Comercialització d'un servei d'ad-blocking

Pol Carulla Sanchez

Keywords—ReactJS, Web development, JSX, Ad-blocking, MongoDB, NodeJS, RouterOS, MikroTik, DNS

I. Introducció

Els anuncis s'han tornat una gran part del món digital, des dels bàners que omplen les pàgines web d'informació confusa i irrellevant fins a pop-ups molestos que intenten descarregar arxius corruptes a l'ordinador. Aquest treball parteix del concepte de "ad-block", que engloba tota mena de productes o softwares que serveixin per filtrar la publicitat a les pàgines web.

Aquests softwares, anomenats "ad-blockers", es poden descarregar directament des de el nostre ordinador en un apartat d'extensions. Les més utilitzades són: "ad-block" o "adblock plus". El problema és que aquests programes funcionen en el navegador i no es poden instal·lar a dispositius mòbils. A més, moltes pàgines web ja han pres contramesures fent que només es mostri el contingut si s'han carregat els anuncis.

Això pot portar a pensar si aquesta pràctica és legal, la resposta és sí. Així ho va sentenciar un tribunal d'Hamburg l'any 2015 després de quatre mesos de judici contra un dels softwares mencionats anteriorment, "adblock plus". La resposta del tribunal va ser que els usuaris estan en el seu dret de decidir el contingut que volen veure i, per conseqüència, també el que no volen.

Un cop explicada la part burocràtica, quina és la solució a les limitacions que presenten els "ad-blockers"? Com construir una infraestructura que permeti funcionar en dispositius mòbils, que no només funcioni de forma local i intentar que les contramesures de les pàgines web no afectin?

Pi-hole, és un programari de codi lliure que permet muntar en una raspberry pi un servidor DNS molt eficient que bloqueja més de 3 milions de dominis de publicitat o de dubtosa reputació, aquest software funciona diferent de les extensions citades abans. Quan un usuari fa una petició per internet necessita carregar una sèrie d'adreces web, aquestes estan a servidors remots i es consulten mitjançant el servei creat per pi-hole que elimina automàticament tots els URL de publicitat. Com és un servidor DNS podrà atendre totes les peticions de múltiples usuaris i des de qualsevol plataforma: sigui ordinador, tauleta, dispositiu mòbil o consola. Per tant, només integrant Pi-hole es resoldria el problema de les contramesures i de la compatibilitat de dispositius.

Per resoldre l'últim punt i accedir a aquest servei des de qualsevol lloc es necessita muntar un servei VPN per sobre que permeti establir una connexió segura entre el servidor DNS i la persona que vulgui utilitzar el servei. MikroTik, una companyia encarregada del desenvolupament de routers, i un encaminador programable de la seva línia "RouterBoard" és la solució perfecta per muntar un servidor OpenVPN i reencaminar el tràfic a la raspberry. A la figura 1 es mostra el sistema de forma genèrica, on l'usuari fa una petició al VPN que s'envia al servidor DNS per obtenir la resposta.

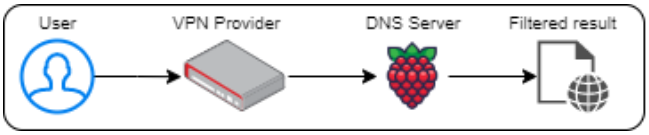

Fig. 1. Arquitectura genèrica del servei

Aquest treball consisteix en la incorporació d'un front-end i un back-end que permetin comercialitzar el servei explicat anteriorment. Els clients pagaran una subscripció mensual o anual per obtenir els certificats de la xarxa VPN sense anuncis. Els usuaris es podran connectar amb el seu usuari a la pàgina web que s'ha creat, un cop allà tindrà un apartat de clients que podrà veure els certificats que té i fins a quant de temps li queda de subscripció. Els certificats es poden descarregar i amb un client d'OVPN connectar-se directament al servei del MikroTik, d'aquesta forma podran navegar sense publicitat des de qualsevol lloc. Per la part del back-end, s'ha d'integrar una base de dades segura per guardar la informació d'aquests clients i crear el codi per generar una restFul API per a les peticions de la web. El back-end també s'encarregarà de la creació i gestió dels certificats per als usuaris, així com la seva revocació quan no es renovi la subscripció mensual.

II. Metodologia

El desenvolupament d'aquest projecte es fa seguint una metodologia àgil, ja que en utilitzar React es té l'avantatge de separar els nostres objectius en components senzills que després es poden cridar entre ells i, al tractar-se de moltes tasques diferents i concretes, es necessita una organització espacial i temporal. A més, el desenvolupament es fa fent servir una instància de GitLab en un servidor local, eina que permet tenir un seguiment de versions, i poder obrir incidències quan es troben errors en el codi. Es treballa amb dos projectes diferents, un primer és el front-end, on estan tots els arxius necessaris per crear la pàgina web; i el back-end, que conté els scripts de creació de les bases de dades, i els codis necessaris per crear la restFul API sobre la que farà les peticions el front-end. En ser part d'un projecte més gran d'empresa les tasques estan dividides en grans projectes pare: programació, administració i comercialització.

III. Objectius

L'objectiu d'aquest treball és aconseguir una plataforma web funcional que sigui capaç d'interactuar amb el servei d'VPN creat pel MikroTik creant i imprimint els certificats requerits, que consti d'un sistema de pagament via subscripció i seguretat pels usuaris, ja que el client introduirà dades personals i targetes bancàries. Per realitzar aquest objectiu principal, s'ha dividit el problema en objectius secundaris que dividiran el treball en apartats mes sencills. D'aquests objectius hi han de quatre tipus:

- Estudi (**E**): Es refereix als objectius que no necessiten implementació. Son treballs de documentació, anàlisis d'informació que acaben resolvent un dubte.
- Creació i Configuració (**C**): En els objectius d'aquest tipus s'instal·la i s'implementa un component de forma individual.
- Adaptació (**A**): Les adaptacions son les accions que s'han de realitzar per permetre la comunicació entre dos elements diferents.
- Validació (**V**): Objectius que es centrin en el testeig del correcte funcionament d'un component.

A la "Taula-1:Objectius" del annex, es pot veure la classificació dels objectius secundaris.
Els esquelets dels dos primers blocs s'han desenvolupat en paral·lel, ja que no tenen dependències entre ells, però després s'ha treballat de forma simultània entre els dos blocs. Abans de començar amb els objectius de tipus **A** s'han d'haver finalitzat les seves dependències amb els objectius de tipus **C** corresponents. L'anàlisi de vulnerabilitats s'ha fet un cop ha estat el sistema en fase de proves, seguit de l'estudi dels mètodes de pagament en criptomonedes.

