar el temps de resposta.

Un cop finalitzada l'etapa d'inicialització, el client emmagatzema en memòria els temps de resposta i queda a l'espera que l'usuari decideixi iniciar l'execució d'una aplicació.

8.2.2 Inici d'execució d'aplicació

Arribat el punt en què l'usuari decideix iniciar l'execució d'una aplicació, s'inicia la seqüència d'arrancada. Aquesta consta de les següents passes:

1. El client envia al servidor de control el llistat de servidors d'aplicacions amb els seus respectius temps de resposta.

2. El servidor comprova que l'usuari disposa de permís d'execució i escull un d'entre tots els servidors, tenint en compte els temps de resposta i l'ocupació de cada servidor. Un cop escollit, genera un *token* d'accés i l'envia a l'usuari.

3. El client rep el *token* del servidor de control i procedeix a inicialitzar la connexió amb el servidor d'aplicacions que se li ha assignat. En la sol·licitud de connexió hi envia el *token* generat pel servidor de control.

4. El servidor d'aplicacions rep el *token* i el valida, seguidament inicia l'execució del contenidor de l'aplicació seleccionada. Veure 8.2.2.1.

5. Un cop iniciada l'execució de l'aplicació, la instància de docker ho notifica al servidor d'aplicacions i aquest ho reenvia cap al client. En aquest instant, el servidor d'aplicacions procedeix a actuar com a proxy[1] entre la connexió de client i instància d'aplicació.

6. El client, en rebre la confirmació d'execució, inicia el protocol ICE amb la instància de l'aplicació utilitzant el servidor d'aplicacions com a intermediari.

7. Un cop client i instància d'aplicació han decidit com establir la connexió, s'intercanvien el seu SDP (*Session Description Protocol*).

Un cop completada la seqüència, s'haurà establert una connexió utilitzant WebRTC que permetrà a la instància d'aplicació retransmetre el vídeo capturat.

8.2.2.1 Inici de contenidor d'aplicació

Quan un servidor d'aplicacions rep l'ordre vàlida d'iniciar l'execució d'una instància d'aplicació, segueix els següents passos:

1. El servidor d'aplicacions obre un port d'escolta local i el registra

2. Inicia l'execució del contenidor de docker amb l'aplicació i hi assigna el número de port registrat.

[1] Un proxy és un servidor encarregat de realitzar d'intermediari entre les peticions d'un client i un altre servidor.

3. Un cop la instància d'aplicació s'ha encès, estableix connexió amb el port obert i així el servidor d'aplicacions s'assabenta del correcte inici de l'aplicació.

La connexió establerta serà aprofitada per a la retransmissió d'informació entre client i instància d'aplicació.

8.2.3 Control de l'aplicació

Un cop la connexió entre client i instància d'aplicació ja s'ha establert, s'estableix una comunicació asíncrona entre client i instància d'aplicació.

- La instància d'aplicació envia de forma constant un flux de vídeo amb els gràfics de l'aplicació.
- El client, quan succeeix alguna acció (moviment de ratolí, *click*, introducció de tecla...) envia l'esdeveniment cap a la instància d'aplicació utilitzant com a intermediari la connexió feta amb el servidor d'aplicacions.

8.3 Disseny de la base de dades

Per tal de satisfer les necessitats del sistema, es dissenya una base de dades minimalista que contingui exclusivament els camps necessaris per a cobrir els requisits. Amb la base de dades es pretén dur el registre d'usuaris i aplicacions del sistema juntament amb els permisos d'accés que disposen aquests usuaris sobre les aplicacions.

8.3.1 Model relacional

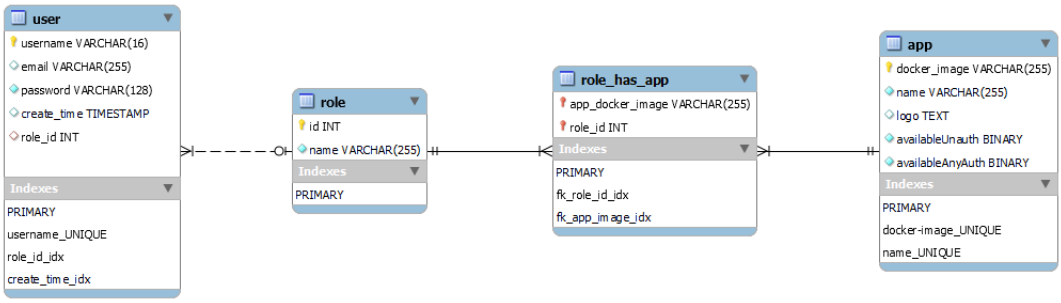

FIGURA 8.6: Model relacional

8.4 Disseny de l'API del servidor de control

A continuació es detalla el disseny plantejat per al servidor web. En concret es documenta cadascun dels recursos (*endpoints*) a desenvolupar de l'API per tal de complir amb els requisits de l'aplicació. El disseny de l'API seguirà el patró **MVC** (Model - Vista - Controlador) per tal d'oferir una estructura organitzativa que ajuda a millorar la manejabilitat, l'escalabilitat i la reutilització del codi.

8.4.1 Recurs: usuari

8.4.1.1 Registre d'usuari

Descripció	Registre d'usuari al sistema i retorn de token d'autenticació.
Recurs	/user/signup
Mètode	POST

Capçalera	{
	"Content-Type": "application/json;"
	}

Cos	{
	"username": string,
	"email": string,
	"password": string,
	"password2": string
	}

Requisits	- password és igual a password2 i la seva longitud és superior a 7 caràcters. - email és un correu electrònic vàlid. - username i email no existeix al sistema

Resposta	{
	"token": string
	}

Resposta alternativa	{
	"error": string
	}

TAULA 8.1: Recurs: Registre d'usuari

8.4.1.2 Identificació d'usuari

Descripció	Identificació d'usuari al sistema i retorn de token d'autenticació.
Recurs	/user/login
Mètode	POST

Capçalera	{
	"Content-Type": "application/json;"
	}

Cos	{
	"username": string,
	"password": string,
	}

Resposta	{
	"token": string
	}

Resposta alternativa	{
	"error": string
	}

TAULA 8.2: Recurs: Identificació d'usuari

8.4.1.3 Obtenció d'usuaris

Descripció	Obtenció del llistat d'usuaris registrats al sistema.
Recurs	/user/all
Mètode	GET

Capçalera	{
	"Authorization": "Bearer **<token>**"
	}

Requisits	- El token és vàlid i pertany a l'administrador

Resposta	[
	{
	"username": string,
	"email": string,
	"role": int,
	"createTime": timestamp
	}
	]

Resposta alternativa	{
	"error": string
	}

TAULA 8.3: Recurs: Obtenció d'usuaris

8.4.1.4 Actualització rol d'usuari

Descripció	Actualització del rol d'un usuari.
Recurs	/user/{username}/{role-id}
Mètode	PUT

Capçalera	{
	"Authorization": "Bearer **<token>**"
	}

Requisits	- El token és vàlid i pertany a l'administrador - Existeix usuari amb nom d'usuari {username} - Existeix rol amb id {role-id}

Resposta	{
	"success": bool
	}

Resposta alternativa	{
	"error": string
	}

TAULA 8.4: Recurs: Actualització del rol d'un usuari

8.4.2 Recurs: rol

8.4.2.1 Creació d'un nou rol

Descripció	Creació d'un nou rol al sistema.
Recurs	/role
Mètode	POST

Capçalera	{
	"Content-Type": "application/json;"
	"Authorization": "Bearer **<token>**"
	}

Cos	{
	"rolName": string,
	}

Requisits	- El token és vàlid i pertany a l'administrador - rolName conté almenys 1 caràcter

Resposta	{
	"id": int,
	"name": string
	}

Resposta alternativa	{
	"error": string
	}

TAULA 8.5: Recurs: Creació d'un nou rol

8.4.2.2 Actualització de nom d'un rol

Descripció	Actualització de nom d'un rol.
Recurs	/role
Mètode	PUT

Capçalera	{
	"Content-Type": "application/json;"
	"Authorization": "Bearer **<token>**"
	}

Cos	{
	"id": int,
	"name": string
	}

Requisits	- El token és vàlid i pertany a l'administrador - Existeix un rol amb l'id entrat - name conté almenys 1 carácter

Resposta	{
	"success": bool,
	}

Resposta alternativa	{
	"error": string
	}

TAULA 8.6: Recurs: Actualització de nom d'un rol

8.4.2.3 Eliminació d'un rol

Descripció	Eliminació d'un rol al sistema.
Recurs	/role/{id}
Mètode	DELETE

Capçalera	{
	"Authorization": "Bearer **<token>**"
	}

Requisits	- El token és vàlid i pertany a l'administrador - Existeix un rol amb l'id {id} - El rol no està en ús per cap usuari ni aplicació

Resposta	{
	"success": bool,
	}

Resposta alternativa	{
	"error": string
	}

TAULA 8.7: Recurs: Eliminació d'un rol

8.4.2.4 Obtenció del llistat de rols

Descripció	Obtenció del llistat de rols disponibles.
Recurs	/role/all
Mètode	GET

Capçalera	{
	"Authorization": "Bearer **<token>**"
	}

Requisits	- El token és vàlid i pertany a l'administrador

Resposta	[
	{
	"id": int,
	"name": string,
	}
	]

Resposta alternativa	{
	"error": string
	}

TAULA 8.8: Recurs: Obtenció del llistat de rols

8.4.3 Recurs: aplicació

8.4.3.1 Obtenció del llistat d'aplicacions disponibles per a l'usuari

Descripció	Obtenció del llistat d'aplicacions disponibles per a l'usuari.
Recurs	/app/my-apps
Mètode	GET

Capçalera	{
	"Authorization": "Bearer **<token>**" ← *opcional, buit si no està identificat*
	}

Resposta	[
	{
	"image": string,
	"name": string,
	"ico": string,
	}
	]

Resposta alternativa	{
	"error": string
	}

TAULA 8.9: Recurs: Obtenció del llistat d'aplicacions disponibles per a l'usuari

8.4.3.2 Sol·licitud d'accés a una aplicació

Descripció	Sol·licitud d'accés a l'execució d'una aplicació.
Recurs	/app/{tag}
Mètode	POST

Capçalera	{
	"Content-Type": "application/json;"
	"Authorization": "Bearer **<token>**" ← *opcional, buit si no està identificat*
	}

Cos	{
	"servers": [{
	"server": string,
	"msToResponse": int
	}]
	}

Requisits	- servers conté com a mínim un candidat - l'usuari disposa de permís per a l'execució de l'aplicació amb nom {tag}

Resposta	{
	"server": string
	"token": string,
	}

Resposta alternativa	{ "error": string }

TAULA 8.10: Recurs: Sol·licitud d'accés a una aplicació

8.4.3.3 Obtenció del llistat d'aplicacions per configuració

Descripció	Obtenció del llistat d'aplicacions disponibles en el sistema per a configurar.
Recurs	/app/admin-list
Mètode	GET
Capçalera	{ "Authorization": "Bearer **<token>**" }
Requisits	- El token és vàlid i pertany a l'administrador

Resposta	[
	{
	"docker_image": string,
	"name": string,
	"logo": string,
	"pendingConfig": bool,
	"availableUnauth": bool,
	"availableAnyAuth": bool,
	"roles": [int],
	}
	]

Resposta alternativa	{ "error": string }

TAULA 8.11: Recurs: Obtenció del llistat d'aplicacions per configuració

8.4.3.4 Actualització de la configuració d'una aplicació

Descripció	Actualització de la configuració d'una aplicació
Recurs	/app
Mètode	POST

Capçalera	{
	"Content-Type": "application/json;"
	"Authorization": "Bearer **<token>**"
	}

Cos	{
	"docker_image": string
	"name": string
	"logo": string
	"availableUnauth": bool,
	"availableAnyAuth": bool,
	"roles": [int],
	}

Requisits	- El token és vàlid i pertany a l'administrador - Existeix una aplicació al sistema amb tag docker_image - Els camps name i logo no son buits

Resposta	{
	"success": bool
	}

Resposta alternativa	{
	"error": string
	}

TAULA 8.12: Recurs: Actualització de la configuració d'una aplicació

8.4.4 Recurs: servidor

8.4.4.1 Subscripció d'un servidor d'aplicacions al servei

Descripció	Subscripció d'un servidor al servei i manteniment d'estat
Recurs	/server/keepalive
Mètode	POST

Capçalera	{
	"Content-Type": "application/json;"
	"Authorization": "Bearer **<token>**"
	}

Cos	{
	"ip": string
	"ram": string
	"cpu": string
	}

Requisits	- El token és vàlid i signat per l'administrador
Resposta	STATUS: 200 OK
Resposta alternativa	{ "error": string }

TAULA 8.13: Recurs: Subscripció d'un servidor d'aplicacions al servei

8.4.4.2 Obtenció de servidors d'aplicacions disponibles (vista administració)

Descripció	Obtenció de servidors d'aplicacions disponibles (vista administració)
Recurs	/server
Mètode	GET

Capçalera	{
	"Authorization": "Bearer **<token>**"
	}

Requisits	- El token és vàlid i pertany a l'administrador

Resposta	[
	{
	"ip": string
	"ram": string
	"cpu": string
	}
	]

Resposta alternativa	{
	"error": string
	}

TAULA 8.14: Recurs: Obtenció de servidors d'aplicacions disponibles (vista administració)

8.4.4.3 Obtenció de servidors d'aplicacions disponibles (vista usuaris)

Descripció	Obtenció de servidors d'aplicacions disponibles (vista usuaris)
Recurs	/server/getIps
Mètode	GET
Resposta	[string]

Resposta alternativa	{
	"error": string
	}

TAULA 8.15: Recurs: Obtenció de servidors d'aplicacions disponibles (vista usuaris)

8.5 Disseny de la interfície d'usuari

Seguidament es mostra el disseny de la interfície d'usuari per l'aplicació web.

8.5.1 Vista principal

En la vista principal de l'aplicació web s'efectuen totes les accions requerides per usuaris visitants i identificats, a més, es permet l'accés a les altres seccions de l'aplicació com són la identificació o el panell d'administració.

8.5.1.1 Visió inicial

La plana d'inici té com a objectiu mostrar a l'usuari el llistat d'aplicacions disponibles i permetre'n l'execució. Observi's la figura 8.7 per tal d'apreciar els elements principals en el moment de càrrega de la plana.

1. Accedir a la identificació
2. Canviar el format de la vista (amagar llistat d'aplicacions)
3. Llistat d'aplicacions disponibles per a l'usuari

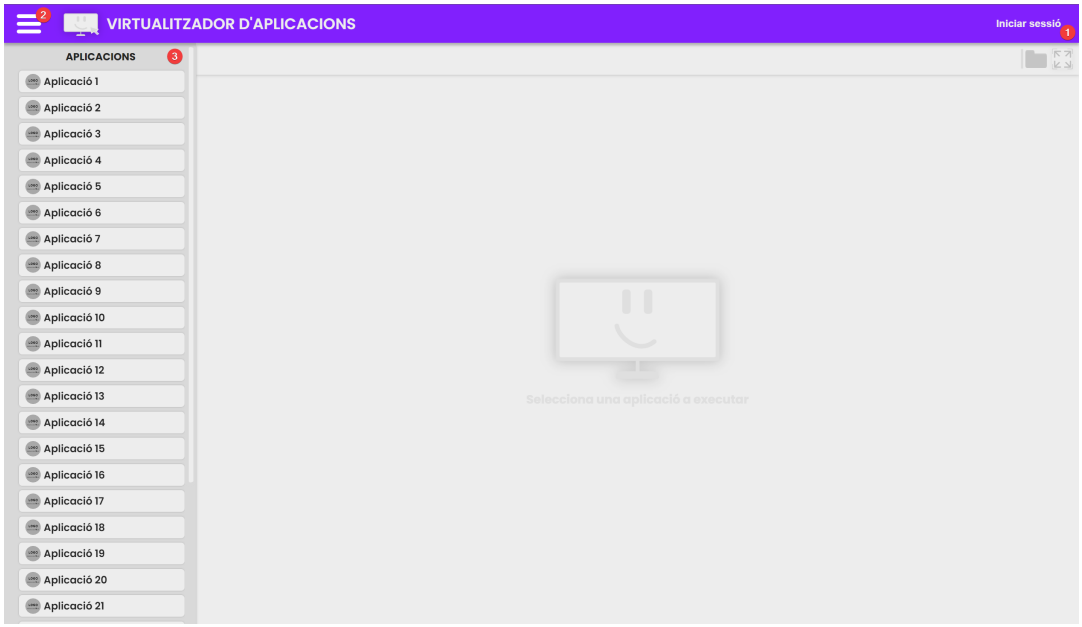

FIGURA 8.7: Plana d'inici, visió inicial

8.5.1.2 Execució d'aplicació

Un cop l'usuari decideix iniciar l'execució d'una aplicació, la vista inicial canvia a la vista d'execució d'aplicació. Des d'aquest instant es pot apreciar en qualsevol moment l'aplicació que es troba en primer pla i la resta d'aplicacions en execució. Sempre que hi hagi alguna aplicació en execució s'habilitaran els següents botons (observi's la figura 8.8):

1. Obrir modal d'interacció amb el sistema de fitxers de l'aplicació
2. Establir vista en pantalla completa de l'aplicació en primer pla

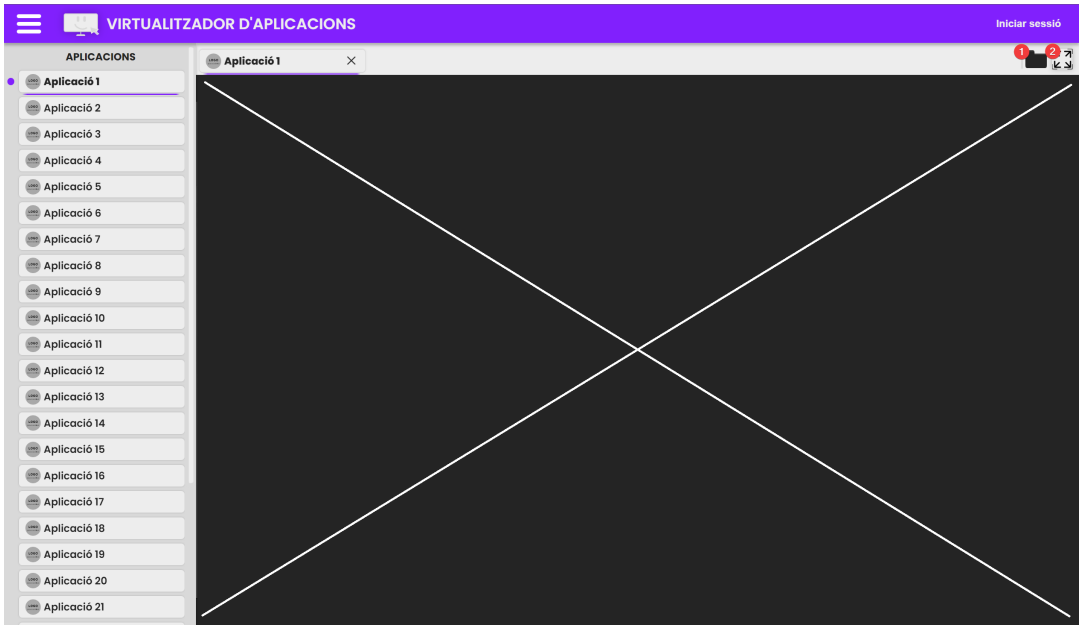

FIGURA 8.8: Plana d'inici, execució d'aplicació

8.5.1.3 Modal d'interacció amb el sistema de fitxers

En cas de voler interactuar amb el sistema de fitxers, s'obrirà un modal mostrant els directoris i fitxers de l'àrea compartida de l'aplicació. Observi's la següent figura per a més detall.

