Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels

UNIVERSITAT POLITÈCNICA DE CATALUNYA

TREBALL FINAL DE GRAU

TÍTOL DEL TFG: Disseny i implementació d'un servei de deteccions i d'accés a Internet mitjançant tecnologia WiFi en les platges de Vilanova i la Geltrú

TITULACIÓ: Grau en Enginyeria Telemàtica

AUTOR: Albert Anguela Calvet

DIRECTOR: José Luis De la Torre

SUPERVISOR: Lluís Casals Ibáñez

DATA: 1 de Juliol del 2016

Títol: Disseny i implementació d'un servei de deteccions i d'accés a Internet mitjançant tecnologia WiFi en les platges de Vilanova i la Geltrú

Autor: Albert Anguela Calvet

Director: José Luis De la Torre

Supervisor: Lluís Casals Ibáñez

Data: 1 de Juliol del 2016

Resum

Aquest Treball de Final de Grau es realitza en coordinació amb l'empresa ***Redman TH*** (www.redman.es), ubicada a Vilanova i la Geltrú, mitjançant un conveni de pràctiques extra-curriculars i el desenvolupament d'un projecte encarregat per un client extern.

El client en qüestió sol·licita el desenvolupament d'un complex sistema que permeti oferir dos serveis en paral·lel, ubicats en la zona turística de les platges de Vilanova i la Geltrú; són els següents:

- Detecció de dispositius que circulin per la zona en concret (associats a una adreça MAC), mitjançant la tecnologia Wifi de 2.4GHz. L'objectiu d'aquest servei és la recopilació de dades a gran escala per a la creació d'estadístiques, amb informació basada en: ubicació, temps, recurrència i concurrència, entre d'altres. Serà dotat d'una aplicació d'escriptori que permeti als administradors del sistema rebre, administrar i extreure les dades obtingudes. Aquesta part necessita la configuració i adaptació d'un servidor de base de dades (Microsoft SQL Server 2012) per l'emmagatzematge i tractament de dades.

- Servei d'accés a Internet, amb un portal captiu i l'opció de triar diferents plans d'abonament, mitjançant la mateixa tecnologia Wifi. L'objectiu d'aquest és crear un patrocini rentable per al sistema (amb col·laboracions de tercers), i recopilar informació mitjançant dades personals dels usuaris del servei. Aquesta part necessita la configuració i adaptació de diversos servidors per garantir el correcte funcionament; són els següents: Apache Tomcat7, Apache2 HTTP Server, FreeRadius i MySQL.

Els dos serveis aniran ubicats, de cara a l'usuari, sobre l'estàndard IEEE 802.11, per la qual cosa haurem de planificar i dimensionar una xarxa d'antenes que suportin aquesta tecnologia; a més, els radioenllaços adients fins a un router centralitzat, connexions des de Redman a l'entorn de producció i la capacitat del hardware d'interconnexió.

Títol: Disseny i implementació d'un servei de deteccions i d'accés a Internet mitjançant tecnologia WiFi en les platges de Vilanova i la Geltrú

Autor: Albert Anguela Calvet

Director: José Luis De la Torre

Supervisor: Lluís Casals Ibáñez

Date: July 1st del 2016

Overview

This Final Grade Work is done in coordination with the company **Redman TH** (www.redman.es), located in *Vilanova i la Geltrú*, through an extra-curricular arrangement and a development of a project commissioned by an external client.

The client requested the development of a complex system to offer two services at the same time, located in tourist beaches of Vilanova; they are as follows:

- Device mobility discovery, within the particular area (associated with a MAC address) by 2.4GHz wireless technology. The aim of this service is the collection of large-scale data to create statistical information based on: location, time and recurrence, among others. It will be equipped with a desktop application that allows system administrators to receive, manage and extract data. This part requires the configuration and adaptation of a database server (Microsoft SQL Server 2012) for the storage and processing of data.