IV. Planificació

Per decidir la planificació a seguir s'han tingut en compte els objectius explicats anteriorment i la duració s'ha calculat de forma coherent amb els meus coneixements i la dificultat que la tasca proposa. El primer propòsit ha estat una investigació inicial per decidir quin entorn és més adient pel treball a realitzar (React,Vue o Angular), és una tasca de recerca ha tingut una duració d'un dia. Un cop triat l'entorn de treball s'ha treballat en quatre tasques diferents que formaran els pilars d'aquest treball:

- Creació del back-end
- Creació del front-end
- Creació de la base de dades
- Creació del servei VPN

Aquestes tasques engloben el funcionament individual de cada apartat, podent-les fer de forma paral·lela. En el cas d'aquest treball, al tractar-se de només un programador, s'han realitzat de forma seqüencial seguint l'ordre anterior. Cadascuna d'aquestes tasques està dividida en petites tasques més simples com s'ha comentat anteriorment a l'apartat de metodologia. Aquestes subtasques marquen petits objectius, durant els quatre primers blocs les subtasques que es segueixen no es poden paral·lelitzar. Un cop fets els punts anteriors, s'ha treballat amb una de les tasques més extenses: la comunicació entre cadascun d'aquests pilars i l'adaptació dels codis per funcionar amb les llibreries utilitzades; a més, de forma paral·lela, s'han anat duent a terme les proves de validació i testeig. Durant aquests dos últims blocs la varietat de les subtasques és més gran, ja que es treballa amb diferents softwares, això permet realitzar les subtasques de forma simultània.

Un cop implementades totes les funcionalitats de la web s'ha fet una anàlisi de vulnerabilitats del front-end, del back-end, de la base de dades i del servei VPN. Com s'han d'analitzar diferents factors, canviar codis, canviar llibreries per problemes de seguretat..., és una de les tasques que més temps porta amb una duració de deu dies. Per acabar, s'ha fet una investigació sobre la implementació de les noves tecnologies de criptomonedes a la pàgina web amb una duració de tres dies.

Al final no s'utilitza la llibreria mongoose, ja que al treballar exclusivament amb una raspberry hi ha una limitació per culpa de l'arquitectura ARM que utilitza. No hi ha una bona compatibilitat entre mongoDB i el seu connector a JavaScript perquè la versió de mongoDB era molt inferior, és un error explícit de ARM.
Les taules 2.1 i 2.2 de l'annex es mostren la nova planificació tenint en compte els canvis fets. L'anàlisi de vulnerabilitats s'ha fet per components, i durant la programació s'ha intentat només donar l'accés necessari a cada element. Per exemple, la base de dades consta d'un usuari exclusiu per interactuar amb node i només te accés a comandes de "select" o "insert". A la "Figura 1: Diagrama de Gantt" de l'annex hi ha una vista de la programació seguida per fer les tasques.

V. Eines i Tecnologies

En aquesta secció es tracten les eines o els diferents softwares que s'han utilitzat per a la realització del treball, així com algunes altres que no s'han utilitzat, però s'han plantejat com a possibles candidates.

A. ReactJS

ReactJS és una llibreria de JavaScript basada en components i, actualment, una de les llibreries de JavaScript més usades al món segons Google Trends. S'encarrega de la Vista en un model MVC (Model Vista Controlador) i es caracteritza pels seus components reactius que permeten la creació d'aplicacions web complexes en què els seus elements són reactius i poden canviar el seu estat sense haver de recarregar la pàgina, proporcionant una millor experiència d'usuari i fluïdesa. El front-end d'aquest treball estarà desenvolupat amb aquesta llibreria, juntament amb altres llibreries més petites que s'explicaran al llarg del document.
D'aquesta llibreria hi ha dos grans pilars per entendre el seu

funcionament: El react DOM i la separació en components.

1) React DOM: El DOM (les sigles del qual signifiquen "Document Object Model") és la representació de la interfície gràfica de la nostra aplicació. Per tant, cada cop que l'estat de l'aplicació canvia també ho farà aquesta interfície per adaptar-se a les modificacions introduïdes. No obstant això, actualitzar el DOM és una tasca costosa en quant a rendiment es refereix a causa de la seva estructura en forma d'arbre, on si s'actualitza un node pare, tots els seus fills s'hauran de tornar a crear.
React ofereix un DOM virtual, més eficient i lleuger, augmentant així el temps de resposta de la nostra aplicació web. Això és perquè ReactJS no interactua directament amb el DOM generat pel navegador sinó que disposa d'un propi emmagatzemat en memòria. ReactJS només actualitza el que és necessari, ja que en tenir el DOM desat pot comparar els seus elements i fills i només aplicar els canvis explícitament necessaris perquè el DOM del navegador tingui el format desitjat.

2) Divisió en components: Els components a ReactJS són conceptualment similars a les funcions en JavaScript i permeten a l'usuari separar l'UI en peces independents i reutilitzables. Els components accepten uns atributs i tornen un nou element propi de React amb què ha d'aparèixer per pantalla. Aquests atributs o inputs que els passem als components són només de lectura i no han de ser modificats, a aquest tipus de funcions se les denomina com a funcions pures, i el seu resultat sempre serà el mateix per a un determinat o determinats inputs.[4] Una altra funcionalitat dels components és que són capaços de tornar altres components o una estructura que involucri diferents components més senzills. Es poden separar en dos tipus: els class components i els function components. Tots dos han de respectar la característica React de començar el nom del component en majúscules, això es fa per evitar que JSX confongui els components amb tags d'HTML.[5]

A continuació, es fa una breu comparativa dels altres dos candidats a fer de front-end comparant-los amb ReactJS. La decisió final ha estat presa de forma personal ja que cadascun te els seus punts forts i febles i tots tres podrien utilitzar-se per desenvolupar el mateix entorn.

B. VueJS

Tant Vue.js com React.js fan servir DOM virtuals però amb implementacions diferents. React i Vue són lleugers, posseeixen una arquitectura basada en components i exposen mètodes de lifecycle [7]. El seu acompliment és força similar. Ambdues tecnologies funcionen amb qualsevol aplicació web existent, encara que no és una aplicació de pàgina única. Aquest és el cas de l'editor de Gutenberg, està construït amb React i es va implementar recentment a l'ecosistema de WordPress. Similar és el cas de GitLab, on el codi base de jQuery està sent reemplaçat gradualment per Vue. Cal destacar que Vue és un framework mentre que React una llibreria, en aquesta última, el desenvolupador és l'encarregat d'on i quan trucar-la, trucant només els recursos necessaris. Per contra, un framework s'encarrega d'anomenar-lo els elements o parts de codi que necessita. És per això que un framework sol relacionar-se amb la facilitat d'implementació i una llibreria amb la flexibilitat. La diferència principal és l'ús del llenguatge JSX de React, VueJS utilitza templates que es declaren per renderitzar l'objecte i faciliten implementacions més senzilles. A més sent Vue un framework, pot ser integrat en un projecte ja existent, creant noves implementacions interactives per seccions.

Un cop dit això, no es pot obtenir un resultat d'aquesta comparativa, tots dos tenen un gran ventall de possibilitats respecte a la creació d'aplicacions web. No hi ha una gran diferència de rendiment, ni un salt pel que fa a nivell de dificultat. La decisió queda a les mans del desenvolupador i les seves preferències personals.

C. AngularJS

AngularJS és un framework de codi obert desenvolupat per Google el 2009. Tant ReactJS com AngularJS són de gran utilitat a l'hora de crear llocs web interactius i, juntament amb Vue.js, Angular és dels frameworks de Java més populars i eficients a tot el món. Aquest marc estructural està dissenyat especialment per simplificar el procés de desenvolupament de front-end del lloc web. El framework AngularJS és popular principalment perquè aprofita l'ús d'HTML com a llenguatge de plantilla, però el desavantatge d'usar HTML és que no és un llenguatge eficient a l'hora de desenvolupar aplicacions web.