1. Modal d'interacció amb el sistema de fitxers de l'aplicació en primer pla
2. Visió de directoris disponibles
3. Visió de fitxers disponibles
4. Pujar un fitxer local a l'espai compartit

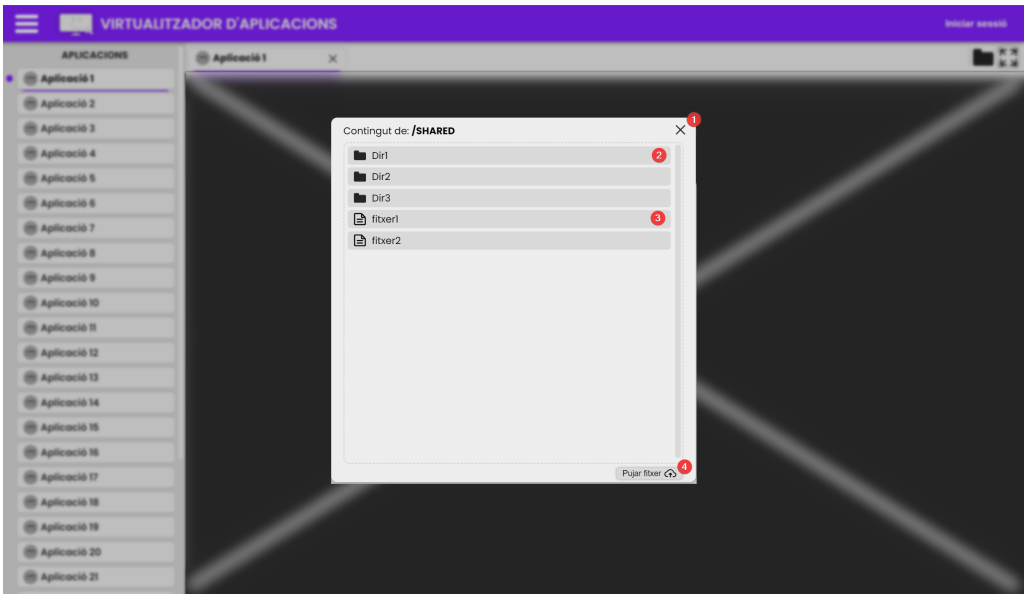

FIGURA 8.9: Plana d'inici, modal d'interacció amb el sistema de fitxers

8.5.1.4 Menú contextual

Si l'usuari es troba a la plana inicial però ja ha estat identificat, l'accés d'inici de sessió es veu modificat per un menú contextual d'opcions. Vegeu figura 8.10.

1. Accés al panell d'administració *(Només visible si l'usuari és administrador)*
2. Tancar sessió

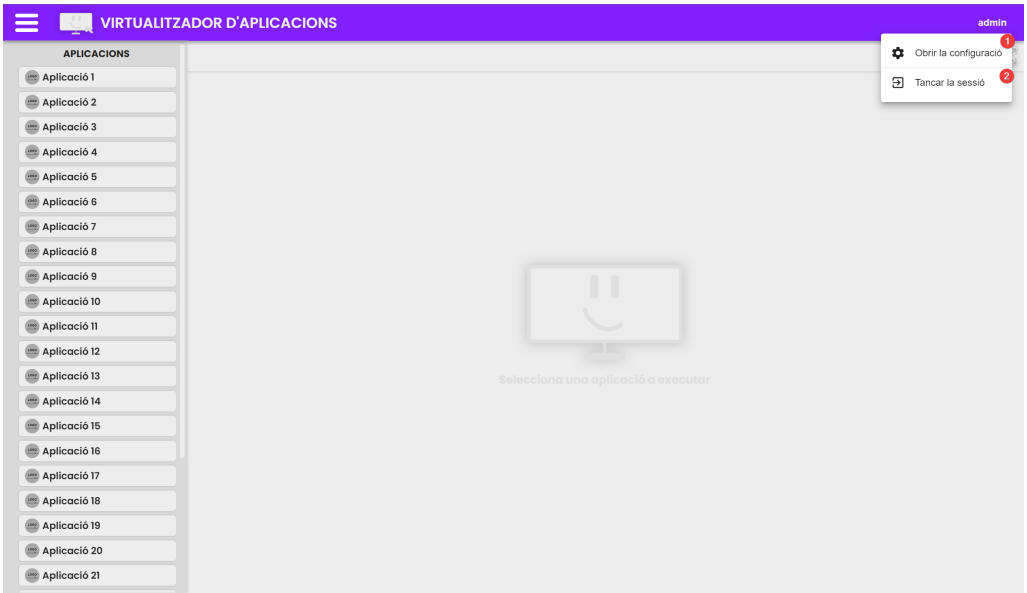

FIGURA 8.10: Plana d'inici, menú contextual

8.5.2 Vista d'identificació i registre

La visió d'identificació i registre presenta a l'usuari formularis per a realitzar aquesta tasca.

8.5.2.1 Identificació

El sistema presenta a l'usuari un formulari per tal d'identificar-se o bé accedir al registre. Vegeu figura 8.11.

1. Camp d'introducció del nom d'usuari
2. Camp d'introducció de la contrasenya
3. Botó d'accés i identificació
4. Accés al formulari de registre

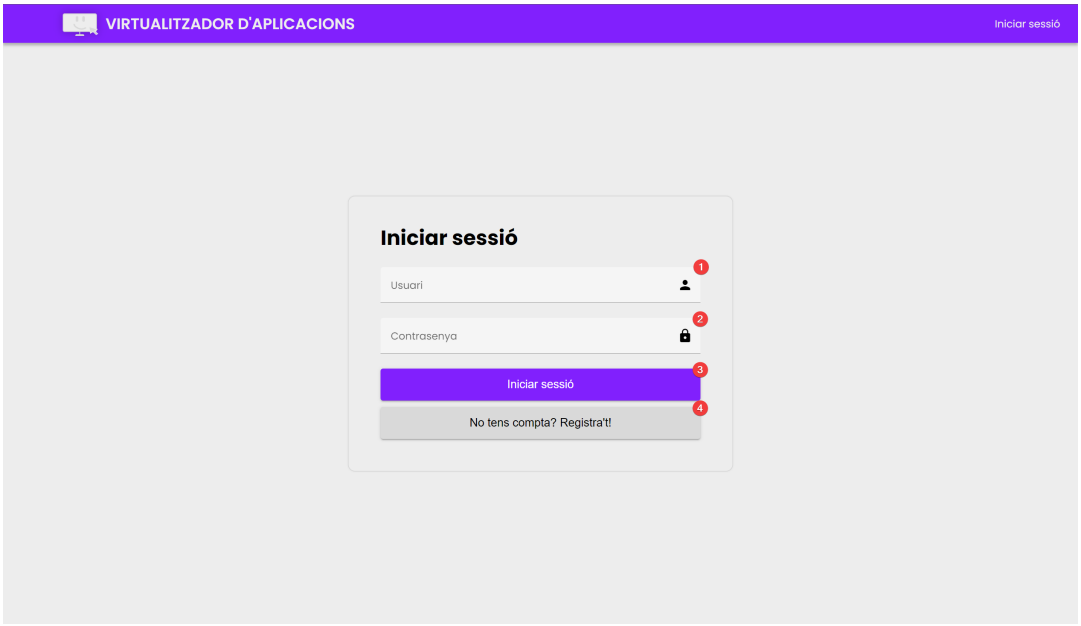

FIGURA 8.11: Identificació d'usuari

8.5.2.2 Registre

El sistema presenta a l'usuari un formulari per tal de registrar-se al sistema. Vegeu figura 8.12.

1. Camp d'introducció del nom d'usuari
2. Camp d'introducció del correu electrònic
3. Camp d'introducció de la contrasenya
4. Camp d'introducció de la validació de contrasenya
5. Botó d'accés i registre

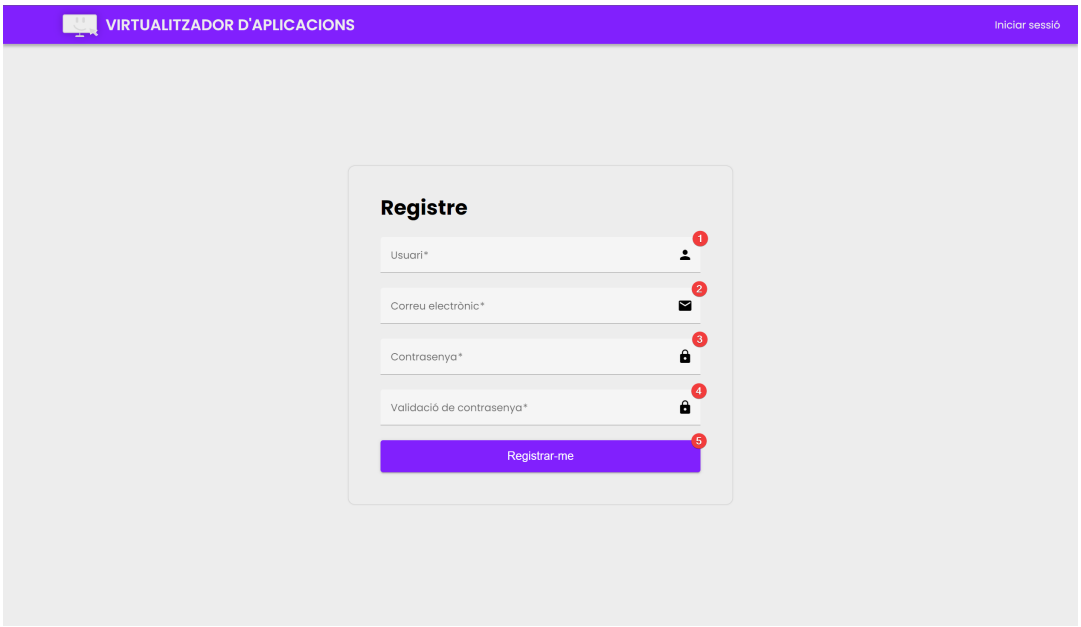

FIGURA 8.12: Registre d'usuari

8.5.3 Vista panell d'administració

El panell d'administració és una plana d'accés restringit, només hi pot accedir l'usuari administrador. En ella es podrà veure l'estat del sistema, aplicacions i usuaris.

8.5.3.1 Usuaris i rols

El sistema presenta a l'usuari administrador la pàgina d'administració d'usuaris i rols. Vegeu figura 8.13.

1. Plana actual: gestió d'usuaris i rols
2. Gestió de rols
3. Visió d'usuaris i assignació de rols

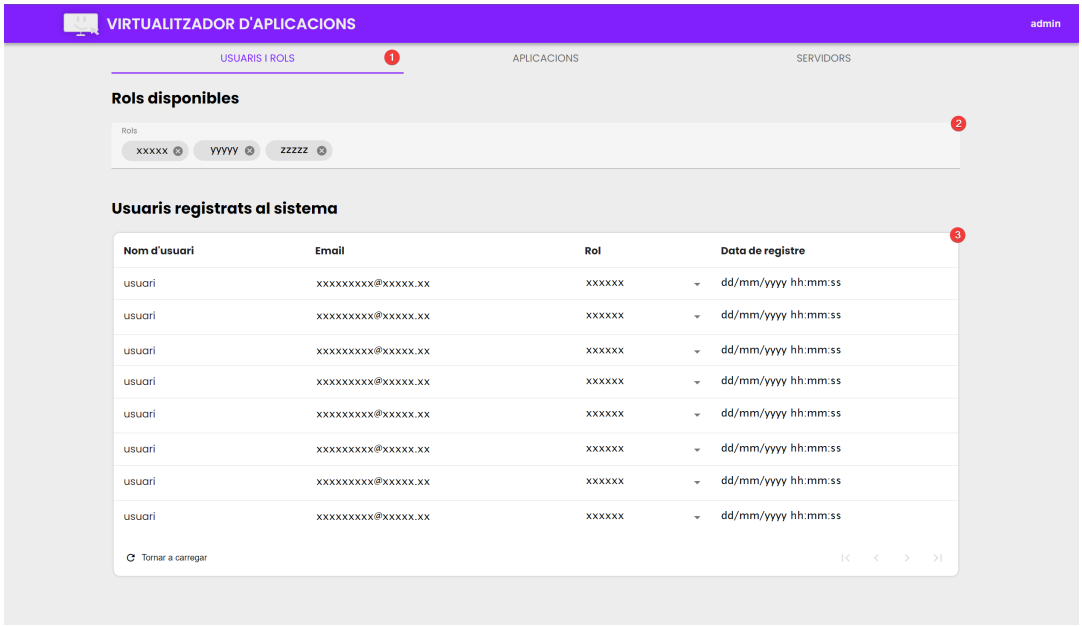

FIGURA 8.13: Gestió d'usuaris i rols

8.5.3.2 Aplicacions

El sistema presenta a l'usuari administrador les aplicacions disponibles en el sistema per tal de configurar-les. Vegeu figura 8.14.

1. Targeta d'aplicació
2. Nom visible de l'aplicació
3. Font del logo de l'aplicació
4. Check aplicació accessible sense autenticació
5. Check aplicació accessible sense cap rol
6. Llistat de selecció de rols que hi tenen accés
7. Desar canvis

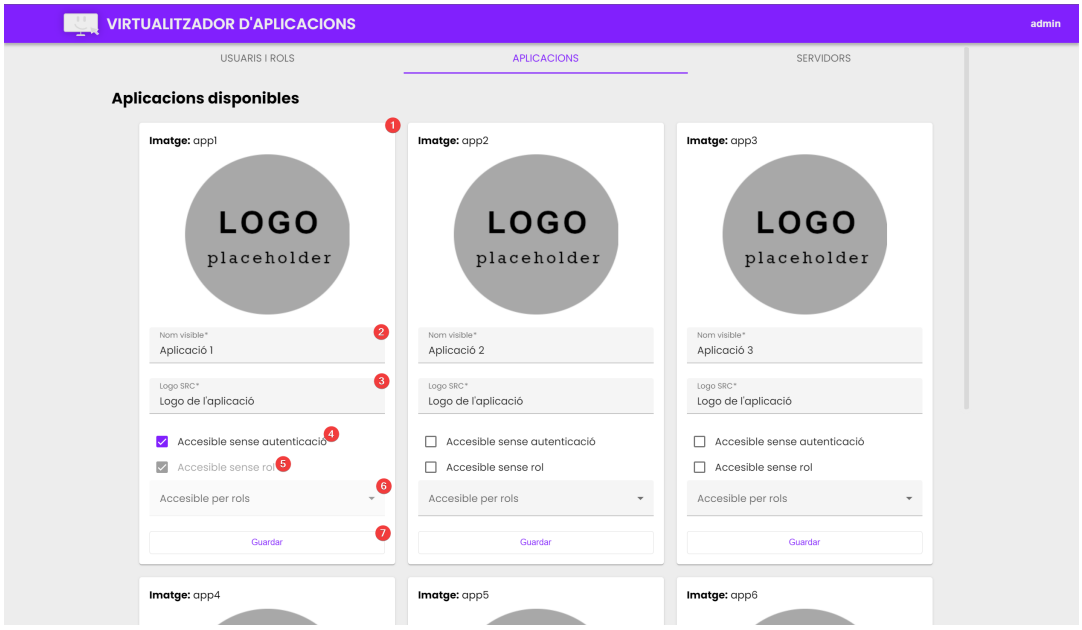

FIGURA 8.14: Gestió d'aplicacions del sistema

8.5.3.3 Servidors

El sistema presenta a l'usuari administrador una vista amb els servidors d'aplicacions actius i el seu estat actual. Vegeu figura 8.15.

1. Recompte total de servidors actius
2. Targeta de servidor
3. Adreça IP del servidor
4. Ús de CPU actual
5. Ús de RAM actual

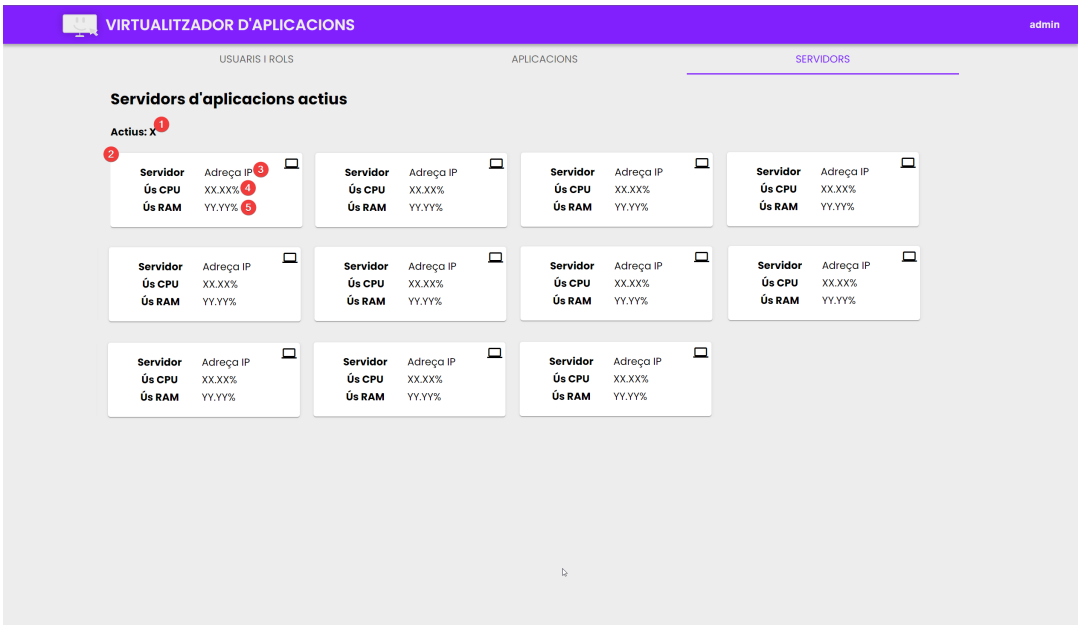

FIGURA 8.15: Visió dels servidors d'aplicacions disponibles

9. Implementació i proves

A continuació es detalla la implementació i proves realitzades per a la validació del funcionament de cadascuna de les parts del sistema.