- Internet access service with a captive portal and the option to choose different payment plans through the same wireless technology. The aim of this is to create a profitable sponsorship for the system (third-part collaboration) and collect personal information about the service users. This part requires the configuration and adjustment of multiple servers to ensure a proper behaviour: Tomcat7 Apache, Apache2 HTTP Server, FreeRadius and MySQL.

Both services will be over the IEEE 802.11 standard, so we have to plan and measure a network that supports this technology; furthermore, an appropriate radio links to a centralized router, connections from the production environment and the ability to interconnect hardware.

ÍNDEX

INTRODUCCIÓ

L'objectiu principal d'aquest treball tracta sobre el desenvolupament d'un complex sistema que permeti oferir dos serveis en paral·lel, ubicats en la zona turística de les platges de Vilanova i la Geltrú; són els següents:

- Detecció de dispositius que circulin per la zona en concret (associats a una adreça MAC), mitjançant la tecnologia Wifi de 2.4GHz. L'objectiu d'aquest servei és la recopilació de dades a gran escala per a la creació d'estadístiques, amb informació basada en: ubicació, temps, recurrència i concurrència, entre d'altres. Serà dotat d'una aplicació d'escriptori que permeti als administradors del sistema rebre, administrar i extreure les dades obtingudes. Aquesta part necessita la configuració i adaptació d'un servidor de base de dades (Microsoft SQL Server 2012) per l'emmagatzematge i tractament de dades.

- Servei d'accés a Internet, amb un portal captiu i l'opció de triar diferents plans d'abonament, mitjançant la mateixa tecnologia Wifi. L'objectiu d'aquest és crear un patrocini rentable per al sistema (amb col·laboracions de tercers), i recopilar informació mitjançant dades personals dels usuaris del servei. Aquesta part necessita la configuració i adaptació de diversos servidors per garantir el correcte funcionament; són els següents: Apache Tomcat7, Apache2 HTTP Server, FreeRadius, MySQL.

Fig. 0.1 *Ubicació de les antenes*

Els dos serveis aniran ubicats, de cara a l'usuari, sobre l'estàndard IEEE 802.11, per la qual cosa haurem de planificar i dimensionar una xarxa d'antenes que suportin aquesta tecnologia; a més, els radioenllaços adients fins a un router centralitzat, connexions des de Redman a l'entorn de producció i la capacitat del hardware d'interconnexió.

La resta del document s'organitza, a partir d'aquí, com es detalla a continuació:

En el capítol 1 veurem una planificació clara del projecte, explicant els objectius que el client requereix a l'empresa, quins són els serveis de l'aplicació i els seus detalls; tot explicat amb un llenguatge que no s'introdueix excessivament en aspectes tècnics.

Seguidament, en el capítol número 2, aprofundirem en la vessant tecnològica del projecte, es parlarà sobre l'arquitectura física a nivell de hardware, i justificarem els diferents apartats de software que s'ha utilitzat en el conjunt i les seccions del projecte.

En el capítol 3, i separat degudament en dos subapartats, s'analitzen els aspectes relacionats amb el *back-end* i el *front-end* del projecte. És la vessant més tècnica del treball, on es desenvolupen les explicacions de l'aplicació en si.

Com a capítol 4 trobem una enumeració dels problemes (més rellevants) que han anat sorgint durant els mesos de treball. A més es fa una anàlisi de com han afectat al conjunt, i una observació de la solució que s'ha aplicat en cada cas.

El treball finalitza amb un capítol destinat a les conclusions, així com les futures implementacions que aquest projecte real té programades o pensades; es conclou amb una valoració personal de la dificultat i realització del projecte.

CAPÍTOL 1. DISSENY I PLANIFICACIÓ

En el primer capítol introduirem les bases dels serveis, explicant quins són els seus objectius. En el segon subapartat mostrarem un esquema general del projecte, amb una primera planificació a grans trets del sistema.

1.1 OBJECTIUS, SERVEIS I EXPLICACIÓ

L'objectiu principal d'aquest treball tracta sobre el desenvolupament d'un complex sistema que permeti oferir dos serveis en paral·lel, ubicats en la zona turística de les platges de Vilanova i la Geltrú; són els següents:

- Detecció de dispositius que circulin per la zona en concret. L'objectiu d'aquest és la recopilació de dades a gran escala per a la creació d'estadístiques, amb informació basada en: ubicació, temps, recurrència i concurrència, entre d'altres.