Igual que React amb el sistema per evitar el cross-site scripting, Angular també disposa d'una gran robustesa quant a seguretat es refereix ja que utilitza serveis web com RESTful API com a interfície HTTPS per interactuar amb els servidors i mostrar la informació. Gràcies a això Angular protegeix les empreses de possibles codi maliciós o accessos no autoritzats. Una altra diferència que té Angular és l'ús de diferents estructures, que dificulten aprendre a fer-les servir. En canvi, tant React com Vue se centren en els components com a úniques estructures capaces de realitzar totes les funcionalitats.

D. NodeJS

NodeJS és l'encarregat de la capa del servidor de la pàgina web. És un entorn basat en JavaScript i juntament amb diferents llibreries serà l'encarregat de passar els fils d'informació de ReactJS fins a la base de dades de MariaDB. La idea principal de NodeJS és construir una comunicació lleugera, segura i eficient de les dades que necessiti l'aplicació en temps real. Es pot descarregar en qualsevol dispositiu linux i windows a traves del instalador de paquets NPM que es comenta també a l'apartat d'implementació, on s'explica el seu funcionament.

E. RouterOS

RouterOS es el sistema operatiu que utilitzen tots els dispositius de MikroTik. Es tracta d'un sistema de codi obert

que es pot instal·lar a qualsevol ordinador convertint-lo en un router programable. Winbox es l'interfície gràfica que s'utilitza i proporciona una visió molt simple de totes les possibilitats que ofereix el SO.

VI. Disseny

En aquest treball hem de tenir en compte dues arquitectures diferents. La primera és l'arquitectura del servei que s'ofereix i la segona l'arquitectura de la pàgina web.

A. Arquitectura del servei

Per a definir l'arquitectura del servei s'ha utilitzat la distribució que apareix a la "Figura 2: Arquitectura del servei" que es troba a l'annex, els nodes són els següents:

- DDNS Provider: Aquest es un servei que permetrà registrar la nostra ip pública com un domini.
- MikroTik RouterBoard: És el dispositiu encaminador que utilitzarem per fer el servidor de OpenVPN i la porta d'entrada per als usuaris del servei.
- RaspBerry Pi: Aquesta raspberry portara el software de "pi-hole" i actuara com a servidor DNS per al nostre trafic.
- ISP: Aquest element es refereix al proveÏdor que permet al mikrotik conectar-se a internet.
- Client X: Usuari client conectat al Mikrotik i navegant utilitzant el DNS de la raspberry.

El primer que es necessita perquè funcioni el servei és obtenir una IP pública, hi ha dues formes d'obtenir-la. La primera és demanant una IP pública estàtica al proveïdor d'Internet, és una solució eficaç, però té un preu mensual mantenir-la. La segona opció, escollida per aquest treball, és utilitzar un proveïdor d'un servei DDNS, que s'encarregarà d'assignar al domini que es triï la nostra ip publica dinàmica. El Mikrotik es configura de tal forma que el seu servidor DNS preferit apunti a la IP de la Raspberry. Quan un client estableix una connexió a la xarxa VPN, per temes de seguretat, només podrà tenir accés a internet, i qualsevol paquet que intenti mirar la xarxa interna serà bloquejat. A la següent figura es pot veure el domini que es té reservat a NO-IP.

Fig. 2. IP pública dinàmica NO-IP

B. Arquitectura de la pagina web

La pagina web seguira l'arquitectura que es mostra a la "Figura 3: Arquitectura de la pagina web" situada a l'annex. A continuació, s'explica la funcionalitat de cada component que apareix:

- localhost: Localhost es refereix al dispositiu on es desplegarà el servidor web així com els altres serveis necessaris per al funcionament de la pàgina. Cada servei té assignat un port que és el número que acompanya cada node. El port 80, el default de http serà al que escolta la pàgina de React, el port 3001 és el que escolta el servidor node i el port 3306 conte el servei de MariaDB.
- App.js: És l'arxiu d'entrada a la pàgina web, conte un router i va encaminant totes les peticions que li arribin.
- Home.js: Pàgina d'inici on s'explica el producte i com funciona.
- Login.js: Pàgina d'inici de sessió per als usuaris.
- Register.js: Pàgina de registre dels usuaris.
- User.js: Pàgina de perfil de cada usuari, allà pot veure les targetes que té, la seva subscripció i altres aspectes de la seva conta.
- Certificates.js: Aquesta pàgina mostra els certificats que té l'usuari i permet descarregar-los.
- About.js: Pàgina per mostrar informació sobre com funciona el servei i com ha estat muntat.
- Payment.js: Pàgina encarregada de guardar els pagaments que efectuí un usuari, ja sigui guardant la targeta per pagar en un futur, com per cancel·lar el pagament.
- SetupProxy.js: Aquest component s'encarrega de redirigir les peticions que utilitzin informació de la base de dades cap al port del servidor de node.
- Index.js: Es tracta de la pàgina que fa de servidor amb node i express, s'encarrega de gestionar les peticions que li arriben del front-end.
- db.js: Encarregat de crear la connexió amb la base de dades, s'exporta com un mòdul i es pot cridar des de Index.js
- encryptHandler.js: Quan un usuari es registra o fa un inici de sessió s'encripta la contrasenya a la part del servidor per millorar la seguretat.

La pàgina web funciona de tal forma que App.js encamina cadascun dels altres components a través de la url. Un "/" és el path inicial i ens porta al component Home.js. Quan es vol accedir a informació de la part del servidor, es definirà el camí al component SetupProxy.js perquè reencamini la informació cap al port corresponent. Per seguretat totes les cerques a la base de dades es fan a través del servidor amb un usuari propi i privilegis limitats. A la figura 3 es mostra el component NavLink, encarregat de crear l'index de navegació. Quan l'usuari fa clic a un element canvia el path a la url i el component Route, que es pot veure a la figura 4, carrega la nova pàgina.

```
<NavLink
  exact= "true"
  to="/user"
  activeclassname="pageSwitcherItem-active"
  className="pageSwitcherItem"
>
  User
</NavLink>
```

Fig. 3. Component NavLink React

```
<Routes>
  <Route exact= "true" path="/" element={<Home/>} />
```

Fig. 4. Component Route React

VII. Implementació

Per implementar tot aquest sistema s'ha utilitzat una única raspberry, tot i que no és recomanable fora d'un entorn de proves, ja que significa tenir el servidor DNS, la pàgina web i la base de dades tot junt en un mateix servidor. El benefici d'aquesta configuració és que els únics elements que es necessiten comprar seran els següents:

- Raspberry Pi 4 : S'encarrega del servei web i del servidor DNS. Té un preu de 40 € i funciona amb una distribució adaptada de linux anomenada Raspbian basada en Debian.
- MikroTik RB2011uias-in : Encaminador escollit per fer de servidor VPN. Té un preu de 120 € i utilitza RouterOS com a sistema operatiu.

La implementació està dividida en dos apartats diferents, el primer explica la implementació de la pàgina web, i l'altre apartat la creació d'un servidor VPN amb el ús del sistema operatiu RouterOS.