9.1 Implementació del virtualitzador d'aplicacions

Per tal d'implementar l'entorn del virtualitzador d'aplicacions què, tal com s'ha especificat en la secció de disseny 8.1.1, es tracta d'un contenidor de Docker amb les eines necessàries per a realitzar la connexió i preparat per a instal·lar-hi una aplicació de forma senzilla, s'ha preparat una imatge de Docker i s'ha fet pública en un registre en línia. A continuació es mostren els detalls.

9.1.1 Estructura i implementació de la imatge base

La imatge base segueix una configuració bàsica per tal de complir amb els requisits necessaris. S'usa com a base la imatge del sistema operatiu ubuntu per tal de disposar ja dels elements bàsics del sistema operatiu. Seguidament es mapegen els següents elements en l'estructura del sistema de fitxers de la imatge:

```
/
├── start.sh ← Script d'arrancada dels programes i entorn
├── run.sh ← Script d'arrancada de l'aplicació
├── gateway/ ← Software desenvolupat per al control i retransmissió[1]
├── SHARED/ ← Directori establert per a la compartició de fitxers
└── home/
    └── USER/
        └── .config/
            └── i3/
                └── config ← Fitxer pre-configurat per a l'entorn del gestor
                    de finestres
```

[1] En l'annex del document es pot trobar el codi desenvolupat.

Per tal de generar la imatge, es prepara un *Dockerfile*[2] què, de forma resumida, s'encarrega de:

1. Descarregar la configuració base (partim del sistema d'ubuntu).
2. Còpia els fitxers pre-configurats i de l'aplicació desenvolupada.
3. Descarrega totes les dependències necessàries per al funcionament i compilació.
4. Es compila l'aplicació desenvolupada i s'eliminen les dependències de compilació.[3]
5. S'estableixen els permisos i usuaris bàsics (usuari sense permisos).

Seguidament es mostra el *Dockerfile* que genera la imatge del contenidor base.

```
#Imatge inicial ubuntu
FROM ubuntu

#Entorn no interactiu per instal·lació
ENV DEBIAN_FRONTEND=noninteractive

#Copiem els fitxers de configuració i codi a compilar
COPY config/start.sh /
COPY config/i3-config /home/USER/.config/i3/config
COPY config/gateway /gateway

#Ens posem en l'directori on compilar i executar el gateway
WORKDIR /gateway

#Instal·lació de dependències, compilació i desinstal·lació un cop ja no són necessàries
RUN apt-get update \
    && apt-get install xvfb i3 golang-go ffmpeg gcc libc6-dev libx11-dev xorg-dev libxtst-dev xcb libxcb-xkb-dev x11-xkb-utils libx11-xcb-dev libxkbcommon-x11-dev libxkbcommon-dev ca-certificates -y \
    && go build && rm -r *.* \
    && apt-get purge -y --auto-remove golang-go \
    && useradd -M -u 1000 USER \
    && mkdir /SHARED \
    && chmod +x /start.sh \
    && chown USER:USER /start.sh \
```

[2] Fitxer amb comandes per a la preparació d'una imatge de docker.

[3] Es realitza la compilació dins del *Dockerfile* i no fora ja que aixi permetem implementar el mateix fitxer en múltiples arquitectures i reduïm la mida de la imatge.

```
    && chown -R USER:USER /SHARED \
    && chown -R USER:USER /home/USER/ \
    && chown -R USER:USER /gateway

#Execució com a usuari sense permisos
WORKDIR /SHARED
USER USER

#Script d'arrancada amb els elements necessaris per a formar l'entorn
CMD ["/start.sh"]
```

L'script d'arrancada `start.sh` és el punt d'entrada del contenidor i s'encarrega d'iniciar en paral·lel els processos necessaris per a l'execució de l'aplicació. Com que en la imatge base encara no es disposa d'aplicació instal·lada es crida al fitxer `run.sh` per a tenir el punt d'entrada preparat. Les accions realitzades per l'script del punt d'entrada són:

1. Inici de la pantalla virtual
2. Inici del gestor de finestres
3. Inici del programa desenvolupat per al control i retransmissió de l'aplicació
4. Crida a l'script d'inici de l'aplicació

A continuació es mostra el seu codi:

```
#!/bin/bash
/usr/bin/Xvfb $DISPLAY -screen 0 1920x1080x24+32 -nolisten tcp -nolisten unix &
sleep 3
i3 &
sleep 1
/gateway/main > /gateway/log 2>&1 &
/run.sh
```

9.1.2 Implementació d'imatge amb aplicació

Un cop desenvolupada la imatge base, aquesta s'ha fet pública en el registre oficial de docker, al qual s'hi pot accedir a través del següent enllaç i identificar-la amb el tag `aniolfdz/appvirt:base`: https://hub.docker.com/r/aniolfdz/appvirt/tags
D'aquesta manera, preparar una imatge d'una aplicació concreta és tant senzill com realitzar les següents passes:

1. Especificar la imatge base com `aniolfdz/appvirt:base`.
2. Afegir l'script d'arrancada de l'aplicació.
3. Instal·lar l'aplicació que es vol per al sistema.

9.1.2.1 Exemple d'imatge de contenidor amb l'aplicació GIMP:

```
#Bastat en la imatge base creada pel projecte
FROM aniolfdz/appvirt:base

#Root per instal·lar dependències
USER root
ADD run.sh /run.sh

#Instal·lem l'aplicació a executar
RUN apt-get install gimp -y \
    && chmod +x /run.sh \
    && chown USER:USER /run.sh

USER USER
```

En aquest cas, l'script `run.sh` només contindrà l'ordre d'execució del programa `gimp`. Aquest script es podria complicar amb més instruccions si fos necessari per a l'execució de l'aplicació.

9.1.3 Dificultats trobades i solució

Durant la implementació del virtualitzador d'aplicacions s'han trobat diverses dificultats que han causat petites desviacions en el temps dedicat al desenvolupament de la tasca.

9.1.3.1 Dificultats en el procés de depuració del programari desenvolupat

Al tractar-se d'un programari desenvolupat a mida per l'entorn del contenidor, l'execució i depuració del programari havia de dur-se a terme dins del mateix contenidor

en execució, aquest fet dificultava la depuració del programa ja que en ser un entorn diferent del de desenvolupament no es podia parar i tornar a executar de forma àgil.

9.1.3.1.1 Llarg temps de preparació de la imatge

La imatge base, conté moltes dependències i elements a instal·lar i compilar, generar una imatge base des de zero comportava un temps d'espera de minuts per la preparació de la imatge, retardant així la prova i depuració del programari.

9.1.3.1.2 Solució

Per tal de pal·liar aquesta dificultat, s'ha optat per realitzar les proves d'execució i depuració dins del mateix contenidor utilitzant les eines que proporciona docker. Per tal d'entrar en la instància del contenidor en execució s'ha usat la següent comanda, la qual ens proporciona una *shell* de *bash* dins de la mateixa instància:

```
1 docker exec -it <id_instància> /bin/bash
```

Així doncs, i de forma temporal pel desenvolupament, s'ha generat una imatge base sense el programari compilat, permetent així situar-nos dins del sistema amb les eines de docker, compilar i executar el programari dins de la mateixa instància en execució.

9.1.3.2 Lentitud en la captura dels gràfics

Durant les proves de funcionament, s'ha trobat que, en capturar els gràfics de l'aplicació usant l'eina **ffmpeg** (vegeu 7.4.2.4) l'ús de CPU del contenidor pujava de forma excessiva causant lentitud no només en la retransmissió, sinó que en tot el sistema amfitrió.

9.1.3.2.1 Causa de la lentitud

Després de realitzar diverses proves es va determinar com a la causa de la lentitud i de l'alt ús de CPU al fet d'utilitzar el codec V8 (libvpx) per a codificar la retransmissió de vídeo.

9.1.3.2.2 Solució

La solució al problema implicava fer servir un codec diferent juntament amb altres paràmetres d'optimització per tal de millorar la captura del vídeo. En aquest cas, s'ha escollit usar el codec H264 (libx264) per tal de realitzar la codificació i compressió del vídeo. S'ha escollit H264 degut a la compatibilitat amb la llibreria de retransmissió per WebRTC (Pion) juntament amb la qualitat i bon resultat obtingut.

Per tal de realitzar la captura i retransmissió dels gràfics s'usa la següent comanda:

```
ffmpeg -r 30 -f x11grab -i $DISPLAY -pix_fmt yuv420p -tune
↪ zerolatency -c:v libx264 -quality realtime -f rtp
↪ rtp://127.0.0.1:$PORT
```

On:

- **-r 30**: Específica usar una taxa de 30 fotogrames per segon.
- **-f x11grab**: Indica que la captura de pantalla prové del protocol X11.
- **-i $DISPLAY**: Es captura el display virtual creat.
- **-pix_fmt yuv420p**: Defineix el format de píxels de sortida com a yuv420p.
- **-tune zerolatency**: Optimitza la configuració per a minimitzar la latència.
- **-c:v libx264**: Especifica l'ús del codec H264.
- **-quality realtime**: Defineix el vídeo com a "temps real", prioritzant la velocitat de codificació sobre la qualitat.
- **-f rtp rtp://127.0.0.1:$PORT**: Especifica l'adreça i el port de destí del flux RTP, en aquest cas, el mateix programa desenvolupat per tal de realitzar la conversió RTP→WebRTC.

9.1.4 Proves

Les proves de funcionament del virtualitzador d'aplicacions com a unitat s'han basat en dues etapes:

1. Proves d'execució d'aplicació gràfica sense component remot
2. Proves d'execució remota amb WebRTC

9.1.4.1 Proves d'execució d'aplicació gràfica sense component remot

Per tal de comprovar que el contenidor podia executar realment aplicacions gràfiques sense realitzar el desenvolupament al complet s'ha optat per executar el contenidor sobre una màquina amb sistema operatiu linux i entorn gràfic d'X11. D'aquesta manera i utilitzant les eines de docker se'ns permet executar el contenidor i redirigir els gràfics de l'aplicació cap al sistema amfitrió usant el seu client d'X11.

Comanda utilitzada:

```
docker run -d --rm -v /tmp/.X11-unix:/tmp/.X11-unix:ro -e
↪ DISPLAY=$DISPLAY --name app appvirt
```

L'anterior comanda ens permet executar la imatge creada muntant com a volum virtual el socket creat pel client d'X11 de la màquina amfitrió i especificar com a variable d'entorn "DISPLAY" la mateixa que a la màquina amfitrió. Així doncs, l'execució de l'aplicació succeirà dins el contenidor mentre que els gràfics es veuran renderitzats en el sistema amfitrió.

9.1.4.2 Proves d'execució remota amb WebRTC

A fi i efecte de provar la retransmissió per WebRTC per separat, s'ha utilitzat fora del contenidor la conversió RTP-WebRTC cap a un client web simple (pàgina preparada únicament per a rebre el vídeo). D'aquesta manera, s'ha muntat un escenari on l'aplicació s'executa dins del contenidor però es renderitza, captura i retransmet dins del sistema amfitrió cap a un client web senzill.

9.2 Implementació del servidor d'aplicacions

Tal com s'ha definit en la secció de disseny 8.1.2, el servidor d'aplicacions consta del programari encarregat de proporcionar la connexió entre client i servidor d'aplicacions i alhora orquestrar les instàncies d'aplicacions. A continuació es mostren els detalls més importants de la seva implementació.

9.2.1 Implementació del servei

El servei encarregat del servidor d'aplicacions implementa una API web per a permetre a l'usuari establir la connexió WebSocket amb el servidor per altra banda, accedir al sistema de fitxers compartit entre client i instància d'aplicació.

9.2.1.1 Connexió WebSocket entre client i servidor d'aplicacions

Tal com s'ha especificat en la secció de disseny 8.2 sobre la seqüència de connexió, quan el servidor d'aplicacions rep una petició del client per a iniciar una aplicació, aquest realitza el procediment de validació i posada en marxa de la instància de l'aplicació. Per tal d'implementar aquesta funcionalitat, s'implementa un *socket handler* que mantindrà el cicle de vida de la connexió amb el client. Cada connexió entre client i servidor d'aplicacions viurà en el seu propi procés dins d'aquest *handler*. Per tal de dur a terme la gestió, es realitzen els següents passos;

1. Es realitza l'*upgrade* del protocol HTTP cap a WebSocket aprofitant la connexió TCP ja establerta entre client i servidor.

2. Es comproven els permisos del client per a realitzar l'execució de l'aplicació. Aquesta validació es realitza comprovant la signatura del token rebut i validant que s'ha generat amb la clau privada que només comparteixen els servidors.

3. S'inicia el contenidor de docker que conté l'aplicació desitjada i s'hi munta un volum virtual per tal que posteriorment s'hi puguin compartir fitxers entre client i aplicació. S'envia al client un *token* únic que li permetrà accedir a aquest volum.

4. Un cop posat en marxa el contenidor de docker, s'arranca el bucle d'esdeveniments de teclat i ratolí que es redirigirà cap a l'aplicació.

Per tal d'implementar aquestes funcionalitats, es fa ús de les **goroutines**[4] del llenguatge GO i canals entre aquestes per tal de comunicar-se entre elles.

A continuació és mostra i comenta el cos principal del *socket handler*:

```
1  func SocketHandler(w http.ResponseWriter, r *http.Request) {
2      //*******
3      //  1. Upgrade HTTP a WebSocket
4      //*******
5      socket, err := upgrader.Upgrade(w, r, nil)
6      if err != nil {
7          log.Println("Error upgrade http -> ws", err)
8          return
9      }
10     defer socket.Close() //Tanquem la connexió al sortir
11     log.Println("Client connectat.")
12
13     //*******
14     //  2. REP QUINA APP ENCENDRE I COMPROVAR ELS PERMISOS
15     //*******
16     _, accessToken, err := socket.ReadMessage()
17     if err != nil {
18         log.Println("Error al llegir el token d'accés", err)
19         return
20     }
21     appImage, ok := ProcessToken(string(accessToken))
22     if !ok {
23         log.Println("Error de procés amb el token. Finalitzant
   ↪  connexió.")
24         return
25     }
26
```

[4]Una goroutine és un fil d'execució separat que executa una part del codi de forma concurrent

```
27      //*******
28      //  3. S'encén el docker. Esperar a rebre un señal de que
    ↪   ja està actiu i estableix la connexió
29      //*******
30      // Inici de socket local per la comunicació amb docker
31      WSinTCPout := make(chan string) //Canal de comunicació
    ↪   entre socket ws i tcp
32      TCPinWSout := make(chan string) //Canal de comunicació
    ↪   entre socket tcp i ws
33      proxyStopSignal := make(chan struct{}) //Canal d'ordre de
    ↪   finalització del socket tcp
34      defer close(proxyStopSignal)
35      go DockerProxy(WSinTCPout, TCPinWSout, proxyStopSignal)
36
37      var portSocket string = <- TCPinWSout //Espera a l'inici
    ↪   del socket per obtenir el nº de port
38
39      //Inici de la imatge de docker
40      dockerStopSignal := make(chan struct{}) //Canal d'ordre de
    ↪   finalització de docker
41      token, fullPath := InitializeDirectory()
42      socket.WriteMessage(websocket.TextMessage,
    ↪   []byte(base64.StdEncoding.EncodeToString([]byte(
    ↪   fmt.Sprintf("{\"token\":\"%s\"}", token)))))
43      defer close(dockerStopSignal)
44      defer DeleteDirectory(fullPath)
45      go StartDockerImage(appImage, portSocket, fullPath,
    ↪   dockerStopSignal)
46
47      //*******
48      //  4. Inici del proxy WS <-> TCP
49      //*******
50      go func(){ //Llegir tot el que arriba del WS i reenviar-ho
    ↪   cap al docker
51          for {
52              _, message, err := socket.ReadMessage()
53              if err != nil {
54                  break
55              }
56              WSinTCPout <- string(message)
57          }
58      }()
59      for {//Llegir tot el que arriba de TCP i reenviar-ho cap
    ↪   al client
60          var tcpMsg string = <- TCPinWSout
61          if tcpMsg == ""{ //Parem si es rep stop
```

```
62              log.Println("Stop signal rebut.")
63              break
64          }
65          err = socket.WriteMessage(websocket.TextMessage,
    ↪  []byte(tcpMsg))
66          if err != nil { //Parem si hi ha error
67              log.Println("Error a l'enviar TCP -> WS", err)
68              break
69          }
70      }
71
72      //Fi.
73      log.Println("Finalitzat Proxy WS <-> TCP.")
74  }
```

9.2.1.1.1 *Upgrade* d'HTTP a WebSocket

En el moment que es rep una petició HTTP al punt d'entrada de WebSocket, es negocia amb el client la millora del protocol cap a WebSocket reaprofitant la mateixa connexió TCP ja establerta [25]. En aquest punt cal especificar en l'*upgrader* que cal acceptar connexions des de qualsevol origen. Recordem que el client web ha estat servit des del servidor central i que per tant, serà diferent del servidor d'aplicacions. Veure dificultat 9.2.2.1.

9.2.1.1.2 Recepció de *token* i ordre d'execució

Un cop establerta la connexió amb WebSocket, el primer missatge rebut per part del client serà el *token* d'accés proporcionat pel servidor de control. Aquest *token* conté la informació necessària per a identificar quina aplicació executar i validar si el servidor d'aplicacions al que s'està establint la connexió és l'indicat pel servidor de control juntament amb si el *token* proporcionat és vàlid.

9.2.1.1.3 Arranc del contenidor de Docker

Per a dur a terme l'arranc del contenidor de docker, s'inicien dos nous fils d'execució.