- Servei d'accés a Internet, amb un portal captiu i l'opció de triar diferents plans d'abonament, mitjançant la mateixa tecnologia Wifi. L'objectiu és crear un patrocini rentable per al sistema (amb col·laboracions de tercers), i recopilar informació mitjançant dades personals dels usuaris del servei.

Fig. 1.1 *Interfície ràdio utilitzada [R11e-2HPnD]*

Com hem comentat, segons l'explicat prèviament, la solvència del sistema es basa en la informació que produeix el volum d'usuaris al recórrer tota la zona de la costa, cosa que crearà una base de dades de grans dimensions amb informació aprofitable per al sector turístic i el propi ajuntament de la ciutat.

Un cop arriba un usuari del servei a la zona de cobertura de les nostres antenes, es trobarà una xarxa oberta a la qual es podrà connectar. Quan ha estat establerta la connexió, depenent del dispositiu hi haurà dues possibilitats (un cop el terminal fa la comprovació que aquesta xarxa no dóna accés a internet); o bé el dispositiu obre el portal captiu pel seu compte, o bé l'usuari ha d'intentar realitzar una navegació per internet i així es redirigeix, amb un avís previ que s'ha d'iniciar sessió.

En aquest moment segueix el recorregut del portal: registre, únicament la primera vegada que s'accedeix; i l'elecció d'un pla de connexió, que segons les exigències del client seria una de les següents opcions:

- Visualització d'un patrocinador: 30 minuts d'accés a Internet
- Ampliació del perfil bàsic: 60 minuts d'accés a Internet
- Inserció d'un codi promocional: 60 minuts d'accés a Internet
- Resposta d'enquestes: 45 minuts d'accés a Internet
- Pagament variable: descartat finalment en la primera versió del projecte

Per la part del servidor, una aplicació de back-end, conjuntament amb un servidor que gestionarà el protocol d'autenticació i autorització, s'encarregarà d'accedir a la base de dades i realitzar les operacions CRUD (lectura, escriptura, actualització i esborrar), sobre les dades del client i l'autorització d'accés a Internet.

Pel front-end s'ha escollit la realització d'una web en comptes d'una aplicació mòbil per diversos motius. Principalment, perquè els darrers dispositius *smartphone*, al detectar una xarxa wifi sense accés a Internet, obren el navegador per defecte sense la interacció de l'usuari, ja que s'accepta el sistema de portal captiu entre els fabricants d'aquests dispositius, per la qual cosa de manera intuïtiva aquests usuaris no obriran una App.

A més d'això, podem unificar tots els dispositius utilitzant només el llenguatge web; no haver de crear un accés diferenciat per qui utilitza un portàtil o un fabricant en concret de *smartphone* ens estalviarà una gran quantitat de codi i de coneixement en la gran quantitat de llenguatges de programació que existeixen.

1.2 PLANIFICACIÓ

A continuació es mostra un esquema general del projecte, que engloba els dos serveis (detecció de dispositius/servei d'accés a Internet). Hi trobem tres blocs que són crítics i independents entre si, a l'hora d'obtenir un bon servei i les funcionalitats adients; a tenir en compte: conjunt d'antenes amb connexió, servidor físic i accés a Internet. El darrer queda subcontractat a una empresa operadora d'aquest servei (ISP), per la qual cosa garantim una velocitat de connexió de 300/30Mbps, i no hi entrarem en més detall.