A. *Implementació de la pagina web*

Per descarregar les dependències necessàries en Raspbian s'ha utilitzat el programa d'administració de paquets de Linux APT (de les sigles en angles, Advanced Packaging Tool) que serveix per instal·lar, actualitzar o esborrar qualsevol programa o paquet del sistema. Amb la comanda "apt install" podrem descarregar el segon administrador de paquets que es necessita per la pàgina web: NPM (Node Package Manager), que ens permetrà descarregar totes les llibreries de JavaScript.

1) Esquelet del back-end amb Node :

Les primeres llibreries necessàries son Node.js i express, s'encarreguen de la part del servidor. A més, per la part del servidor es necessita la llibreria bcrypt, encarregada de l'encriptació i des-encriptació de paraules de pas.
El servidor fa servir express com a entorn de treball juntament amb un mòdul, "body-parser", que permet a l'entorn llegir les peticions POST de l'app de React a la part del front-end. Totes les peticions es faran utilitzant "app.post" (tenint en compte que app és el nostre element express()). Necessitarem escoltar les peticions d'inici de sessió, registre, consultes d'informació d'usuari, consultes d'informació dels certificats, creació de certificats i realització de pagaments. Per consultar totes aquestes dades hem d'unir el nostre node servidor amb la base de dades.

2) Creació i connexió a la base de dades :

MariaDB es un software lliure que permet crear una base de dades relacional, es pot instal·lar des de l'administrador de paquets APT i inicialitzar amb la comanda "mysql-secure-instalation" que guiarà a l'usuari durant la seva configuració. A aquesta base de dades hi tenen accés total dos usuaris: un és el propi root (o usuari sudoer) de la raspberry i l'altre usuari només té l'entrada permesa si la IP pertany a la xarxa LAN a la que està conectada. Aquest ultim s'utilitza per poder-se connectar a la base de dades en remot des d'un software amb GUI. La base de dades haurà de tenir una taula amb la informació dels usuaris: clau primària és el user-id i te com a claus úniques l'email i el telèfon. Fent servir el user-id com a clau foràNia es relaciona cada usuari amb la taula de certificats i la taula de les targetes de pagament.
Per crear la connexió entre la base de dades de MariaDB i el servidor node, s'ha d'assignar un usuari amb privilegis molt limitats definit com a "node-interact" i instal·lar la llibreria de mariaDB amb el repositori npm. A més, per evitar que intentin manipular la connexió amb la base de dades es declara l'usuari en qüestió en un arxiu específic i s'exporta a l'app principal d'express utilitzant el mètode "Object.freeze" de java que guarda l'estat d'un objecte evitant que és canviïn paràmetres o s'afegeixin de nous. Amb aquest objecte podrem fer consultes a la base de dades i inserir-ne de noves. A la figura 2 es mostra una petició POST al back-end i com es comunica amb "db", l'objecte que conté la connexió amb la base de dades.

```
app.post("/validatelogin", async (req, res) => {
    let todo = req.body;
    try{
        const busqueda = await db.conection.query("select user_id,Password,\
        IV from Users where Telefono = ? or Email = ?", [todo.user,todo.user])
        if (busqueda.length == 0 ){
            res.send({ body:"ERR_0"});
        }
        else{
            const u_id = busqueda[0].user_id;
            password = decrypt({iv: busqueda[0].IV, pwd: busqueda[0].Password});
            if (password === todo.password){
                res.send({body: u_id});
            }
        }
    } catch (err){
        throw err;
    }
});
```

Fig. 5. Exemple petició POST al back-end

3) Creació i descàrrega de certificats :

Per a la generació de certificats, fa servir la llibreria "python-shell" que serveix per executar scripts de Python des del servidor. Aquesta llibreria permet passar arguments al codi de Python i capturar el seu output. El MikroTik disposa d'un usuari amb permisos que li permet fer SSH i escriure-li comandes de forma automatitzada amb el nostre script de python. Aquest script s'encarrega de executar la comanda a traves de SSH, i guardar el secret a la base de dades. Més endavant, a la secció de RouterOS, s'expliquen les comandes utilitzades i el ús dels secrets.
L'usuari ha de descarregar una sèrie de certificats i arxius de configuració per poder fer ús el client de OpenVPN sense haver de tocar res, és per això que el servidor ha de poder crear arxius ".zip" perquè els descarregui. Per crear-los utilitzarem la llibreria "jszip", aquesta llibreria ens permet dues coses: els arxius de certificats que són sempre els mateixos ("ca.crt","client.crt", "client.key" i l'arxiu de configuració ".ovpn") es poden agafar d'una carpeta estàtica, i, d'altra banda, també permet crear els arxius dins del codi creant el ".txt" necessari per posar de forma dinàmica l'usuari i la contrasenya.

4) Sistema de subscripcions :

Per a les passarel·les de pagament s'ha utilitzat "Stripe" una API per fer pagaments que està dissenyada de manera molt senzilla, quan es crea una conta a "Stripe" es proporciona un codi personal necessari perquè l'API sàpiga a quin usuari van dirigits els pagaments. És una API que no costa res, però s'emporta un percentatge de cada pagament que s'efectuí emprant el seu sistema i té compatibilitat amb node. A la figura 6 es poden veure els diferents plans disponibles a la pàgina web.

En aquest cas, com el que es vol oferir es un servei, es fa servir un sistema de subscripció. El servidor escolta les peticions del front-end, que són les següents:

- "stripe.subscriptions.create" serveix per crear les subscripcions dels usuaris, crea un intent de pagament i, si el pagament ha estat correcte es guarda a la base de dades.
- "stripe.subscriptions.del" s'encarrega de cancel·lar les subscripcions i retorna al front-end si ha estat possible o no.
- "stripe.subscriptions.retrieve" per mirar si existeix una subscripció.
- "stripe.subscriptions.update" per actualitzar les subscripcions en cas que es vulgui canviar la informació de pagament o la targeta que utilitza.

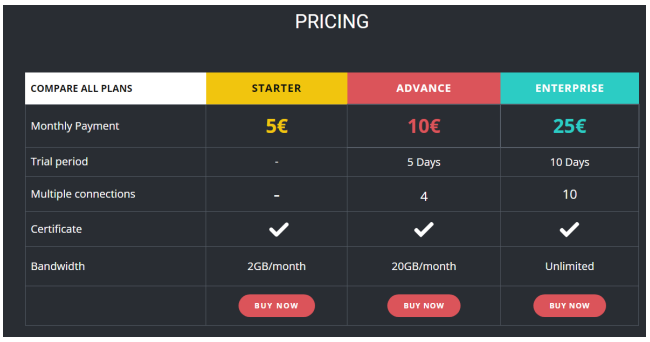

Fig. 6. Taula de plans de subcripció

5) Esquelet del front-end amb React :

Movent-nos al front-end, React ve amb una comanda integrada "npx create-react-app" que seguida del nom que es vulgui posar s'encarrega de generar l'estructura inicial de la pàgina web. El funcionament de React és senzill, cada component té les seves funcions i els seus atributs, a part d'això el mètode render() treballa amb declaracions HTML i serveix per dir-li al DOM que volem mostrar els elements de dins la funció.