1. El primer fil, serà l'encarregat d'obrir un socket TCP a l'espera que la instància del contenidor s'hi connecti. Aquest socket s'obrirà en un port local aleatori dels disponibles pel sistema (*localhost:0*)[5]. Més tard serà reaprofitat com a socket intermediari entre Client i aplicació. El canal complet serà:
   Client ↔ WebSocket ↔ Servidor d'aplicacions ↔ TCP ↔ Instància d'aplicació.

[5]Múltiples clients es poden connectar simultàniament i els hem de poder servir a tots. L'especificació localhost:0 indica al S.O obrir el socket en algun port disponible per l'adreça de *loopback*.

2. El segon fil, serà l'encarregat d'iniciar la imatge del contenidor de docker amb l'aplicació seleccionada. En iniciar la imatge, es passarà a través de variables d'entorn el port obert pel primer fil de manera que el contenidor pugui establir connexió amb el socket TCP.

9.2.1.1.4 *Proxy* WebSocket ↔ TCP

Un cop establerta la connexió amb el contenidor, s'indicarà al client que s'ha establert correctament la connexió. Des d'aquest instant, el fil principal encarregat de la gestió del WebSocket, es dedicarà completament en llegir els missatges enviats pel client a través de WebSocket i reenviar-los cap a la instància de la imatge a través del socket TCP. Simultàniament, un nou fil es dedicarà a llegir els missatges TCP que pugui enviar la instància de l'aplicació i es reenviaran utilitzant el WebSocket cap al client.

9.2.1.1.5 Seqüència temporal dels fils en execució

Per tal de visualitzar de forma senzilla els fils oberts i la seva comunicació durant el temps de vida de la connexió amb el client, es presenta el següent diagrama de seqüència:

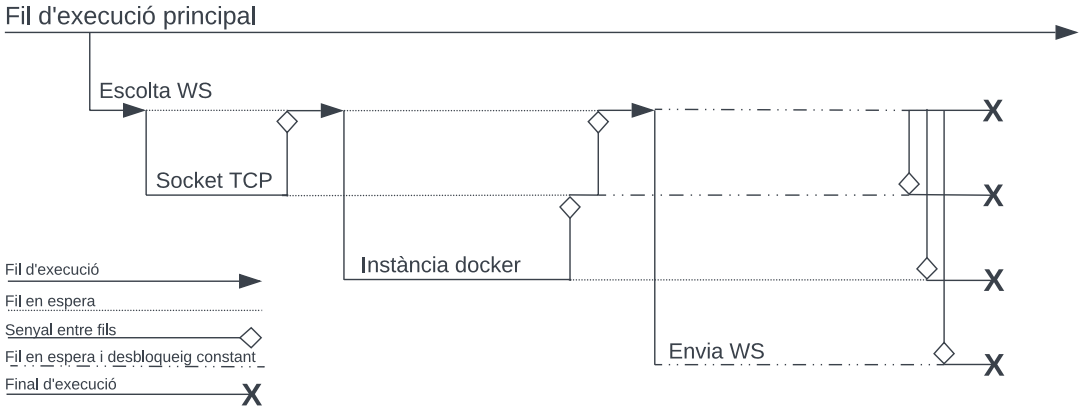

FIGURA 9.1: Seqüència temporal dels fils d'execució

Notar com el temps en "espera i desbloqueig constant" és en realitat un conjunt de senyals indefinit entre fils de sockets per tal de reenviar els missatges entre ells. L'execució finalitza quan el client envia missatge de finalització.

9.2.1.2 Accés al sistema de fitxers de l'aplicació

Per tal de permetre l'accés al sistema de fitxers de l'aplicació, durant l'arrancada dels contenidors (vegeu 9.2.1.1.3) també s'assigna a la sessió de l'usuari un *token* únic per a aquella sessió per tal de permetre l'accés al sistema de fitxers. Cada instància de contenidor en execució, disposa d'un volum virtual propi el qual té accés compartit amb la màquina amfitrió del servei. Així doncs, l'usuari pot interactuar directament i sense passos intermedis amb el servidor d'aplicacions perquè aquest obtingui o afegeixi fitxers dins del volum virtual. Per a fer-ho, el servidor d'aplicacions posa a disposició de l'usuari 3 *endpoints* HTTP perquè l'usuari realitzi les accions de llistat, addició i eliminació de fitxers i directoris dins del seu volum assignat per la sessió. La figura 9.2 representa un diagrama de la connexió entre usuari i servidor d'aplicacions tot muntant el volum virtual perquè l'aplicació hi tingui accés.

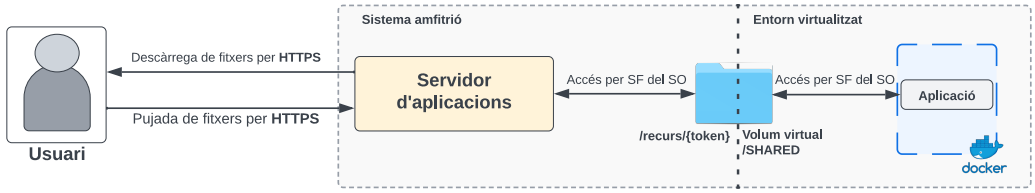

FIGURA 9.2: Diagrama de connexió per l'accés al sistema de fitxers.

A continuació es detallen els *endpoints* implementats per a l'accés al volum compartit. Notar que en tots els casos, el paràmetre d'accés *path* estarà limitat a l'arrel del volum virtual de la sessió de l'usuari.

- Obtenció de fitxers i directoris d'un directori donat.

Descripció	Obtenció de fitxers i directoris d'un directori donat.
Recurs	/list
Paràmetres	token → string amb el token assignat path → string amb el path del directori a explorar.
Mètode	GET
Requisits	- El token és vàlid i existeix un volum amb el token.

Resposta	{
	"fullpath": string
	"parent": string
	"files": [{
	"name": string
	"isFile": bool
	}]
	}

Resposta alternativa	Status code: 400 - Bad Request

TAULA 9.1: Recurs: Llistat de fitxers i directoris del volum compartit

- Descàrrega d'un fitxer.

Descripció	Descàrrega d'un fitxer.
Recurs	/download
Paràmetres	token → string amb el token assignat path → string amb el path del directori on hi ha el fitxer. file → string amb el nom del fitxer.
Mètode	GET
Requisits	- El token és vàlid i existeix un volum amb el token.
Resposta	Descàrrega dinàmica del contingut segons el tipus de fitxer. Es respon amb el header de resposta "Content-Disposition": "attachment; filename={fitxer}".
Resposta alternativa	Status code: 400 - Bad Request

TAULA 9.2: Recurs: Descàrrega de fitxer del volum compartit

- Pujada d'un fitxer.

Descripció	Pujada d'un fitxer.
Recurs	/upload
Paràmetres	token → string amb el token assignat path → string amb el path del directori on es vol pujar el fitxer.
Mètode	POST
Cos	El fitxer es puja com a camp de formulari de tipus fitxer.
Requisits	- El token és vàlid i existeix un volum amb el token. - La mida del fitxer és inferior a 10MB.
Resposta	Status code: 200 - OK
Resposta alternativa	Status code: 400 - Bad Request

TAULA 9.3: Recurs: Descàrrega de fitxer del volum compartit

9.2.1.3 Arrancada de contenidors i *mapeig* de recursos.

Tal com s'ha comentat en la secció 9.2.1.1.3, l'arrancada del contenidor es veu sotmesa a un seguit de processos i accions conjuntes. Per tal d'arrancar el contenidor, s'utilitza l'SDK propi de Docker, desenvolupat en el llenguatge GO [26]. Els punts que cal tenir en compte durant l'arrancada són els següents:

1. **Muntatge del volum virtual compartit**
   Durant l'arrancada d'un contenidor amb una aplicació, cal crear un directori en el sistema de fitxers amfitrió que servirà com a volum virtual per al contenidor. Aquest volum serà l'usat com a element compartit entre sistema de fitxers real i de l'aplicació, per a així poder-hi accedir des de fora i permetre a l'usuari la seva interacció. El directori generat en el sistema de fitxers amfitrió es genera de forma automàtica i amb nom aleatori dins de la ruta especificada per la

configuració del servei (per defecte a /tmp/appvirt). El volum muntat dins del contenidor tindrà sempre la ruta `/SHARED`.

2. **Mode de xarxa del contenidor**
   L'arrancada del contenidor es realitza en mode de xarxa compartit amb l'amfitrió, és a dir, la instància virtual i la màquina real compartiran la mateixa adreça IP. Es realitza d'aquesta manera per tal de simplificar el sistema i permetre una connexió WebRTC directe entre la instància de l'aplicació i l'usuari sense haver de passar per un servidor de *relay*.

3. **Muntatge del DRI**
   El DRI (*Direct Rendering Interface*) és una interfície usada pel sistema de finestres d'X perquè les aplicacions gràfiques puguin accedir directament al hardware de renderitzat. La interfície permet l'acceleració per hardware a través d'implementacions d'OpenGL. Per defecte, docker no utilitza el DRI de la màquina amfitriona i aquest fet degrada l'experiència d'usuari. Degut a això, s'ha optat per muntar el DRI de la màquina amfitriona com a DRI del contenidor (aquesta opció està habilitada per defecte però és configurable en tot moment a través de la configuració del servei durant la instal·lació, vegeu el manual d'usuari A.2.3.2).

9.2.2 Dificultats trobades i solució

9.2.2.1 Accés des de diferents orígens

9.2.2.1.1 Denegació de connexió per WebSocket

Els servidors web, disposen d'una mesura de seguretat per tal d'evitar que es realitzin peticions fraudulentes contra el servei des de d'altres planes web. La mesura pretén protegir a l'usuari per si aquest accidentalment entra a una pàgina maliciosa que realitza peticions contra un servei al que es troba autenticat. Aquesta mesura de seguretat se la coneix com a CORS (*Cross-Origin Resource Sharing*) [27].

Durant el disseny i implementació, ja es preveia alleugerar aquesta mesura de seguretat per tal d'accedir als servidors d'aplicacions ja que la pàgina web és servida per un servidor diferent. Així doncs, es va establir una política d'accés permissiva que permetés l'accés des de qualsevol origen.

El problema però, succeïa a l'hora d'establir la connexió per WebSocket. Tot i poder realitzar crides HTTP estàndard des del client cap al servidor d'aplicacions només amb la política de CORS, aquesta no era suficient per establir la connexió amb WebSocket.

9.2.2.1.2 Solució

El problema venia donat en la mateixa implementació de la llibreria utilitzada per l'ús de WebSockets [28]. A banda d'especificar la política de CORS, cal definir a part, un mètode per a la gestió de l'*upgrade* del protocol HTTP cap a WebSocket que defineixi que fer si es tracta d'una connexió d'un origen diferent del del servidor en què s'executa.

```
1 // Permetem accés des de qualsevol origen al fer l'upgrade
  ↪ http -> ws
2 var upgrader = websocket.Upgrader{
3     CheckOrigin: func(r *http.Request) bool {
4         return true
5     },
6 }
```

9.2.2.2 HTTPS i certificats autosignats

9.2.2.2.1 Fallada en la connexió per WebSocket

Com és habitual, el navegador web avisa a l'usuari quan s'està intentant accedir a una plana web que disposa d'un certificat autosignat[6]. Durant el procés de desenvolupament d'un projecte, és usual utilitzar aquest tipus de certificats per a provar la connexió tal com serà en producció. En fer-lo servir i accedir a la plana web, el navegador avisarà a l'usuari que la pàgina no és segura i li habilitarà un botó per permetre'n l'accés.

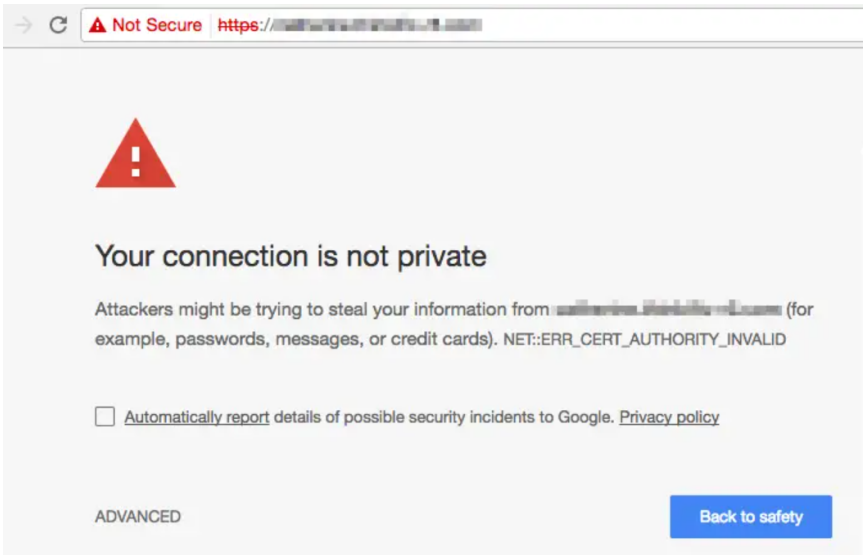

FIGURA 9.3: Avís del navegador al accedir a una plana amb certificat autosignat

[6]Certificiat TLS signat manualment i no validat per cap CA

En acceptar l'avís de seguretat, el navegador concedeix accés a la plana "insegura". En el cas del projecte però, el problema venia pel fet que l'usuari accedeix a una plana "insegura" (servida pel servidor web i de control) i intenta realitzar una connexió segura contra un dels servidors d'aplicacions a través de websocket. Com que aquesta connexió no la realitza directament l'usuari (es realitza mitjançant codi javascript) el navegador web no dona l'opció a l'usuari d'acceptar la inseguretat de la connexió i establir-la amb normalitat.

9.2.2.2.2 Solució

Per tal de permetre la connexió WebSocket i evitar la fallada de connexió pel certificat autosignat, s'ha optat per instal·lar el certificat del servidor en la màquina local, fent així que el navegador el reconegui com a connexió segura.

9.2.3 Proves

Les proves de funcionament sobre programes concurrents són difícils de realitzar, per aquest motiu s'ha optat per provar cadascun dels processos involucrats de forma separada i verificar el seu correcte funcionament per després integrar-los junts.

9.2.3.1 Execució de contenidors de docker

De forma local, s'ha forçat al procés en obrir una imatge específica de docker i parar-la passat un cert temps.

9.2.3.2 Connexió TCP amb el contenidor

Validat ja el funcionament i arrancada del contenidor, s'executa el fil encarregat de rebre la connexió de docker. Es comprova que en iniciar el contenidor es rep la connexió i s'hi poden transmetre missatges.

9.2.3.3 Connexió WebSocket amb el client

De forma aïllada, es realitza una connexió WebSocket amb una pàgina simple, la qual envia un missatge al servidor i aquest respon amb el mateix missatge.

9.3 Implementació de l'aplicació web

Tal com s'ha especificat en l'elecció de tecnologies 7.4.6, la implementació de l'aplicació web s'ha realitzat usant el *framework* Angular. Aquest segueix un patró de disseny orientat a components, així doncs cadascun dels elements desenvolupats per a la interfície d'usuari forma part d'un component.

9.3.1 Components de l'aplicació web

A continuació es detallen els components desenvolupats per a l'aplicació web.

- **AdminPage**: Pàgina global per l'administrador, conté el layout dels elements.
- **AppAdmin**: Component que conté el llistat d'aplicacions disponibles per a la vista d'administrador.
- **AppAdminCard**: Layout que específica de la targeta d'una aplicació per a l'administrador.
- **Application**: Layout que específica de la targeta d'una aplicació per a usuaris.
- **Aside**: Vista del llistat d'aplicacions per l'usuari.
- **ContentContainer**: Component encarregat de contenir l'aplicació en execució i retransmetre els seus controls.
- **ContentNavigation**: Barra de navegació de les aplicacions obertes i accés als botons pel sistema de fitxers i pantalla completa.
- **fsbrowser**: Modal per a la navegació en el sistema de fitxers.
- **header**: Header de l'aplicació amb el títol i botó d'identificació.
- **login**: Component per a la identificació de l'usuari.
- **PageContainer**: Component que engloba tota la vista de l'usuari.
- **ServerAdmin**: Component amb el layout de la vista de servidors per a l'administrador.
- **ServerCard**: Layout que específica de la targeta d'un servidor per a l'administrador.
- **Signup**: Component per al registre de l'usuari.
- **UserAndRoles**: Pàgina amb el layout per la gestió d'usuaris i rols.
- **UserList**: Component amb el llistat d'usuaris registrats al servei.

9.3.2 Serveis de l'aplicació web

A continuació es detallen els serveis desenvolupats per a l'aplicació web.

- **ApiService**: Servei injectable que ens permet comunicar-nos amb el servidor de forma senzilla i asíncrona.

- **AppService**: Servei de control de les aplicacions en execució.
- **RoleService**: Servei de control de rols.
- **SnackbarService**: Servei que proporciona control sobre missatges a mostrar en l'aplicació.
- **ServersService**: Servei per a la comprovació del temps de resposta amb els servidors d'aplicacions.

A continuació i referent al servei de **ServersService**, es mostra un tros de codi on es realitza la part d'inicialització de la seqüència de connexió definida en la secció 8.2.1.

```
//Obtenció dels servidors disponibles per provar latències.
  public getServersWithLatency(callback: any) {
    this.http.get('/server/getIps').subscribe(
      (servers: any) => {
        servers.map((server: string) =>
        ↪  this.testServerLatency(server));
        setTimeout(
            ()=> callback(this.availableServersWithLatency),
            this.TIMEOUT
        );
      },
      () => callback([])
    );
  }

  //Test de latències per cada servidor
  private testServerLatency(server: string) {
    const initialMS = new Date().getTime();
    fetch(`http${State.SECURE ?
    ↪  "s":""}://${server}:${State.APPSERVER_PORT}/ping`,
    ↪  {method:"HEAD"})
      .then(() => {this.availableServersWithLatency.push({
        server: server,
        msToResponse: new Date().getTime() - initialMS
      })})
      .catch(error => {});//No hi ha connexió amb el servidor
  }
```

9.3.3 Desplegament de l'aplicació web cap al servei

Per realitzar el desenvolupament de l'aplicació web, s'ha realitzat un desenvolupament àgil amb un servidor de desenvolupament, ara bé, per tal de portar l'aplicació web en el seu entorn productiu (dins del servidor de control, que serà qui servirà

l'aplicació) s'ha hagut de compilar les dependències i mòduls i dur el resultat com a recurs del servidor de control.