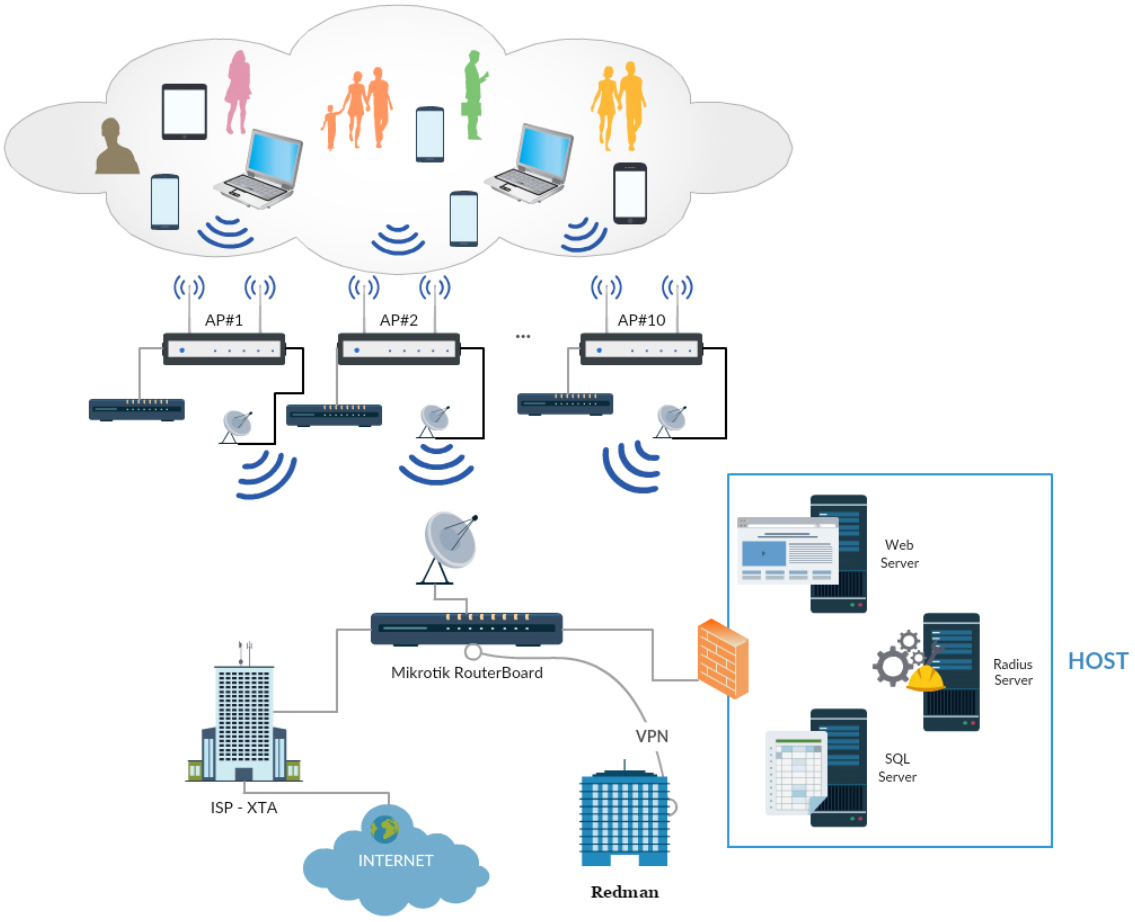


Fig. 1.2 *Esquema general del projecte*[1]

En l'escenari de la figura 1.2, s'ha hagut de planificar diversos aspectes prèviament al seu desplegament.

[1] https://creately.com/diagram/idpmo7s22/bx9SaWwi8c9uXsjCZtYAZUtVw%3D

El primer pas és el repartiment d'adreces IP en tot el sistema, respecte als dispositius i els usuaris, que s'han dissenyat com es mostra seguidament (amb màscara inclosa):

- Usuaris: 172.31.0.2-172.31.255.254/16 (65.022 adreces disponibles)
- AP: 172.25.0.2-11
- Controladora de xarxa: 172.25.0.1/24
- Host Server: 192.168.20.30/24
- Servidors (Màquines Virtuals) :
    - 192.168.20.10/24 (MS-SQL)
    - 192.168.20.20/24 (FreeRadius/MySQL)
    - 192.168.20.25/24 (Web/Tomcat)
- RaspberryPi: 172.25.0.3/24

La controladora de la xarxa, disposada per una empresa de tercers, serà configurada amb les corresponents VLANs per la interconnexió de les diferents subxarxes i el correcte funcionament del sistema. No hi entrarem en detall ja que es encarregat als treballadors de l'empresa esmentada.