Per navegar pels diferents components l'app necessita un encaminador, cada vegada que es demana una direcció nova l'encaminador escull quin component s'ha de carregar. El problema que hi ha és que si aquest encaminador o qualsevol dels components vol fer una consulta al nostre servidor apareix un error de "No 'Access-Control-Allow-Origin' header", ja que per defecte React segueix una política de "same-origin". Això, es fa per només permetre explícitament algunes connexions "cross-origin" mentre que bloqueja totes les altres. Es necessita generar un proxy middleware amb l'ajuda de la llibreria "http-proxy-middleware" per seleccionar els paths que volem redirigir al nostre servidor i permetre canviar d'origen. Les consultes d'aquests components es fan de forma asíncrona utilitzant "fetch" i utilitzant la resposta del servidor.

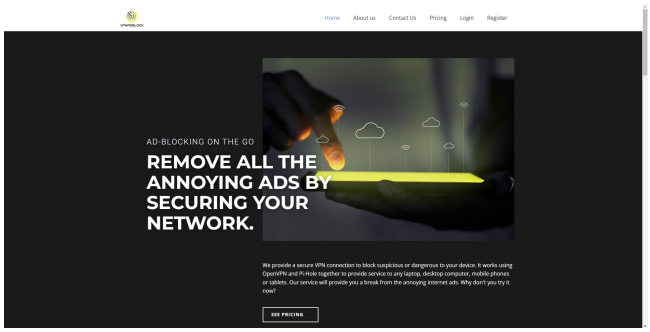

Fig. 7. Landing page de la pàgina web

B. Configurar un VPN amb RouterOS

Per treballar amb el MikroTik s'ha utilitzat Winbox, un software amb GUI dissenyat pel sistema operatiu RouterOS. El router ve per defecte amb l'usuari admin, sense contrasenya i ip "192.168.88.1". El primer que s'ha de configurar són les xarxes en les quals actua el dispositiu, una és la xarxa local que generi el router Digi (o de qualsevol altre proveidor d'internet) que funciona ja amb un dhcp propi i li assigna una adreça dinàmica i la segona és una xarxa privada creada pel MikroTik a on agafaran la ip els clients de la VPN, en aquesta s'ha de declarar la ip (10.x.x.1, perquè faci de gateway de la subxarxa)

El següent pas és configurar el servidor DNS, que serà la ip de la Raspberry i afegir la default route, per indicar que tots els paquets que no sàpiga on van és redirigeixin al router Digi. L'últim pas per poder accedir a internet és afegir una regla a la pestanya de firewall per fer un masquerade a l'interface que arribi a internet. Els interfaces del mikrotik són 10 i es tracta de les entrades Ethernet de les que disposa. A part, compta amb una pantalla que indica el tràfic de cadascun d'ells i els paquets enviats.

Per donar accés als clients, es necessita crear l'autoritat de certificació (CA), el certificat del servidor i el certificat que trametrem als clients. RouterOS disposa de la generació de certificats integrada, amb encriptacio RSA. Dels tres certificats que es creen, s'haurà de fer export al CA i al client (aquest últim genera dos arxius: un ".crt" que és el certificat i l'altre ".key" que conté la clau del certificat). Amb aquesta configuració, el mikrotik ja és capaç d'encaminar els paquets cap a internet i té els certificats necessaris per gestionar un servei VPN.

RouterOS ens permet crear diferents tipus de serveis en un mateix router: ppp, pptp, sstp, l2tp i ovpn. Per crear el servei d'VPN haurem de decidir el port que volem que escolti el Mikrotik per peticions, que per defecte sera el 1194; el certificat de servidor, que s'ha creat anteriorment; s'ha de decidir quin AH (Authentification Header) utilitza, AH és un protocol que proporciona l'autenticació de tot o part del contingut d'un datagrama mitjançant l'addició d'una capçalera que es calcula en funció dels valors del datagrama, aquest mikrotik està configurat amb SHA1 tot i que també permet MD5 o deixar-ho a null en cas que no vulguem autentificació; per últim demana el cipher que és l'algoritme d'encriptació de les dades, que utilitza sha256. A la figura 7 es mostra la configuració a través de RouterOS.
Els clients es defineixen com a secrets i tenen assignat unes plantilles anomenades profiles. Els profiles defineixen a quin servei és connectarà el secret, per exemple: si es tenen dues xarxes, es podria crear un perfil que encamines la subxarxa 10.1.x.x i un altre que encamines la 10.100.x.x i que cap d'elles es pogués connectar a l'altre. Per la part dels secrets, es necessita posar un usuari, una contrasenya, el servei que farà servir (oVPN) i l'adreça de la xarxa que agafarà en remot. Amb tot això tindríem un servidor VPN configurat totalment. Si es volgués provar manualment, s'hauria de crear un arxiu de configuració ".ovpn" per determinar el domini al qual volem accedir, el port, l'autentificació, el xifratge, els certificats i l'usuari i la contrasenya del secret que s'ha creat.

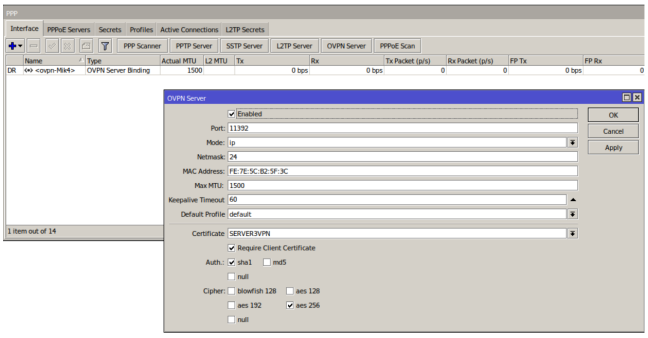

Fig. 8. Configuració oVPN server amb RouterOS

C. *Implementació de nous mètodes de pagament*

Les criptomonedes estan cada vegada més integrades a la societat, l'API utilitzada durant aquest treball per fer els pagaments, Stripe, també integra solucions per si es volgués afegir pagaments mitjançant aquestes noves tecnologies. L'opció resulta molt interessant a l'hora d'aconseguir un sistema en funcionament segur i que a la vegada proporcioni anonimat a tots els usuaris. Encara no està acabat de desenvolupar, ja que de moment només permet pagaments amb criptodivises estables, que són aquelles que segueixen el preu de les monedes reals o monedes fiat. Utilitza la blockchain de Polygon per transferir pagaments utilitzant Stipe Connect a través de l'economia de web3.

1) Web3 : La web3 és el nom que s'especula com el futur d'Internet, un nou model d'integració de l'estructura de blockchain a les pàgines web. La visió d'aquesta nova web basada en blockchain inclou criptomonedes, NFT, DAO, finances descentralitzades i molt més. Ofereix una versió de lectura/escriptura pròpia de la web, en la qual els usuaris tenen una participació financera i més control sobre les comunitats web a les quals pertanyen. No està, però, exempt de risc. Algunes empreses han entrat a l'espai només per enfrontar-se a una reacció contra l'impacte ambiental i l'especulació financera (i el potencial de frau) que comporta els projectes Web3. I tot i que la cadena de blocs s'ofereix com a solució a les preocupacions de privadesa, centralització i exclusió financera, ha creat noves versions de molts d'aquests problemes. Les empreses han de considerar tant els riscos com els beneficis abans de submergir-se.