Un cop desenvolupada l'aplicació web. Es compila utilitzant la línia d'ordres `ng build`. Aquesta s'ha configurat per tal de que generi els fitxers de sortida directament al directori generat pel servidor de control i pugui ser directament integrat com a fitxers de recurs.

9.4 Implementació del servidor de control i la persistència de dades

El desenvolupament del servidor de control ha seguit el disseny de l'API especificada en la secció de disseny 8.4. Per tal de dur a terme el servei, s'ha seguit amb el patró MVC (Model Vista Controlador) típic d'una API. A continuació es comenten els aspectes més rellevants:

9.4.1 Algoritme de selecció de servidor d'aplicacions

Tal com s'ha vist en l'estudi previ 5.2, el temps de resposta entre usuari i servidor d'aplicacions serà un element clau per valorar la qualitat de la connexió. Així doncs, tal com s'havia previst en la secció de disseny 8.2.1, es delega al servidor de control la selecció del servidor òptim per a realitzar la connexió. Per a fer-ho, es té en compte el temps de resposta i l'ús actual dels servidors.

Un cop el client sol·licita l'execució d'una aplicació, el servidor de control rep de l'usuari el llistat amb els temps de resposta de cada servidor, aleshores aplica la següent elecció:

1. Del llistat rebut, es filtren i deixen com a candidats aquells amb un temps de resposta mínim. Des del temps de resposta més baix rebut fins a completar la desviació per blocs, on aquesta desviació és múltiple de 100ms. Així doncs, si el temps de resposta mínim és 140ms, només seran candidats aquells servidors amb temps de resposta inferior a 200ms.

2. Dels servidors candidats, ens quedem amb el que disposa d'una ocupació en RAM més petita.

A continuació es mostra el codi implementat per a l'elecció del servidor.

```
1  chooseServer: function (llistatMs) {
2          //Busquem llindar inferior
3          const minLat = llistatMs.reduce((min, x) =>
           ↪  x.msToResponse < min ? x.msToResponse : min,
           ↪  Number.MAX_SAFE_INTEGER);
```

```
4
5           //Obtenim llindar màxim de temps de resposta
            ↪ (multiples de MS_ACCEPTABLES) i com a mínim
            ↪ MS_ACCEPTABLES
6           const llindar = Math.max(MS_ACCEPTABLES,
            ↪ Math.ceil(minLat / MS_ACCEPTABLES) *
            ↪ MS_ACCEPTABLES);
7
8           //Filtrem aquells servidors que tenen un temps de
            ↪ resposta inferior al llindar
9           const acceptables = llistatMs.filter(x =>
            ↪ x.msToResponse < llindar).map(x => x.server);
10
11          //Obtenim servidor amb la major quantitat de ram
            ↪ disponible
12          let servidorEscollit = null;
13          acceptables.forEach((ip) => {
14              if (servers[ip] && (servidorEscollit == null ||
                ↪ servers[ip].ram >
                ↪ servers[servidorEscollit].ram))
15                  servidorEscollit = ip;
16          })
17
18          return servidorEscollit;
19      }
```

El criteri seguit per a l'elecció del servidor busca minimitzar el temps de resposta entre usuari i servidor d'aplicacions tot assignant el servidor amb més disponibilitat. Sempre es prioritza que l'usuari rebi servei sigui quin sigui l'estat del sistema i temps de resposta. En cap cas es descarta una sol·licitud si hi ha algun servidor actiu.

9.4.2 Implementació de l'API

La implementació de l'API de control ha seguit el procés iteratiu marcat per la metodologia de treball sense cap més alteració i tot seguint amb el disseny especificat per cadascun dels *endpoints* a implementar. A continuació es mostra el punt d'entrada del servidor amb la càrrega de totes les rutes disponibles.

```
1 require('dotenv').config();
2 const express = require('express');
3 const jwt = require('./security/jwt')
4 const cors = require('cors');
5 const bodyParser = require('body-parser')
6 const usersRouter = require('./controller/user');
7 const roleRouter = require('./controller/role');
```

```
const appRouter = require('./controller/app');
const appserverRouter = require('./controller/server');
const fs = require('fs');
const https = require('https');
const port = process.env.PORT || 3000;
const SECURE = !process.env.HTTP;

const app = express()//Configuració del servidor
  //Middlewares:
  .use(jwt.processToken)//Autenticació per token
  .use(cors({ origin: '*' }))//Habilita qualsevol origen
  .use(bodyParser.json());//Parse body a JSON

app.use(express.static('web-client'));//Contingut estàtic

//ROUTER
app.use('/user', usersRouter);
app.use('/role', roleRouter);
app.use('/app', appRouter);
app.use('/server', appserverRouter);

//Arranca el servidor web
if (SECURE) {
  https.createServer({
    key: fs.readFileSync('server.key'),
    cert: fs.readFileSync('server.cert'),
  }, app).listen(port, "0.0.0.0", () => console.log(`Iniciat
  ↪ https://0.0.0.0:${port}`));
}
else
  app.listen(port, "0.0.0.0", () => console.log(`Iniciat
  ↪ http://0.0.0.0:${port}`));
```

9.4.3 *Dockerització* del servei

L'execució del servidor de control serà també sobre una imatge de docker. Per fer-ho, es crea un *Dockerfile* on s'hi afegeix el codi necessari per a executar el servidor.

Dockerfile del servidor central

```
#Imatge original de node
FROM node:18

#Establim el directori de l'aplicació
WORKDIR /app

```

```
# Copia les dades de l'aplicació
COPY ./server .

# Instal·lar les dependències
RUN npm install

# Port de l'aplicació
EXPOSE 3000

# Inicia l'aplicació
CMD ["npm", "start"]
```

La imatge del servei s'ha fet disponible públicament al registre d'imatges de docker amb el tag `aniolfdz/appvirt:web`
https://hub.docker.com/r/aniolfdz/appvirt

9.4.4 Implementació de la persistència de dades

Per tal d'implementar la base de dades i repositori d'imatges s'usarà una implementació a partir de contenidors de docker, tal com s'ha especificat anteriorment en la secció de disseny. En aquest cas i pel desenvolupament, es realitzarà la implementació del servidor central, base de dades i registre d'imatges en una mateixa màquina, per a fer-ho, es fa ús de l'eina *Docker Compose*. Aquesta eina ens permet la configuració i execució simultània de diversos contenidors de docker utilitzant una sola comanda. Aprofitarem per a afegir en aquest mateix fitxer l'execució del servei del servidor web.

9.4.4.1 Fitxer *Dockercompose* amb els serveis

El següent fitxer *Dockercompose* especifica l'arrancada de tres instàncies de docker, mentre que alhora, munta dos volums de persistència de dades per a emmagatzemar permanentment els canvis. A continuació es comenta cada contenidor:

1. La primera instància es tracta del servidor de BDD MySQL on s'hi assigna a través de variables d'entorn el nom de la base de dades i les claus d'accés.

2. La segona instància es tracta del servidor de Docker Registry per a dur a terme la persistència de les imatges de docker de les aplicacions.

3. Finalment, l'última imatge es tracta del servei desenvolupat amb l'API de control i el servidor web amb l'aplicació desenvolupada. Notar com a través de variables d'entorn s'especifica el host de la base de dades, el host del registre i les claus d'accés. A més, munta dins la imatge els certificats necessaris per a realitzar una connexió segura.

La implementació s'ha realitzat d'aquesta manera perquè el servei pot ser completament distribuït, i així ens permetrà, si ho volguéssim, separar cadascuna de les instàncies en màquines diferents.

A continuació es mostra el fitxer preparat amb els serveis i els seus volums:

```yaml
version: '3'
services:
  #Servidor de BDD local.
  mysql:
    image: mysql:latest
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: root
      MYSQL_USER: admin
      MYSQL_PASSWORD: admin
      MYSQL_DATABASE: appvirt
    network_mode: "host"
    volumes:
      - db:/var/lib/mysql
    command: --bind-address=0.0.0.0
    --default-authentication-plugin=mysql_native_password

  #Docker Registry
  registry:
    image: registry:2
    restart: always
    network_mode: "host"
    volumes:
      - registry-data:/var/lib/registry

  #WebServer
  webserver:
    image: aniolfdz/appvirt:web
    restart: always
    environment:
      DB_HOST: 127.0.0.1
      DB_USER: root
      DB_PASSWORD: root
      DB_DATABASE: appvirt
      ADMIN_PW: admin
      REGISTRY_HOST: localhost:5000
    network_mode: "host"
    depends_on:
      - mysql
    volumes:
      - ./server.cert:/app/server.cert
```

```
      - ./server.key:/app/server.key

volumes:
  db:
  registry-data:
```

9.4.5 Proves

Per tal de provar el funcionament de l'API, s'ha utilitzat l'aplicació Postman per tal d'anar reproduint les crides i comprovant el seu correcte funcionament. La validació a seguir ha estat la comprovació que els elements generats concorden amb els planejats durant el disseny.

9.5 Proves del sistema

Per tal de validar el correcte funcionament del sistema i la integració entre tots els components, s'ha dissenyat un control de qualitat a seguir per tal de comprovar els components de l'aplicació. En concret, les proves realitzades són:

1. Prova de registre i identificació d'un usuari
2. Prova d'execució d'una aplicació
    - Prova de control de l'aplicació
    - Prova de control en pantalla completa
    - Prova d'accés al sistema de fitxers
3. Prova d'execució de múltiples aplicacions simultànies
4. Prova de configuració del sistema (administrador)
    - Configurar rols d'usuaris
    - Configuració d'aplicacions i permisos
    - Accés com a usuari per a comprovar els canvis

Les proves descrites s'han provat i verificat en els escenaris detallats en les següents seccions. Els escenaris han estat preparats a partir d'entorns virtualitzats amb màquines virtuals i cadascun d'ells busca verificar que el sistema funciona correctament amb diferents entorns de xarxa. El **per què** de les proves en diferents escenaris de xarxa ve donat pel fet d'haver d'establir diferents tipus de connexió (HTTP, WS i WebRTC) i cal provar que en tots els entorns segueix funcionant. Totes les proves s'han superat amb èxit, a continuació es detallen els escenaris testejats.

9.5.1 Escenari de desenvolupament

L'escenari de desenvolupament fa referència a l'execució de tots els serveis sobre el mateix sistema i l'aplicació d'usuari accedint-hi de forma local. És a dir, tots els elements sobre la mateixa màquina i sistema. El funcionament d'aquest escenari està garantit ja que és el mateix sobre el que s'ha desenvolupat.

Diagrama de l'escenari de desenvolupament:

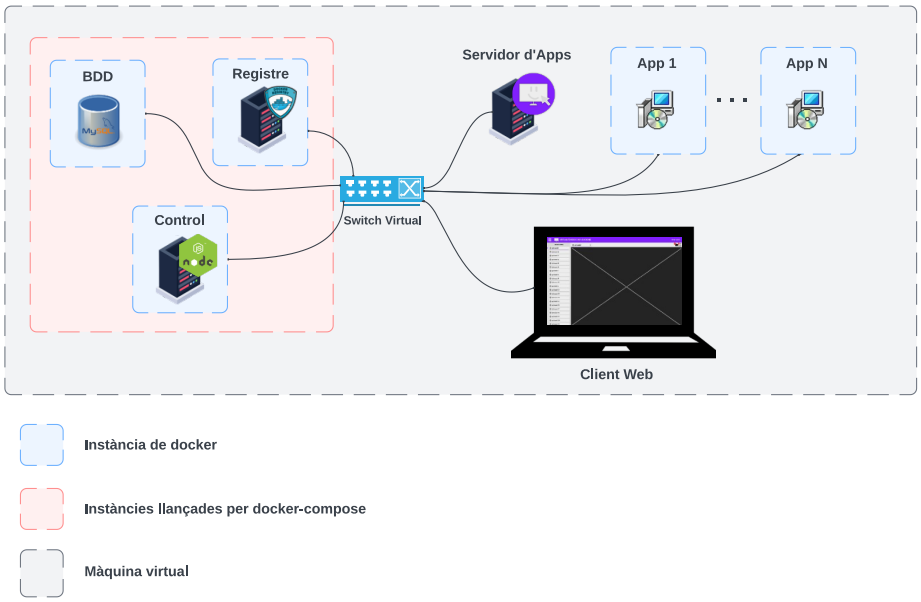

FIGURA 9.4: Escenari de desenvolupament

9.5.2 Escenari xarxa local (monolític)

L'escenari en xarxa local pretén posar a prova el funcionament del sistema desenvolupat usant el mateix entorn que en desenvolupament però aquest cop accedint a la plana des d'un equip físic diferent i connectat a la xarxa local.

Diagrama de l'escenari en xarxa local (monolític):

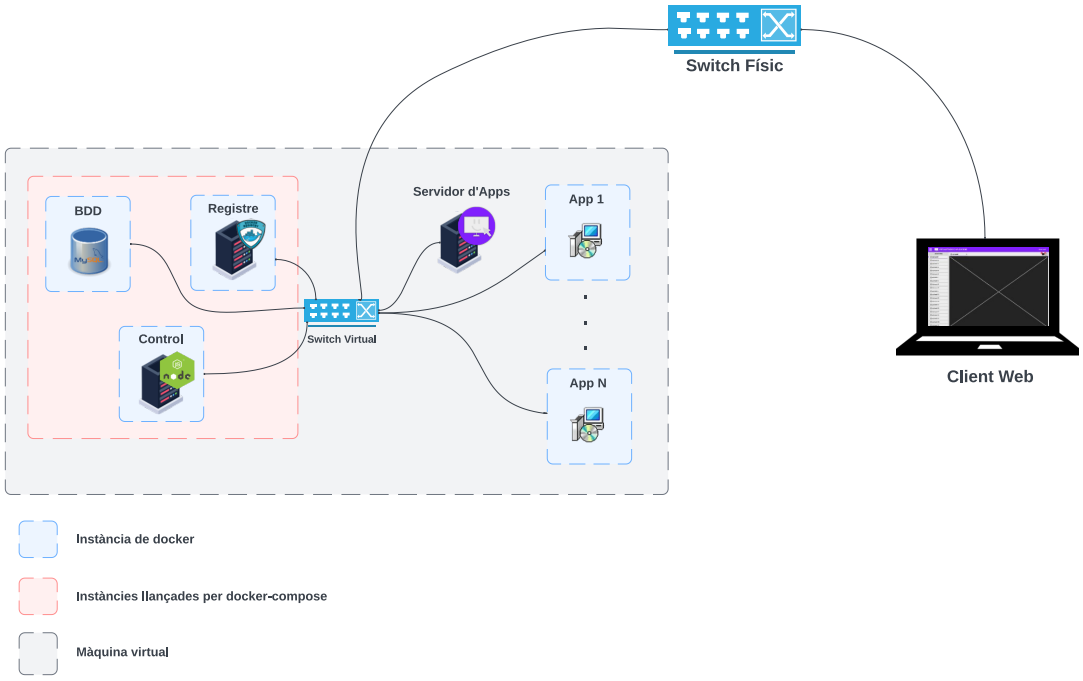

FIGURA 9.5: Escenari en xarxa local (monolític)

9.5.3 Escenari xarxa local (distribuït)

Un cop comprovada la correcta connexió en xarxa local, es pretén posar a prova la connexió usant múltiples màquines servidores d'aplicacions sota un mateix servidor de control. Aquest escenari ens permetrà provar que es pot establir connexió simultània amb diversos servidors alhora i que es comuniquen bé entre ells.

Diagrama de l'escenari en xarxa local (distribuït):

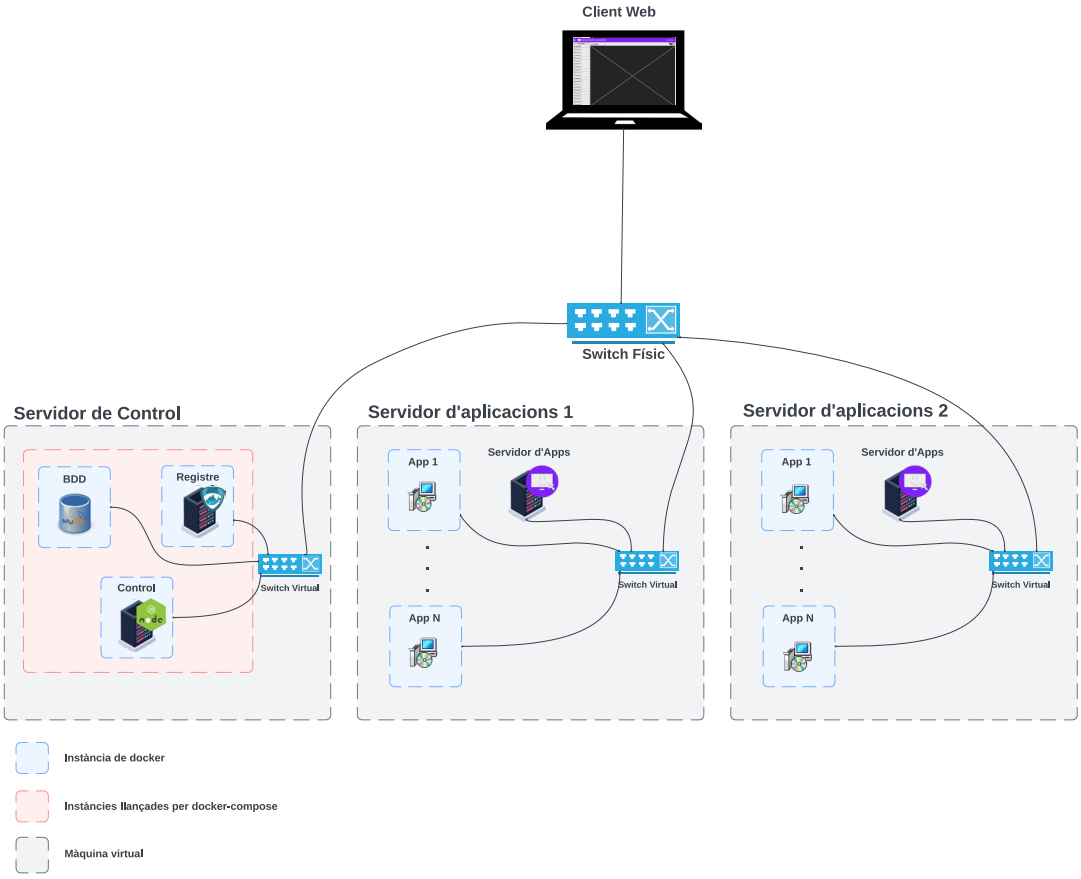

FIGURA 9.6: Escenari en xarxa local (distribuït)

9.5.4 Escenari NAT (distribuït)

Un cop comprovat que l'escenari funciona correctament en la xarxa local, tant de forma monolítica com distribuïda, resta comprovar el funcionament quan el client es troba en una xarxa aliena i darrere d'un router NAT[7]. En l'escenari proposat, les màquines servidores disposen de connexió a internet sense restriccions de NAT i cadascun dels servidors disposa de la seva pròpia IP pública. El client és l'únic element rere un router NAT restrictiu.