Stripe Connect és una API enfocada a la creació de marketplaces que com a objectiu té crear un intercanvi segur entre el proveïdor i els clients. A través de la seva pàgina web pots seleccionar els mètodes de pagament que permets, s'encarrega de l'intercanvi de divises i proporciona l'opció que es vol implementar, utilitzar una criptomoneda, el USDC , per realitzar transferències de criptodivises. L'API de Stripe Connect també ofereix el mateix mètode de subscripcions que es fa servir amb Stripe en aquest treball.

VIII. Proves de validació i Resultats

Per provar el correcte funcionament de la pagina web i les seus components s'han fet diverses proves de validació. Durant la implementació dels components s'han realitzat proves de caixa blanca. Aquest tipus de proves tenen en compte els detalls procedimentals del software, i es mira el comportament d'una entrada concreta. Per exemple, quan un usuari fa un registre a la pagina web però hi ha una clau repetida el component de MariaDB genera una excepció que el servidor identifica i retorna que ja hi ha un usuari amb el e-mail o el telefon ja registrat.
També s'han realitzat proves de validació de caixa negra, aquestes proves consisteixen en simular una serie d'entrades diferents i comprovar els resultats que dona. Es diferencien de les blanques perquè en aquestes no es té en compte les decisions internes que pren el sistema, es limita a fer-ho des de el punt de vista del usuari, i només valida que les sortides siguin les correctes.
De forma general, les proves de caixa blanca s'han realitzat a la zona del back-end, on cal tenir en compte les decisions que es prenen de manera independent; les proves de caixa negra s'han realitzat al front-end, mirant que la sortida que genera el conjunt de decisions que pren el sistema sigui la correcte.
Després s'han realitzat tests d'implementació, creant diversos usuaris nous, i simulant els pagaments utilitzant funcions de la llibreria "Stripe" que permeten fer simulacions dels diferents tipus de retorns de subscripcions. Un cop tenen els certificats s'han provat les connexions VPN des de

dispositius mòbils diferents i ordinadors amb IP's públiques diferents, com la pagina web actua de forma local els certificats s'han descarregat prèviament en local i després s'han realitzat les proves. Aquestes proves consistien en mirar diverses pagines web i veure si la publicitat que es mostrava canviava activant o desactivant el VPN.

Per a comprovar el correcte funcionament del servidor DNS s'han fet peticions a 10 pàgines web diferents, una primera sense el VPN activat, i una segona amb l'VPN activat. S'han realitzat les proves en un dispositiu amb Windows 10 i utilitzant el navegador Chrome. i un dispositiu mòbil amb Android 12, fent servir també el navegador Chrome. Els resultats obtinguts en les proves fetes ha estat una disminució dels anuncis de les pàgines web significatiu sobretot en els dispositius mòbils, ja que els navegadors no permeten instal·lar les extensions dels "ad-blockers". A les pàgines web menys abusives com pot ser la web del diari Marca o la Vanguardia s'elimina el 100% dels anuncis. En canvi, a les pàgines web amb més publicitat (Cuevana, RePelis24, Softonic...) aconseguim resultats molt diversos eliminant d'un 60% a un 90% dels anuncis, aquesta diferència tan gran d'efectivitat es pot corregir identificant els anuncis i bloquejant el domini manualment dins de Pi-hole, millorant el rendiment del servei de forma progressiva. Així i tot, gràcies al bloqueig de publicitat s'assoleix navegar d'una forma molt més còmoda per aquests tipus de pàgines tan carregades d'anuncis. A més, s'ha fet la calibració manual de les pàgines que bloqueja "Pi-hole" per bloquejar nous dominis de publicitat i reduir-ne al màxim la seva aparició. En la figura 7 es mostra un gràfic mes detallat dels resultats obtinguts. D'altra banda, no s'ha obtingut el bloqueig de la publicitat que prové de jocs d'Android com promet la documentació de "Pi-hole", ni tampoc de les xarxes socials Instagram, Twitch o YouTube. Això es deu al fet que aquestes plataformes utilitzen el mateix servidor de reproducció en continu dels vídeos per enviar la publicitat a les aplicacions mòbils, si el software bloqueja la publicitat, també es prohibeix el contingut que prové tot del mateix domini.

Durant el desenvolupament d'aquest treball s'han vist certs punts que fan incompatible la idea de llençar el producte de cara al públic sense abans rectificar els problemes que s'han trobat. El primer inconvenient és que el sistema de DDNS triat ens ofereix una redirecció a un port per tenir la pàgina web sobre un servei HTTPS amb certificats TLS ja validats, però té un nombre de peticions limitat en la seva versió gratuïta, això no és un inconvenient en si, ja que si es té prou tràfic per bloquejar el nivell gratuït, els mateixos ingressos ho haurien d'amortitzar, però queda fora de l'àmbit en el qual s'ha dut a terme aquest projecte. Un altre inconvenient és que per utilitzar un servei de oVPN es necessiten tots els documents i certificats de l'usuari, que queden a la seva disposició. Tot i crear-se un arxiu zip, l'usuari no hauria de poder interactuar amb ells. Finalment, entra el factor extern del proveïdor d'Internet, si aquest veies molt de tràfic sortint d'un dels seus routers ISP, podria preguntar per què s'utilitza tanta amplada de banda o podria limitar directament l'amplada de banda del dispositiu.

Aquest treball ha tingut un rendiment correcte en un entorn no maliciós, però, que no consideraria òptim en un entorn real a raó dels punts febles explicats. Així i tot, és capaç de proporcionar de forma privada una navegació per Internet molt més fluida i còmode per a l'usuari. A més, s'han identificat els problemes que té el sistema actualment i tots tenen solució que, si bé no es podrà realitzar en aquest treball, queden detallades més endavant com a propostes de treball futur.

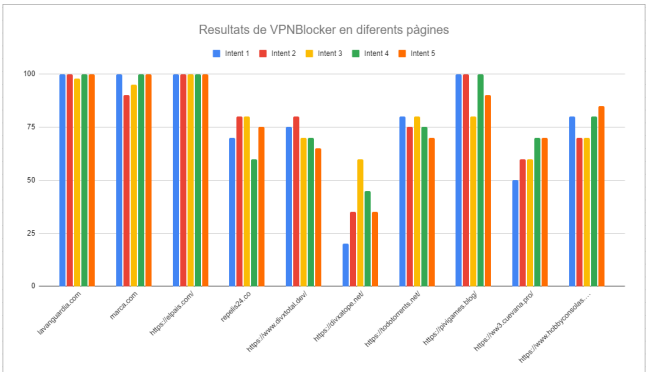

Fig. 9. Gràfic de resultats a diferents pàgines web

IX. Conclusions

Aquest treball ha consistit a desenvolupar un sistema web capaç de donar als usuaris accés a una xarxa VPN privada que funcioni juntament amb el software de Pi-hole per bloquejar tot el tràfic no desitjat.

He realitzat valoració inicial de les tres llibreries més utilitzades de front-end: React, Vue i Angular. Analitzant els seus punts forts i febles, per escollir el que millor s'adapta a l'entorn que s'ha volgut crear. He aconseguit configurar una arquitectura de pàgina web funcional, amb un sistema de front-end construït amb React i un sistema de back-end amb Node.js, que és capaç de comunicar-se a través de peticions POST i un middleware. S'ha creat una base de dades amb MariaDB i s'ha configurat per guardar tota la informació necessària dels usuaris. La base de dades també guarda la informació dels certificats que es creen en un servidor VPN local configurat amb routerOS, un sistema operatiu específic per routers.

Un cop configurats els elements principals, s'han programat les connexions entre ells i s'han afegit funcionalitats extres. Per relacionar la base de dades i el back-end s'ha utilitzat un usuari amb limitacions perquè pugui fer consultes i inserir informació a la base de dades. S'ha fet servir una llibreria a la part del servidor per encriptar les contrasenyes de forma segura. Per generar els certificats de forma automàtica s'ha emprat un codi de python que és capaç d'executar-se directament des del servidor amb la llibreria "python-shell"

de JavaScript. Amb l'API de "Stripe" s'han creat passarel·les de pagament, per crear pagaments de subscripció des del back-end de Node. Per testejar el codi, s'han fet diverses proves de validació per assegurar el comportament correcte de cadascun dels elements i proves de validació utilitzant el sistema entre un grup tancat d'usuaris.

Tot i que, com s'ha comentat a l'apartat de resultats, no és un sistema preparat per comercialitzar-se, està preparat per funcionar de forma privada, a les figures 10 i 11 es pot veure una comparació dels resultats obtinguts aplicant el sistema creat. Per això, s'han estudiat les formes que farien possible que aquest servei es tornés públic d'una forma més segura utilitzant l'estructura creada.

El primer problema es podria resoldre com a treball futur, és eliminar la interacció dels usuaris amb els certificats. Per fer-ho s'hauria d'englobar tot el procés dins d'un executable de tal forma que, un usuari amb openVPN instal·lat, pugui fer doble-clic a l'executable i que s'afegeixi l'entrada al client directament.
També s'hauria d'integrar un mètode per reduir l'amplada de banda que generen els usuaris, per fer-ho hi ha dos mètodes possibles:

- Si el que volem és mantenir l'amplada de banda, però reduir les peticions que pot fer l'usuari: els secrets del mikrotik es poden configurar per permetre una certa quantitat de tràfic al mes. Seria una opció interessant si es volguessin afegir nivells de subscripció.
- Si el que volem és ampliar l'amplada de banda: El router de MikroTik consta de 10 sortides Ethernet, si és connecten dos encaminadors de dues companyies diferents, es pot configurar el router de forma que balancegi la càrrega entre els encaminadors dels proveïdors.

En la meva opinió, s'ha seguit una metodologia correcta durant el desenvolupament del projecte, buscant els elements necessaris per reproduir l'escenari desitjat. S'han resolt problemes durant el desenvolupament, eliminant llibreries que no eren adequades, instal·lant diferents softwares de bases de dades fins a trobar un que em permetés tenir-ho tot configurat en una sola raspberry Pi i replantejant els objectius necessaris per complir l'objectiu principal.

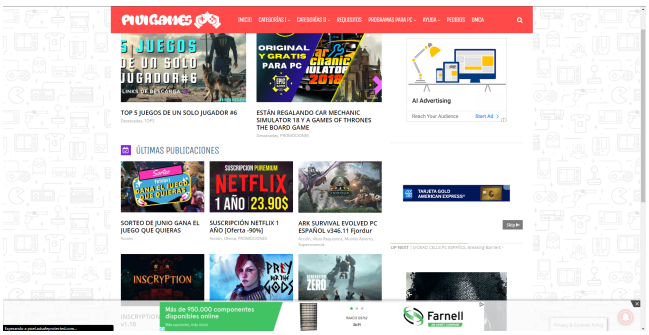

Fig. 10. Pàgina web sense ad-block

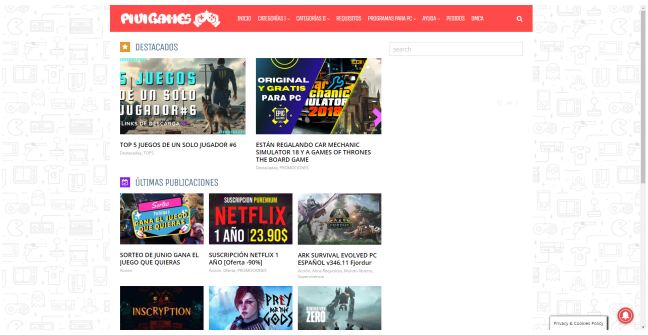

Fig. 11. Pàgina web amb ad-block

REFERENCES

[1] "Introducing JSX." React, reactjs.org/docs/introducing-jsx.html.
[2] "Babel (Transcompiler)." Wikipedia, Wikimedia Foundation, 16 May 2021, en.wikipedia.org/wiki/Babel-(transcompiler).
[3] "Cross-Site Scripting." Wikipedia, Wikimedia Foundation, 9 Jan. 2021, es.wikipedia.org/wiki/Cross-site-scripting.
[4] Yamazaki, Shiori. "Understanding Functional Components vs. Class Components in React." Twilio Blog, Twilio, 17 Feb. 2021, www.twilio.com/blog/react-choose-functional-components:text=Just like in their names,which has a render method.text=The JSX to render will be returned inside the render method.
[5] Buna, Samer. "How to Write Your First React.js Component." FreeCodeCamp.org, FreeCodeCamp.org, 13 Mar. 2020, www.freecodecamp.org/news/how-to-write-your-first-react-js-component-d728d759cabc/.
[6] "Nested Function." Wikipedia, Wikimedia Foundation, 7 Nov. 2020, en.wikipedia.org/wiki/Nested-function.
[7] Ravichandran, Adhithi, and Adhithi Ravichandran is a Software Consultant based in Kansas City. She is currently working on building apps with React. "React Lifecycle Methods - A Deep Dive." Programming with Mosh, 17 Feb. 2019, programmingwithmosh.com/javascript/react-lifecycle-methods/.
[8] "Creació i gestió de la pàgina web del club esportiu Diagonal Mar." UAB, Jun 2011, https://ddd.uab.cat/pub/trerecpro/2012/hdl2072179253/FerrerTasiesJonatanR-ETIGa2010-11.pdf.
[9] "MikroTik RB2011UiAS-IN", Mikrotik, https://mikrotik.com/product/RB2011UiAS-IN.
[10] "Gitlab documentation", GitLab, https://about.gitlab.com/is-it-any-good/

X. ANNEX

TABLE I

TAULA-1: OBJECTIUS

Objectiu	**Tipus**	**Prioritat**
Fer un estudi de les opcions de front-end	E	Essencial
Creació d'un back-end capaç d'escoltar peticions HTTP	C	Essencial
Creació d'un front-end amb sistema de login	C	Essencial
Instal·lació i creació d'una base de dades	C	Essencial
Configuració d'un servidor VPN	C	Essencial
Comunicació entre el back-end i la base de dades	A	Essencial
Sistema d'encriptació de contrasenyes	C	Essencial
Generació automatica de certificats VPN	C	Essencial
Comunicació entre el back-end i els certificats	A	Essencial
Comunicació entre el back-end i les passarel·les de pagament	A	Essencial
Validació i testeig del codi	V	Essencial
Estudi de l'integració d'altres mètodes de pagament	E	Opcional