Diagrama de l'escenari amb client rere NAT (distribuït):

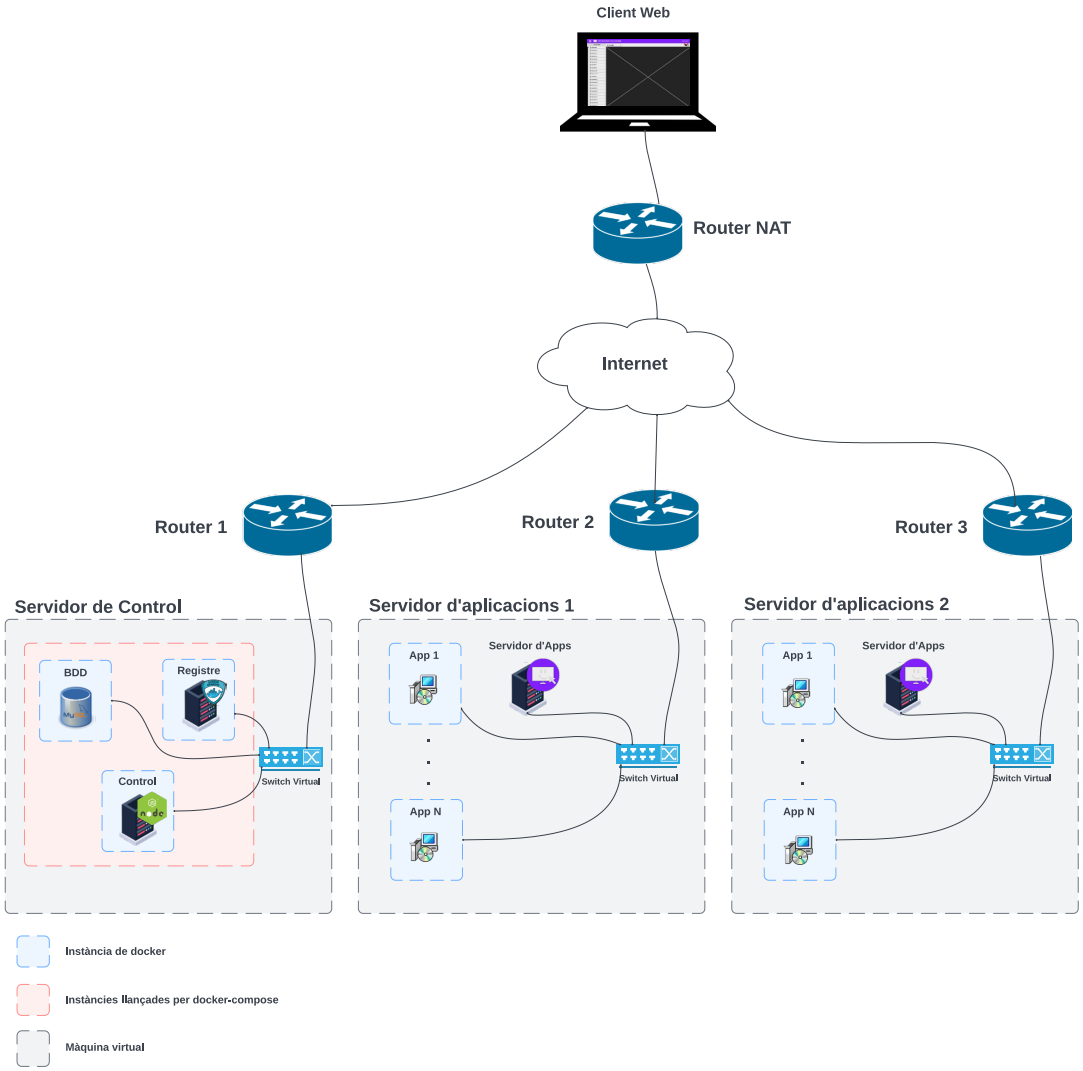

FIGURA 9.7: Escenari NAT (distribuït)

[7] Els routers NAT impedeixen les connexions directes des de fora la xarxa i pot ser una dificultat per l'establiment de la connexió WebRTC. En aquest cas, el disseny del sistema ja té previst el funcionament en aquest escenari.

10. Resultats

Aquest capítol té com a objectiu mostrar els resultats obtinguts del sistema desenvolupat, així com demostrar l'assoliment dels requisits i objectius del projecte. Per tal de fer més interactiva la visió dels resultats, s'ha fet públic un vídeo mostrant el funcionament de l'aplicació amb aplicacions d'exemple[1].

El vídeo es pot trobar en el següent enllaç: https://youtu.be/1SMrREKlg-Q

Seguidament, es mostren captures i comenten els resultats assolits en els aspectes més rellevants. Per a una demostració més detallada es recomana seguir el vídeo.

10.1 Execució d'una aplicació

La següent figura mostra un exemple d'execució de l'aplicació GIMP.

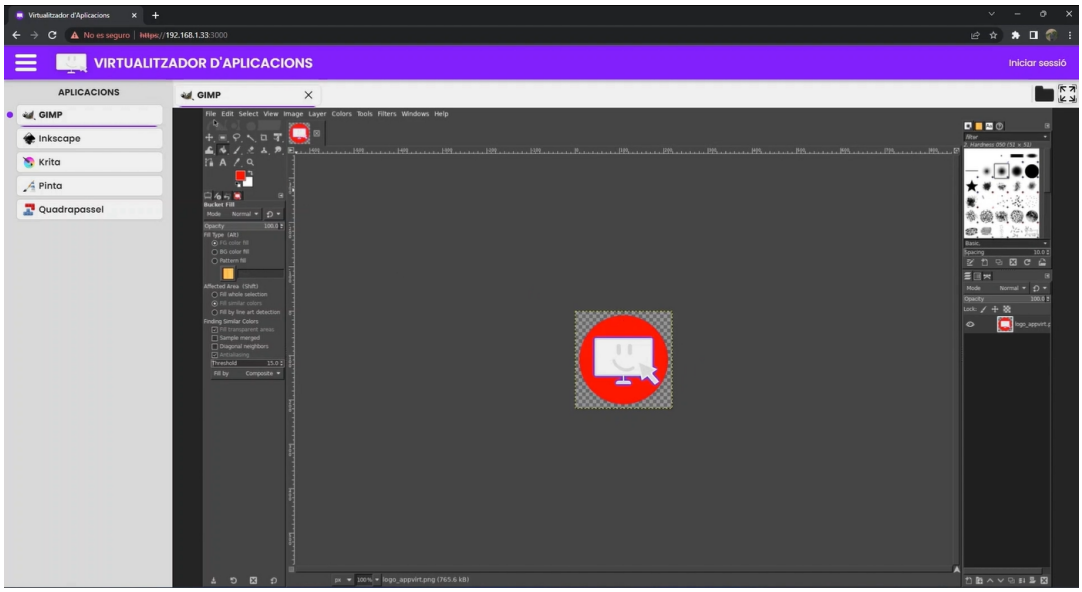

FIGURA 10.1: Execució a través de l'aplicació web de l'aplicació GIMP

[1] Aplicacions que la seva llicència en permet la distribució utilitzant el sistema desenvolupat.

10.2 Accés al sistema de fitxers

Les següents figures mostren s'accedeix al llistat d'elements emmagatzemats al volum compartit de l'aplicació i posteriorment es descarrega l'element al sistema de l'usuari.

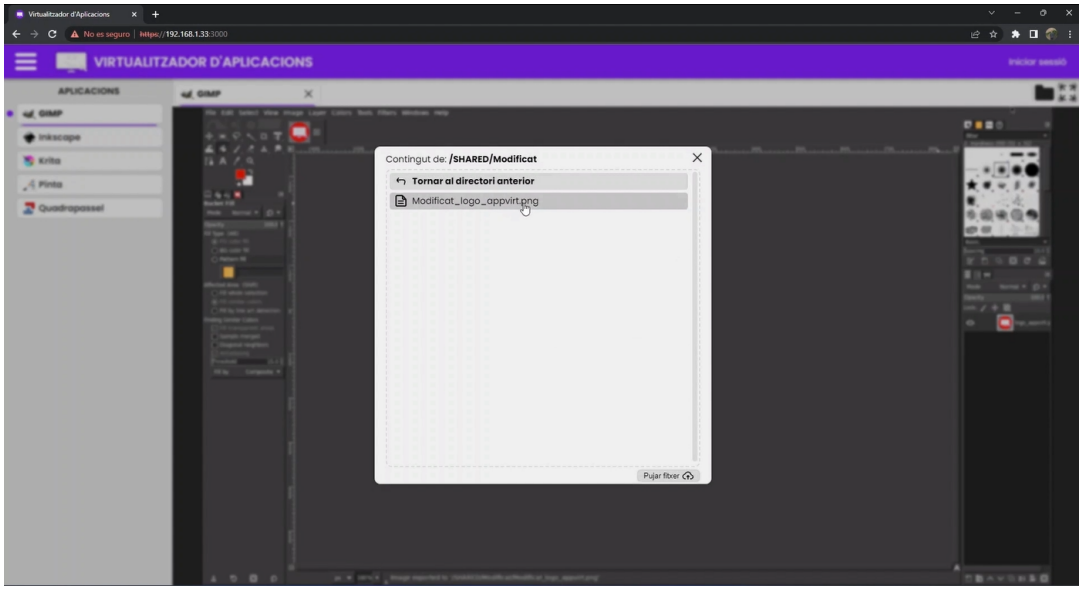

FIGURA 10.2: Llistat dels elements en el volum compartit

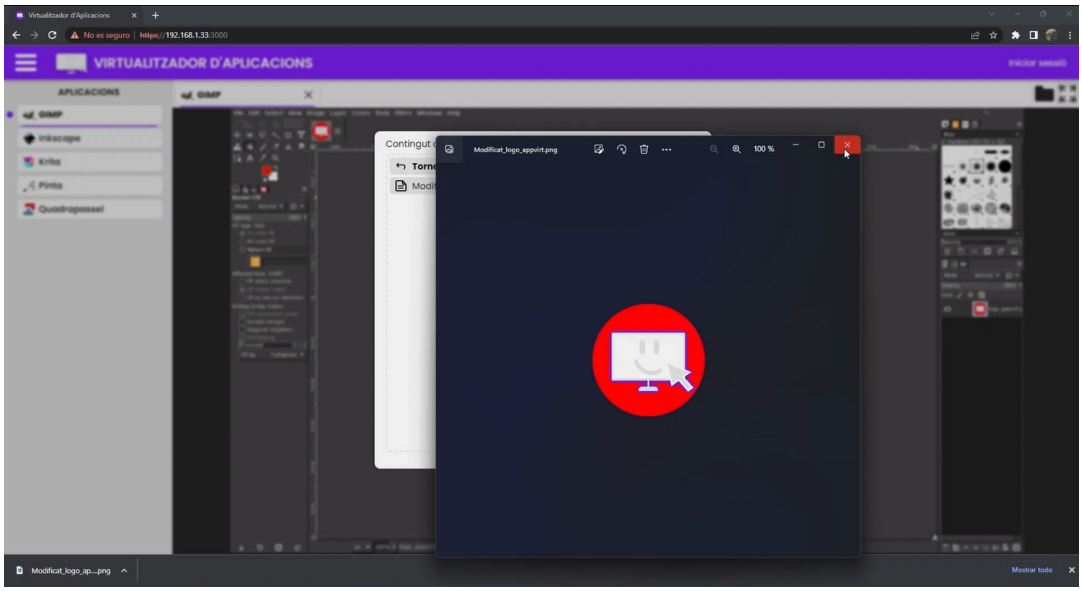

FIGURA 10.3: Descàrrega i accés a l'element descarregat

10.3 Visionat i control en pantalla completa

La següent figura mostra un exemple d'execució de l'aplicació però en visió i control a través de la pantalla completa.

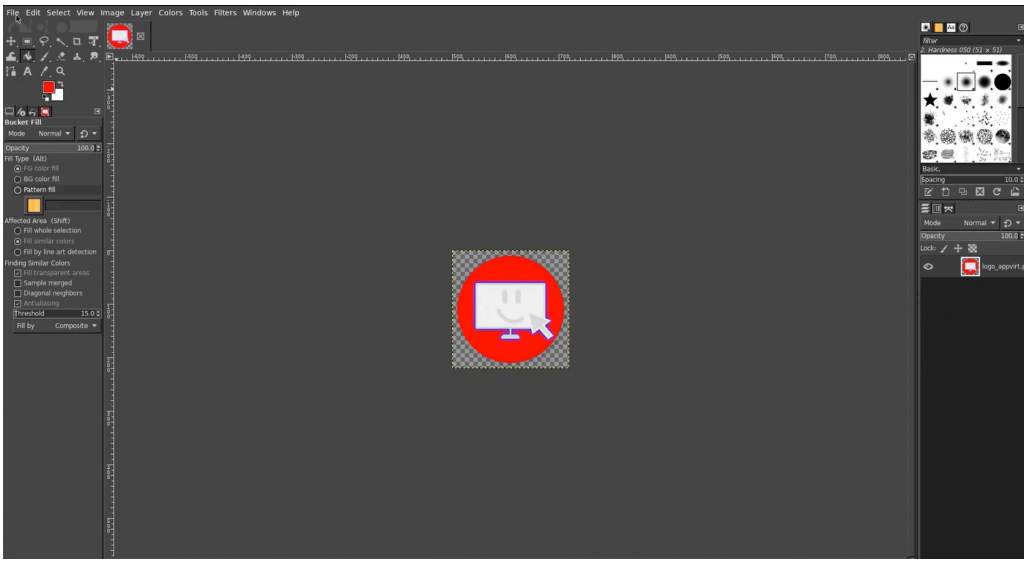

FIGURA 10.4: Visió i control en pantalla completa

10.4 Múltiples execucions simultànies

La següent figura mostra un exemple on es troben diverses aplicacions en execució simultània. A la barra i llistat de navegació es pot apreciar quines aplicacions es troben obertes i quina en primer pla.

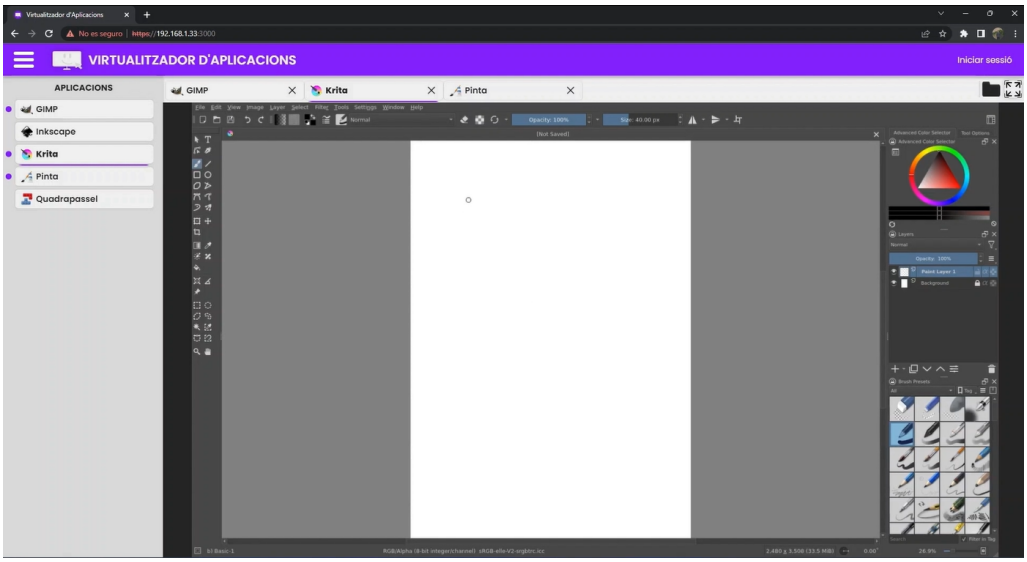

FIGURA 10.5: Múltiples execucions simultànies

10.5 Registre i identificació

Les següents figures mostren les pantalles de registre i identificació d'usuari

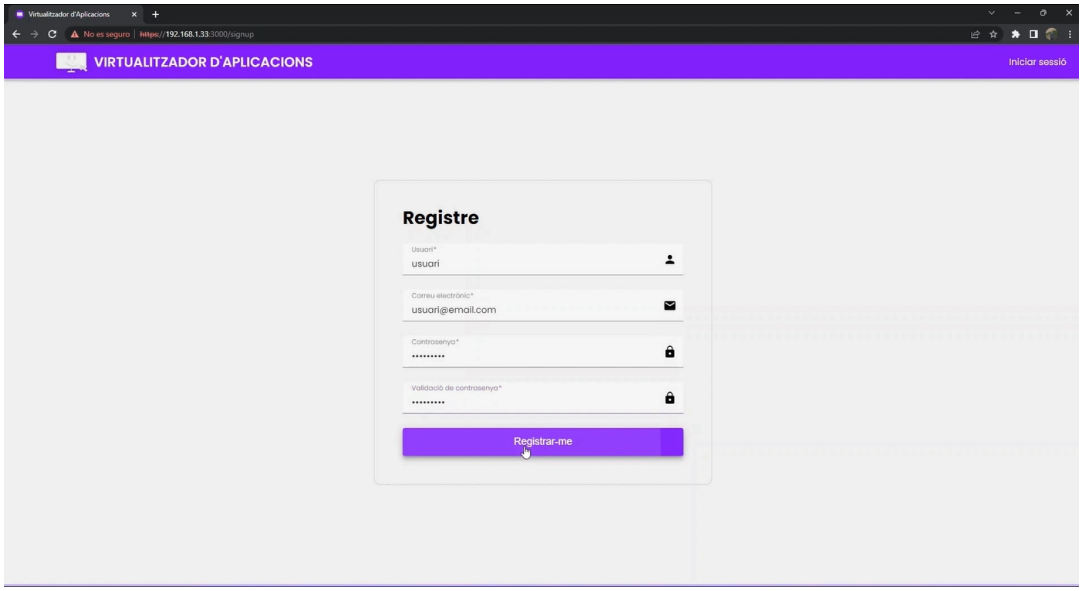

FIGURA 10.6: Formulari de registre d'usuari

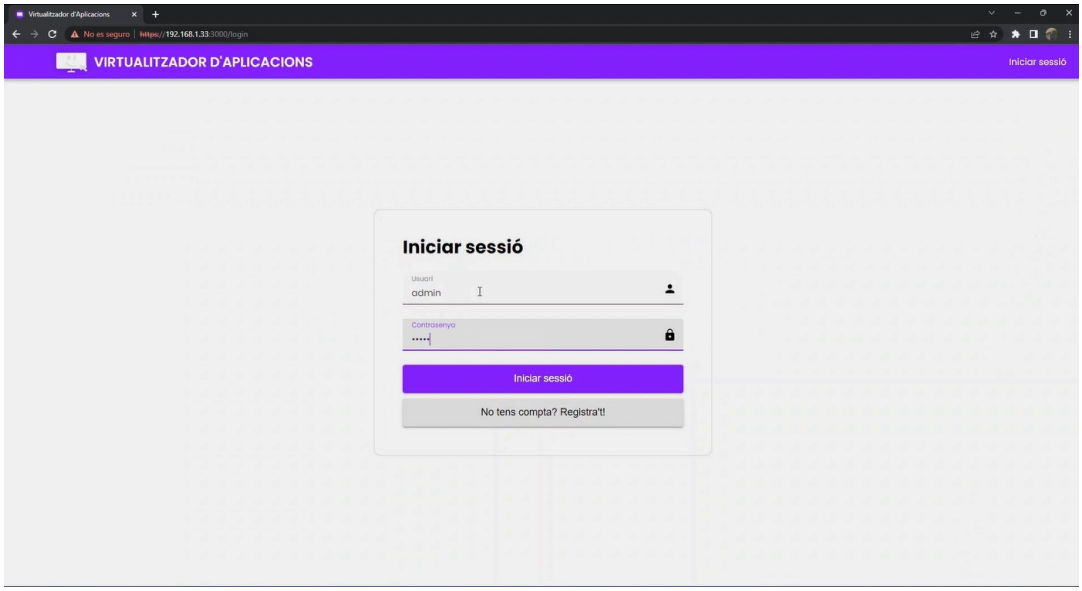

FIGURA 10.7: Formulari d'identificació d'usuari

10.6 Gestió del servei

10.6.1 Gestió d'aplicacions i permisos

La següent figura mostra la pantalla de gestió de les aplicacions juntament amb el control de permisos d'accés.