TABLE II

TAULA-2.1:PLANIFICACIÓ TASQUES

Núm	**Tasca**
1	Estudi de les opcions de front-end
2	Creació del back-end
3	Creació del front-end
4	Creació de la base de dades
5	Creació del servei VPN
6	Adaptació i comunicació
7	Validació i testeig
8	Estudi de mètodes de pagament

TABLE III

TAULA-2.2:PLANIFICACIÓ SUBTASQUES

Num	**Subtasca**	**Tasca**
1	Instal·lació de NPM	2
2	Instal·lació dependències necessàries	2
3	Creació esquelet amb express	2
4	Escoltar peticions POST	2
5	Crear esquelet inicial	3
6	Configurar landing page	3
7	Configurar sistema enrutament	3
8	Crear vistes del portal de subscripcions	3
9	Instal·lar MariaDB	4
10	Configurar Usuaris	4
11	Configurar taules	4
12	Documentació RouterOS	5
13	Creació de certificats del servidor	5
14	Genereació de secrets	5
15	Configuració servidor VPN	5
16	Accés des de un client remot	5
17	Component de node per fer consultes	6
18	Redirecció de consultes del front-end a la BD	6
19	Encriptació de passwords	6
20	Inserció d'usuaris a la base de dades	6
21	Script de configuració de certificats	6
22	Execució de python des de el back-end	6
23	Compressió d'arxius zip del client	6
24	Descarrega d'arxius zip des de el client	6
25	Documentació passarel·les de pagament	6
26	Creació subscripcions del servei	6
27	Cancelació de subscripcions	6
28	Actualització de les subcripcions	6
29	Proves a components inidividuals	7
30	Proves de caixa blanca	7
31	Proves de caixa negre	7
32	Proves d'implementació	7
33	Buscar mètodes que ofereix el software triat	8
34	Documentació criptomonedes	8

Fig. 12. Figura 2: Diagrama de Gantt

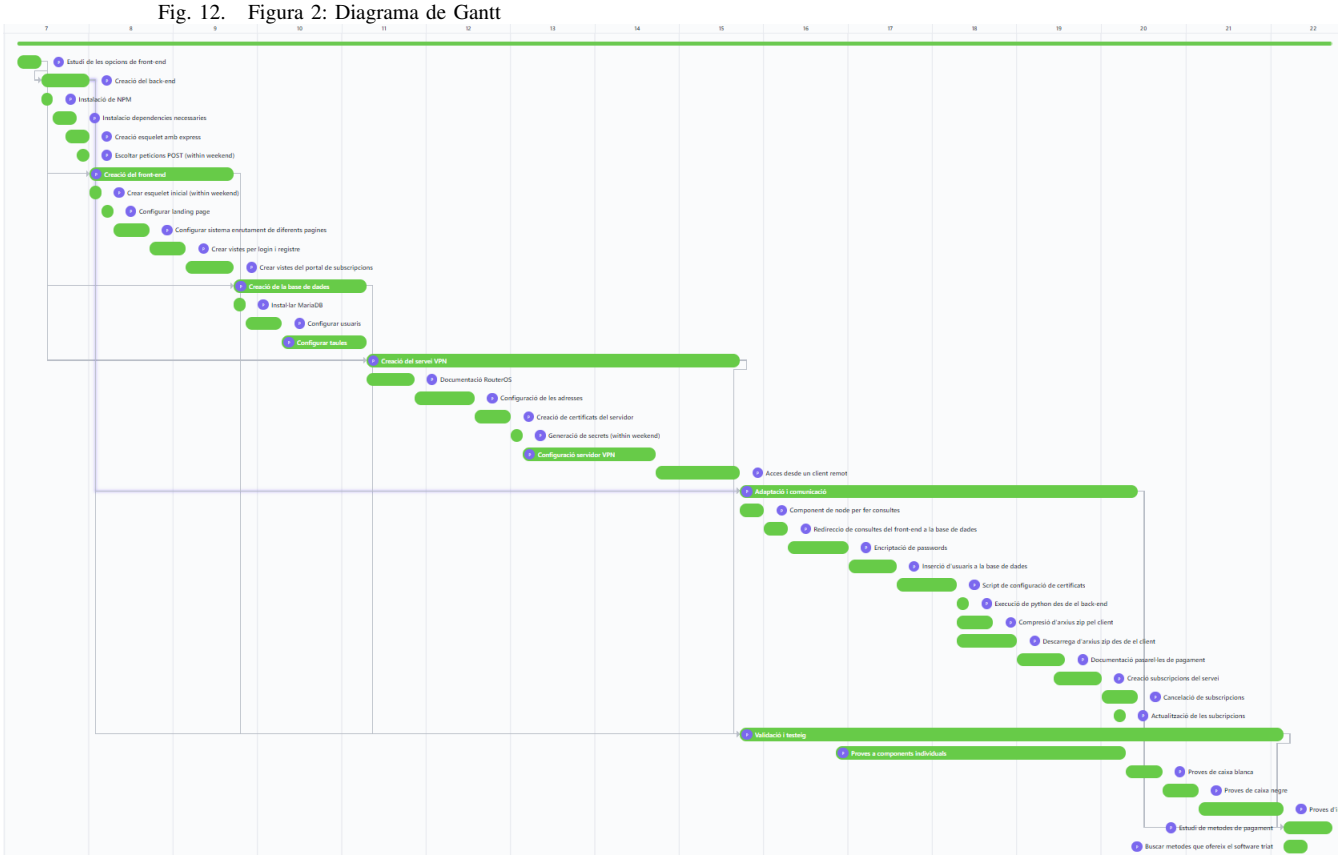

Fig. 13. Figura 3: Arquitectura del servei

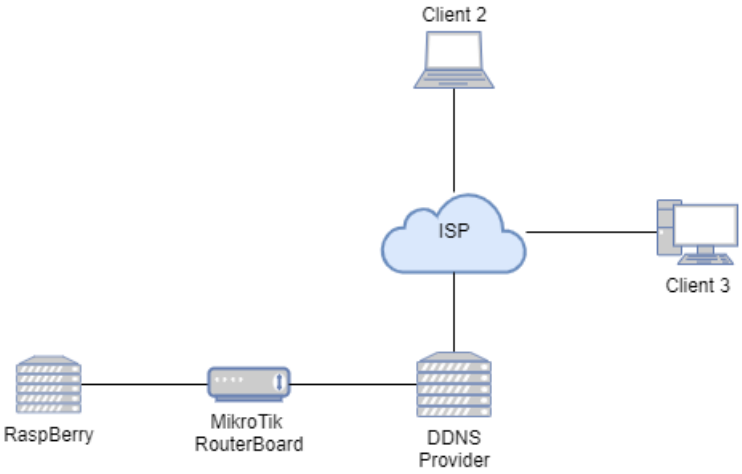

Fig. 14. Figura 4: Arquitectura de la pàgina web