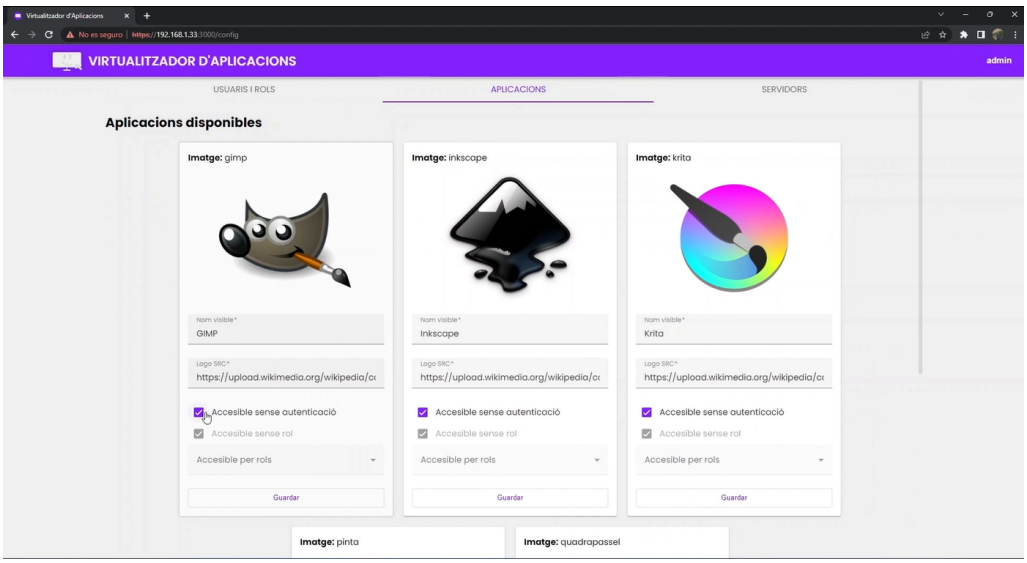

FIGURA 10.8: Gestió d'aplicacions i permisos

10.6.2 Gestió d'usuaris i rols

La següent figura mostra la pantalla de gestió d'usuaris i rols d'usuari.

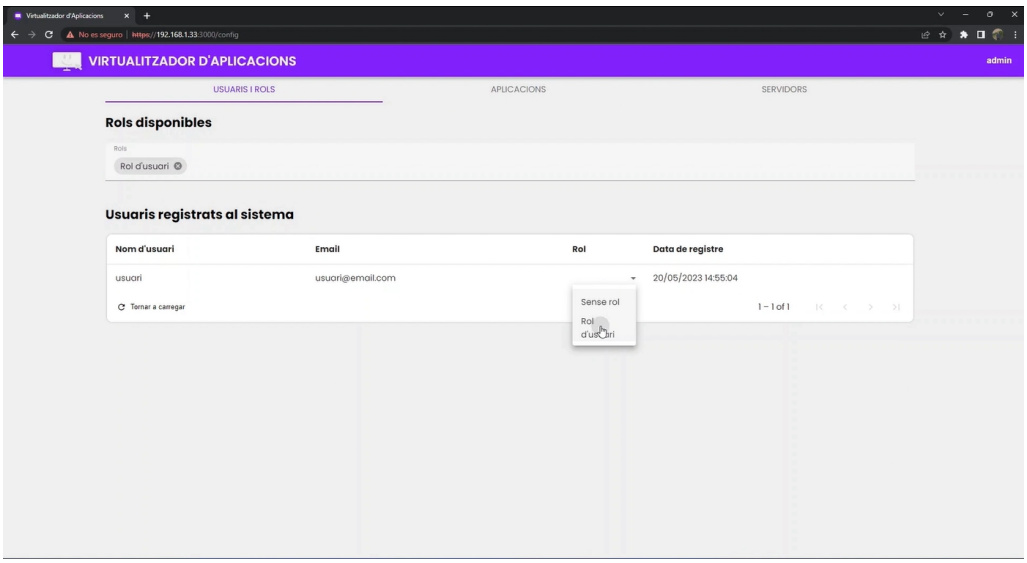

FIGURA 10.9: Gestió d'usuaris i rols

10.6.3 Visió de l'estat dels servidors d'aplicacions

La següent figura mostra la pantalla de visió dels múltiples servidors d'aplicacions que actualment estan actius.

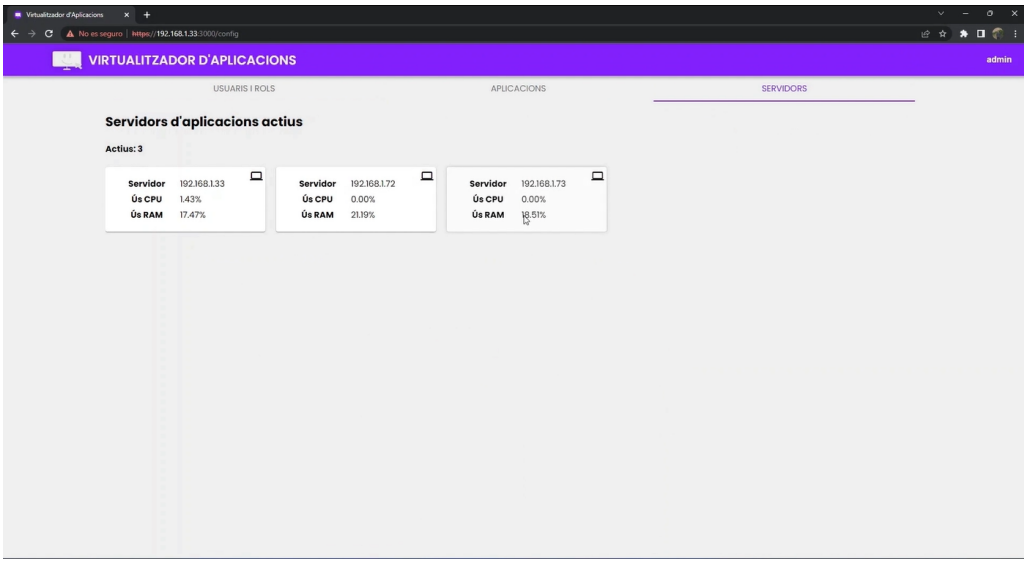

FIGURA 10.10: Visió de l'estat dels servidors d'aplicacions

10.7 Comentari sobre els resultats obtinguts

El sistema desenvolupat compleix amb tots els requisits funcionals especificats juntament amb els requisits de disseny que requerien la implementació d'un servei distribuït amb opció d'escalat horitzontal. L'aplicació web resultant permet a un usuari accedir al servei i executar una aplicació virtualitzada sense haver d'instal·lar ni configurar res en el seu sistema. D'altra banda, permet una implementació distribuïda amb múltiples servidors d'aplicacions que serveixin com a màquines de treball on realment s'està realitzant l'execució de les aplicacions.

11. Conclusions i treball futur

11.1 Conclusions

Fruit del treball realitzat, s'ha creat un nou servei i mètode de desplegament de programari que pot ser de rellevant interès per a aquelles empreses que actualment utilitzen serveis de virtualització i sessions remotes per a permetre als seus usuaris l'ús d'aplicacions o, fins i tot, per a centres educatius per a la distribució de software als seus estudiants sense que aquests requereixin el maquinari o sistema específic.

En finalitzar el projecte i en vista dels resultats obtinguts, es pot concloure que s'han complert tots els objectius i requisits del projecte planificats durant el seu plantejament. En concret, s'han assolit tots els objectius plantejats, i fins i tot s'ha superat les expectatives en alguns aspectes. Els resultats obtinguts han estat positius i s'ha aconseguit implementar un nou sistema de desplegament d'aplicacions on l'usuari final no requereix cap dependència. El projecte ha estat un èxit gràcies a l'encertada planificació, l'exhaustiu estudi inicial juntament amb la realització del disseny i desenvolupament adequat del sistema.

11.1.1 Desviacions en la planificació

En general el projecte no ha patit desviacions en la planificació i s'ha pogut finalitzar cadascuna de les tasques dins la seva data planificada. Si és cert, però, que algunes de les tasques s'han complicat una mica més del previst (tal com s'ha comentat en les seccions 9.1.3 i 9.2.2 sobre el desenvolupament del virtualitzador d'aplicacions i el seu respectiu servei distribuït), tot i el retard sofert en aquestes tasques, d'altres s'han pogut completar amb més agilitat a causa de l'estudi previ i a l'adequat disseny del sistema, fent així, que la planificació a l'engròs no s'hagi vist afectada.

11.1.2 Comentari personal

El projecte realitzat ha estat gratificant i m'ha proporcionat una gran satisfacció personal. Ha estat increïble veure com una idea inicial ha pres forma i s'ha convertit en una implementació real i funcional. Durant aquest procés, he tingut l'oportunitat d'explorar i utilitzar una àmplia gamma de tecnologies, aplicar coneixements adquirits durant el transcurs del grau universitari i alhora ampliar-los en diversos àmbits.

11.2 Treball futur

El projecte presenta un desenvolupament inicial del servei i aquest realitza les opcions bàsiques per a l'execució d'aplicacions virtualitzades a través de la web. Tot i això, el projecte es pot millorar o ampliar en molts aspectes. En concret, es pot estendre la lògica de negoci de la web per a permetre més casos d'ús i agrupacions d'usuaris i permisos. D'altra banda, en l'àmbit de sistema, s'hi poden implementar altres tecnologies de retransmissió i control i deixar escollir a l'usuari administrador quina aplicar en cada cas.

Per tal d'elaborar un possible treball futur, es llisten alguns dels punts que poden ser interessants:

- Afegir casos d'ús en l'administració d'aplicacions
- Permetre l'execució d'aplicacions en finestres flotants fora del navegador principal
- Planificar un sistema de permisos més ampli
- Implementar múltiples tecnologies de transmissió i control de l'aplicació com ara la plantejada amb noVNC i cedir a l'usuari administrador l'elecció de quina usar en cada aplicació del sistema
- Implementar una distribució en mode escriptori de la web a partir d'eines com ElectronJS
- Implementar la vista i control d'aplicacions en dispositius mòbils.
- Millorar la imatge base del virtualitzador per tal d'integrar GPU sobre hardware específic

Quant a la implantació del projecte, seria essencial provar-lo en un entorn real on múltiples usuaris poguessin interactuar amb ell i rebre els seus comentaris.

A. Manual d'administrador

Aquest manual té com a objectiu formar a l'usuari administrador els passos a seguir per a realitzar la implantació i administració del servei.

A.1 Àmbit

L'àmbit d'aquest manual es basa en la implementació en un escenari monolític on tots els serveis són instal·lats en una mateixa màquina. Dit això, la implantació en un escenari distribuït no difereix més que en l'especificació de les variables de connexió ja mencionades en aquest manual.

Les ordres seguides per aquest manual se segueixen en un sistema operatiu Ubuntu amb versió 22.04, altres sistemes operatius poden diferir. **Per tal de facilitar el procés d'instal·lació, es proporcionen scripts d'instal·lació automatitzats per a la configuració i instal·lació de cadascun dels serveis.**

A.2 Instal·lació dels serveis

A.2.1 Prerequisits

1. Cal dispososar d'arquitectura de processador AMD64
2. Cal tenir instal·lat i actiu el servei de Docker. Per més informació sobre la instal·lació, veure'n la guia oficial: `https://docs.docker.com/engine/install/ubuntu/`

A.2.2 Servidor de control, base de dades i registre

La instal·lació del servidor de control, base de dades i registre et pot realitzar tant en el mateix equip com per separat. En aquest cas es mostra tota la instal·lació conjunta. En cas de voler d'instal·lar el sistema per separat només cal indicar-ho.

A.2.2.1 Descàrrega i execució de l'script d'instal·lació

Cal executar la següent ordre en la línia de comandes per a descarregar i executar l'script d'instal·lació:

```
wget https://raw.githubusercontent.com/AniolFernandez/ApplicationVirtualization/main/install/install_webserver.sh -O script.sh && bash script.sh
```

A.2.2.2 Configuració mitjançant script

L'script ens anirà demanant un seguit de paràmetres a configurar. Podem deixar els valors per defecte o bé adaptar-los segons necessitat. Per realitzar una instal·lació local n'hi ha prou amb utilitzar els valors per defecte. Al finalitzar l'script, **caldrà anotar la clau generada** ja que ens serà necessària per la instal·lació del servidor d'aplicacions.

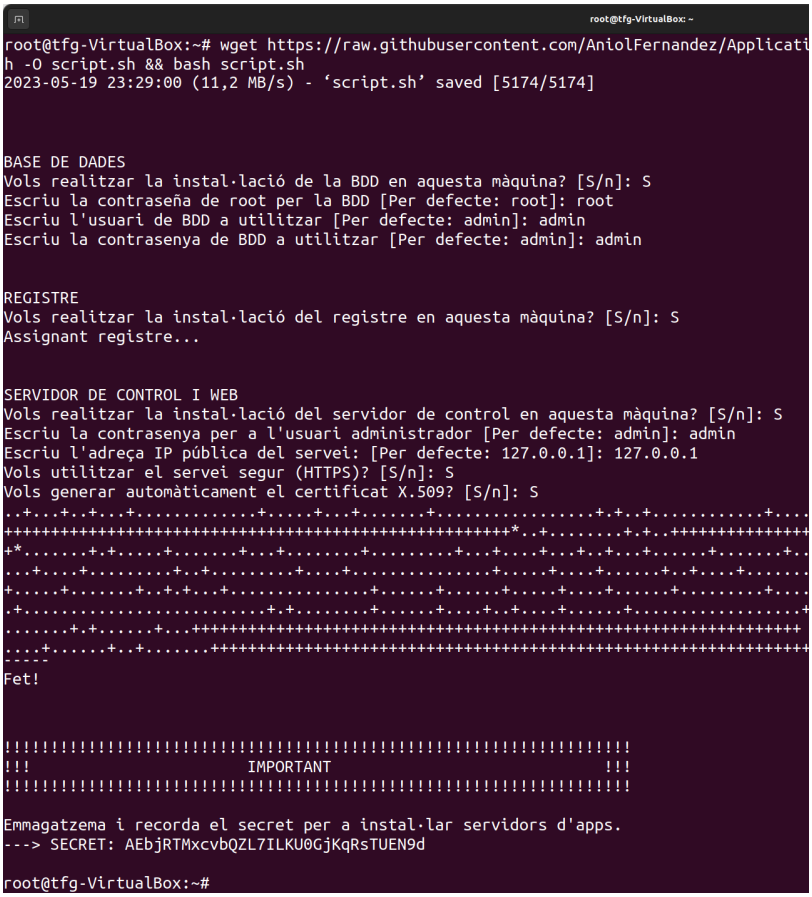

FIGURA A.1: Instal·lació de servidor de control, registre i base de dades automatitzada.

Un cop executat, ens generarà un fitxer *Dockercompose* amb la configuració necessària. El podrem executar amb l'ordre `docker compose up`.

A.2.2.3 Inicialització de la base de dades

Si és el primer cop que s'instal·la el servei, caldrà inicialitzar el contingut de la base de dades. Per a fer-ho, primerament ens assegurem que hem encés el servei amb la comanda `docker compose up` i seguidament descarreguem i executem l'script d'inicialització amb la següent comanda:

```
wget https://raw.githubusercontent.com/AniolFernandez/ApplicationVirtualization/main/install/inidb.sh -O inidb.sh && bash inidb.sh
```

Com abans, l'script ens demanarà les dades necessàries, en aquest cas el host i la contrasenya de la base de dades. Si hem deixat la configuració per defecte en la secció anterior, aquí també podem fer-ho.

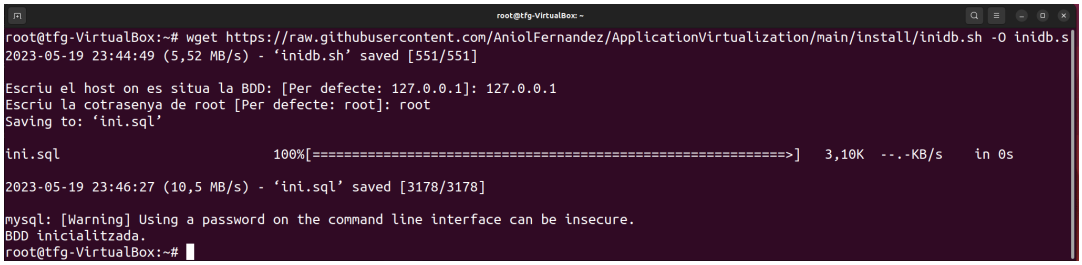

FIGURA A.2: Inicialització de la base de dades.

A.2.2.4 Execució del servei

Un cop instal·lat i configurat el servei, podem executar-lo amb la comanda `docker compose up`. Si es desitgés es podria habilitar com a servei de sistema.

A.2.3 Servidor d'aplicacions

La instal·lació del servidor d'aplicacions es pot realitzar en el mateix equip que el servidor de control, per separat, junt i separat, i de forma distribuïda. En aquest cas es mostra tota la instal·lació en la mateixa màquina que el servidor de control. En cas de voler d'instal·lar el sistema per separat només cal especificar correctament el host de destí del servidor.

A.2.3.1 Descàrrega i execució de l'script d'instal·lació

Cal executar la següent ordre en la línia de comandes per a descarregar i executar l'script d'instal·lació:

```
wget https://raw.githubusercontent.com/AniolFernandez/Applic
  ↪ ationVirtualization/main/install/install_appserver.sh -O
  ↪ script.sh && bash script.sh
```

A.2.3.2 Configuració mitjançant script

L'script ens anirà demanant un seguit de paràmetres a configurar. Podem deixar els valors per defecte o bé adaptar-los segons necessitat. Per realitzar una instal·lació local n'hi ha prou amb utilitzar els valors per defecte.
Important: Caldrà especificar el secret generat durant la instal·lació del servidor de control.

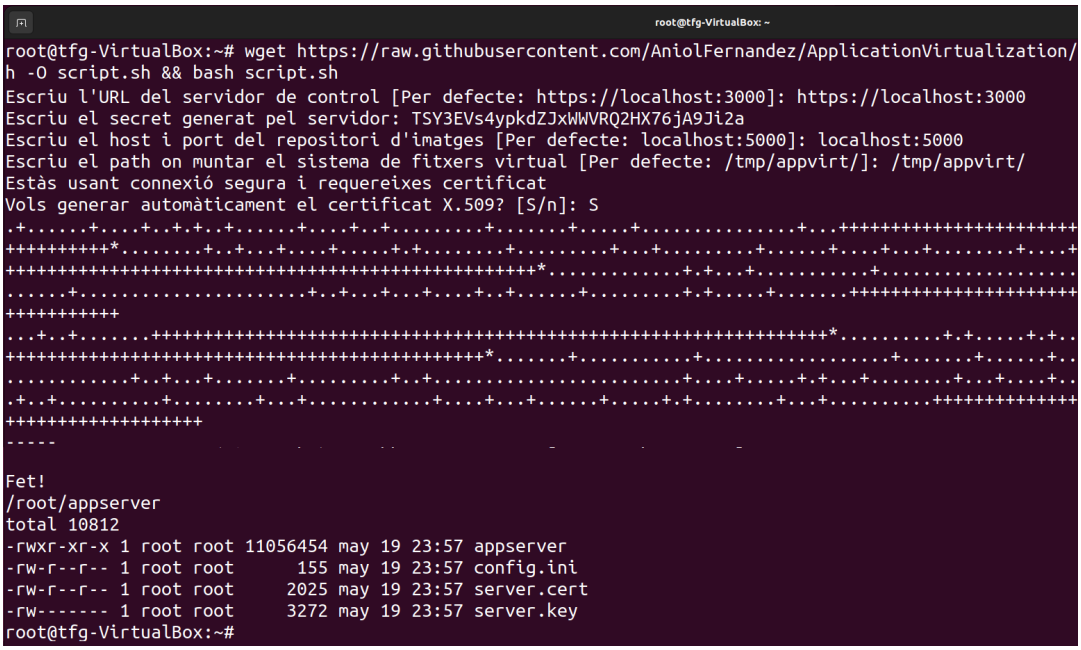

FIGURA A.3: Instal·lació de servidor d'aplicacions.

Un cop instal·lat, el podrem executar directament des del seu binari executable amb l'ordre `./appserver`

A.3 Administració del servei

A.3.1 Afegir una aplicació al servei

Un cop tenim el servei instal·lat i en execució, només resta d'afegir-hi aplicacions. Fer-ho és molt senzill ja que es proporcionen les eines bàsiques per a la creació d'imatges amb l'aplicació.

A.3.1.1 Creació d'imatge amb l'aplicació

Posem per exemple que es vol afegir al sistema l'aplicació GIMP. Per a fer-ho, només caldrà generar un fitxer *Dockerfile* que parteixi de la imatge base desenvolupada i hi instal·li l'aplicació necessària juntament amb les seves dependències.

Exemple de *Dockerfile* amb la instal·lació de GIMP:

```
#Bastat en la imatge base creada pel projecte
FROM aniolfdz/appvirt:base

#Root per instal·lar dependències
USER root
ADD run.sh /run.sh

#Instal·lem l'aplicació a executar
RUN apt-get install gimp -y \
&& chmod +x /run.sh \
&& chown USER:USER /run.sh

USER USER
```

En el mateix directori que el fitxer *Dockerfile*, caldrà afegir un script anomenat `run.sh` amb les ordes necessàries per a l'execució de l'aplicació. En aquest cas seria el següent:

```
#!/bin/bash
gimp
```

Un cop disposem dels fitxers amb la instal·lació de dependències i execució, en generem una imatge amb la comanda:

```
docker build . -t <nom_imatge>
```

on <nom_imatge> pot ser l'identificador que vulguem. Cal procurar que sigui diferent d'altres imatges que ja disposem.

A.3.1.2 Addició de la imatge al registre

Un cop disposem de la imatge generada, cal afegir-la al registre del servei. Per a fer-ho, primerament caldrà especificar un tag per a la imatge on s'especifiqui el seu nom juntament amb el registre d'imatges del sistema. Per a fer-ho usem la següent comanda:

```
1 docker tag <nom_imatge> <registre>/<nom_imatge>
```

Un cop identificada la imatge amb el tag, només restarà pujar-la al registre. Ho aconseguim amb la següent comanda:

```
1 docker push <registre>/<nom_imatge>
```

Exemple: En el nostre cas, si suposem una instal·lació local del registre i un nom d'imatge com "appgimp", la següent captura mostraria l'exemple d'execució:

```
root@tfg-VirtualBox:~/GIMP# docker tag appgimp localhost:5000/appgimp
root@tfg-VirtualBox:~/GIMP# docker push localhost:5000/appgimp
Using default tag: latest
The push refers to repository [localhost:5000/appgimp]
b82459ae0113: Pushed
7bd7a52115bd: Pushed
5f70bf18a086: Pushed
64d8b5efb104: Pushed
8cc53085e7a5: Pushed
98088c5e6c39: Pushed
fa174815b855: Pushed
b93c1bd012ab: Pushed
latest: digest: sha256:b32ebb923b91ad85f286ace717f7962ed362cd576229c4ddf54d1fd6b
054efb2 size: 2197
```

FIGURA A.4: Addició de la imatge de GIMP al registre.

A.3.1.3 Configuració de la imatge i permisos

Finalment, un cop la imatge és disponible al registre, només resta configurar-ne la informació bàsica i de permisos. Per a fer-ho, accedim al servei web i ens identifiquem com a usuari administrador (usuari: admin i la contrasenya especificada durant la instal·lació del servei).

1. Un cop identificats, accedim al panell de configuració
2. Dins del panell de configuració accedim a l'apartat d'aplicacions i cerquem per l'aplicació acabada d'afegir
3. Un cop localitzada, en configurem el nom visible, el logo a mostrar i els permisos d'accés i premem el botó de desar canvis.

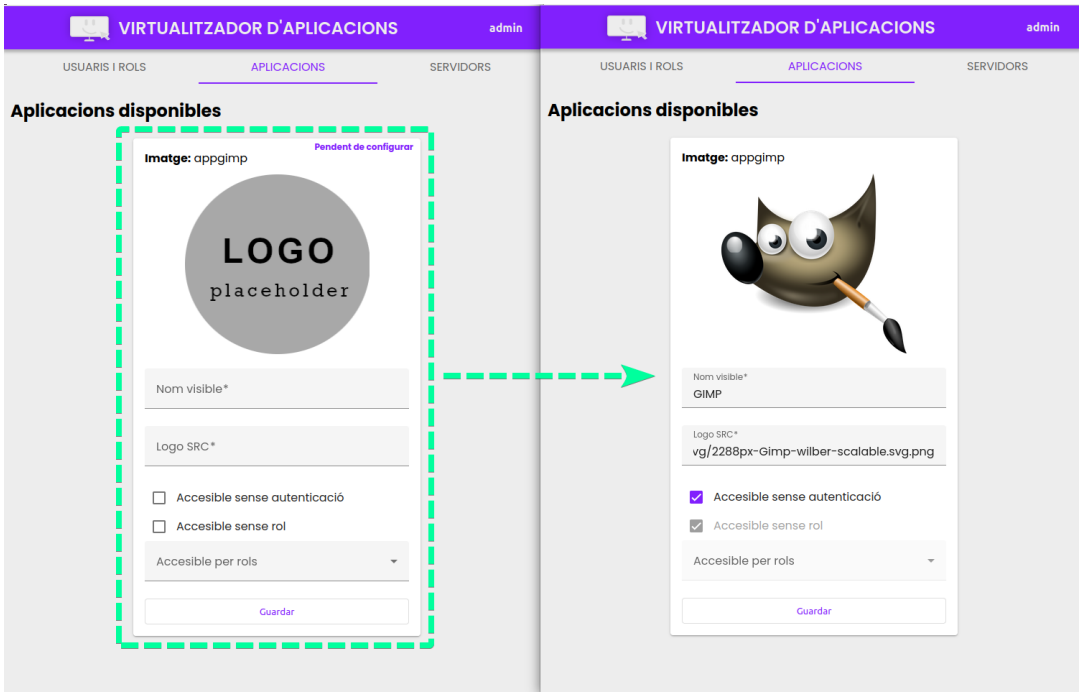

FIGURA A.5: Configuració de l'aplicació.

A.3.1.4 Eliminació i actualització d'imatges d'aplicacions

L'eliminació i actualització de les imatges de docker de les aplicacions es duu a terme mitjançant el registre. Per a més informació sobre com actualitzar o eliminar una imatge d'un registre consultar la documentació oficial de *docker registry* `https://docs.docker.com/registry/`.

A.3.2 Gestió de rols i permisos

Gestionar els rols dels usuaris i els permisos que hi tenen sobre les aplicacions és una tasca senzilla. Com en el pas anterior, cal identificar-se com a usuari administrador (usuari: admin i la contrasenya especificada durant la instal·lació del servei).

1. Un cop identificats, accedim al panell de configuració
2. Dins del panell de configuració accedim a l'apartat d'usuaris i rols i visualitzem el seu estat
3. En la part superior (1) en la figura A.6, podrem afegir, editar i eliminar els diferents rols dels usuaris.
4. En la part inferior (2) en la figura A.6, podrem visualitzar els usuraris registrats al sistema i seleccionar el seu rol.

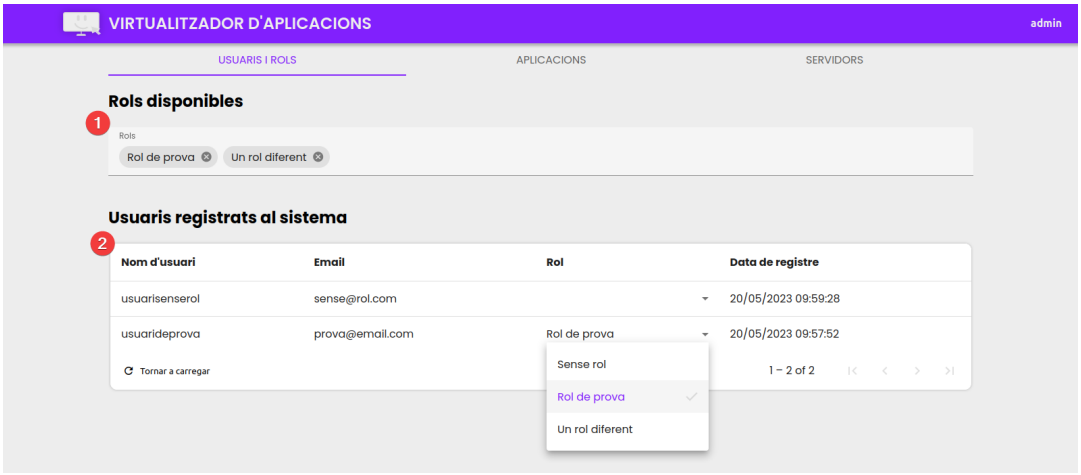

FIGURA A.6: Gestió de rols i usuaris.

Un cop gestionat els rols dels usuaris, en l'apartat d'aplicacions podrem especificar per cadascuna de les aplicacions tres nivells d'accés:

1. Accessible per a tothom (usuaris identificats i no identificats).
2. Accessible sense rol (només usuaris identificats).
3. Accessible per rols (només usuaris identificats que compleixin un dels rols especificats).

B. Annexos

El codi i imatges de docker amb els serveis s'ha fet públic a través de repositoris al núvol i s'hi pot accedir lliurement. Addicionalment, s'ha fet públic un vídeo amb la demostració de la vista d'usuari.

Vídeo de demostració
Es pot trobar la demostració en el següent enllaç:
https://youtu.be/1SMrREKlg-Q

Accés al repositori de codi
Tot el codi desenvolupat està disponible en el següent enllaç:
https://github.com/AniolFernandez/ApplicationVirtualization

Imatge base de l'aplicació de virtualització
La imatge base per a la generació d'imatges d'aplicacions es troba disponible al següent enllaç:
https://hub.docker.com/layers/aniolfdz/appvirt/base

Imatge del servidor de control
La imatge del servidor web i de control es troba disponible al següent enllaç:
https://hub.docker.com/layers/aniolfdz/appvirt/web

Bibliografia

[1] *Native (Computing)*. Últim accés: Maig, 2023. URL: https://en.wikipedia.org/wiki/Native_(computing).

[2] *Aplicación web*. Últim accés: Maig, 2023. URL: https://es.wikipedia.org/wiki/Aplicaci%C3%B3n_web.

[3] *WebGL*. Últim accés: Maig, 2023. URL: https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API.

[4] *ThinClient*. Últim accés: Maig, 2023. URL: https://es.wikipedia.org/wiki/Cliente_liviano.

[5] *The Secure Shell (SSH) Transport Layer Protocol*. Últim accés: Maig, 2023. URL: https://www.rfc-editor.org/rfc/rfc4253.

[6] *Understanding the Remote Desktop Protocol (RDP)*. Últim accés: Maig, 2023. URL: https://learn.microsoft.com/en-us/troubleshoot/windows-server/remote/understanding-remote-desktop-protocol.

[7] *The Remote Framebuffer Protocol*. Últim accés: Maig, 2023. URL: https://datatracker.ietf.org/doc/html/rfc6143.

[8] *Citrix Virtual Apps*. Últim accés: Febrer, 2023. URL: https://docs.citrix.com/es-es/citrix-virtual-apps-desktops/technical-overview.html.

[9] *HDX*. Últim accés: Febrer, 2023. URL: https://docs.citrix.com/es-es/citrix-virtual-apps-desktops/technical-overview/hdx.html.

[10] Jonathan Deber et al. "How much faster is fast enough?: user perceptions of latency & latency improvements in direct and indirect touch". A: (2015). URL: https://www.tactuallabs.com/papers/howMuchFasterIsFastEnoughCHI15.pdf.

[11] *Global Ping Statistics*. Últim accés: Maig, 2023. URL: https://wondernetwork.com/pings.

[12] *Que és una sandbox*. Últim accés: Febrer, 2023. URL: https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/que-es-una-sandbox/.

[13] *What is a container*. Últim accés: Febrer, 2023. URL: https://www.docker.com/resources/what-container/.

[14] *HTTP Live Streaming*. Últim accés: Febrer, 2023. URL: https://en.wikipedia.org/wiki/HTTP_Live_Streaming.

[15] *NoVNC*. Últim accés: Febrer, 2023. URL: https://novnc.com/info.html.

[16] *VNC in High-Latency Environments and Techniques for Improvement*. Últim accés: Febrer, 2023. URL: https://cseweb.ucsd.edu/~pasquale/Research/Papers/globecom10t.pdf.

[17] *WebRTC API*. Últim accés: Febrer, 2023. URL: https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API.

[18] *A pure Go implementation of the WebRTC API*. Últim accés: Març, 2023. URL: https://github.com/pion/webrtc.

[19] *Trickle ICE: Incremental Provisioning of Candidates for the Interactive Connectivity Establishment (ICE) Protocol*. Últim accés: Març, 2023. URL: https://datatracker.ietf.org/doc/rfc8838/.

[20] *The X Keyboard Extension*. Últim accés: Març, 2023. URL: https://www.x.org/releases/current/doc/libX11/XKB/xkblib.html.

[21] *JSON Web Token*. Últim accés: Abril, 2023. URL: https://jwt.io/introduction.

[22] *Xvfb*. Últim accés: Febrer, 2023. URL: https://es.wikipedia.org/wiki/Xvfb.

[23] *I3 Window Manager*. Últim accés: Febrer, 2023. URL: https://i3wm.org/.

[24] *FFMPEG*. Últim accés: Febrer, 2023. URL: https://ffmpeg.org/ffmpeg.html.

[25] *HTTP Protocol Upgrade Mechanism*. Últim accés: Abril, 2023. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Protocol_upgrade_mechanism.

[26] *GO Docker SDK*. Últim accés: Març, 2023. URL: https://docs.docker.com/engine/api/sdk/.

[27] *CORS*. Últim accés: Abril, 2023. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS.

[28] *WebSocket - Origin Considerations*. Últim accés: Abril, 2023. URL: https://pkg.go.dev/github.com/gorilla/websocket#hdr-Origin_Considerations.