Als usuaris se'ls proporcionarà una velocitat màxima de connexió de 2Mbps de baixada i 512kbps de pujada. Les velocitats de connexió però, queden declarades en les bases del servei que poden ser menors, per l'estat de la xarxa d'accés, o de la saturació d'equips, en cas de pics d'usuaris; aquests poden ser donats en la zona durant determinats intervals de temps (Festa Major, Sant Joan, dies d'Agost amb nombre anormalment alt de visitants a la platja, etc.).

S'ha de tenir en compte també, que la implementació real del projecte tindrà una durada de 3 mesos i mig en tota la zona turística, aprofitant tota la temporada en la que l'afluència de gent en aquestes zones pot suposar un interès de cara a recaptar informació. Per tant, la primera implementació és sensible a patir canvis al llarg del temps, segons el feedback rebut a partir del seu ús, ja en l'entorn real. Aquest fet és així, a més, ja que el projecte podrà ser exportat després d'aquest 'pilot' com a producte per a altres localitzacions i objectius semblants.

Per a la realització d'una part molt gran de les tasques, que a grans trets poden ser englobades en: estudi, desenvolupament, desplegament, control i gestió del projecte; necessitarem poder accedir a l'entorn de producció (lloc on es desplegarà el sistema), de forma remota i instantàniament. Per tant, serà necessària l'habilitació i configuració d'una VPN, que permetrà tenir accés directe des de la pròpia oficina de treball.

La nostre aplicació web, que comptarà amb la majoria de característiques REST (tot i no poder considerar-lo RESTful), serà programat en el llenguatge JAVA.

Per proporcionar una millor experiència a l'usuari, la nostra pàgina web serà creada amb la tècnica de disseny responsiu, això es basa en llenguatge CSS3, que permet adaptar i redistribuir els diferents components que estructuren el portal; a més, també ens estalvia el procés de crear una web diferent per a cada entorn (mida, resolució, orientació, etc.).

També ens recolzarem en els darrers estàndards estables de programació pel que fa als llenguatges de la web, d'aplicació, bases de dades, servidors, etc., ja que aquests tenen les darreres actualitzacions en termes de rendiment i seguretat, conceptes claus per a una bona definició i posada a punt del sistema.

Hi haurà certs programes d'escriptori de suport i administració, per tal de tenir un recolzament en la recollida de dades, creació d'enquestes i funcions de gestió. Seran desenvolupades mitjançant .NET amb la col·laboració d'un programador experimentat en aquest *framework*.

A més de quin software i com serà implementat, per la part de hardware s'haurà de fer un estudi de mercat sobre els productes que necessitarem, tenint en compte l'inventari del que disposen les empreses involucrades, així com els temes econòmics.

En termes mediambientals no hem realitzat un estudi exhaustiu; no es considera un apartat d'una important rellevància. Les estructures de les antenes, que poden malmetre el paisatge, estan integrades en els lavabos municipals i no causen un impacte visual apreciable. La despesa energètica tampoc suposa un impediment per al correcte desenvolupament del projecte.

CAPÍTOL 2. ARQUITECTURA

En aquest capítol descriurem de forma extensa les parts que formen l'escenari planificat en el capítol anterior; expliquem detalladament el hardware (antenes, controladors, equips utilitzats), i justificarem el software que es troba ubicat en les màquines virtuals i l'utilitzat per a desenvolupar el projecte.

2.1 HARDWARE UTILITZAT

En l'esquema de la Figura 2.1 es pot observar en detall el que hem planejat, a nivell de hardware, en el projecte; posat en forma de diagrama, a més inclou la distribució d'adreces i una petita descripció. A continuació, en aquest subapartat anirem llistant tots els equips, dispositius i terminals que han format la totalitat del projecte, les seves característiques principals i la configuració que han necessitat. Tota la documentació tècnica i específica es pot trobar en el corresponent annex.

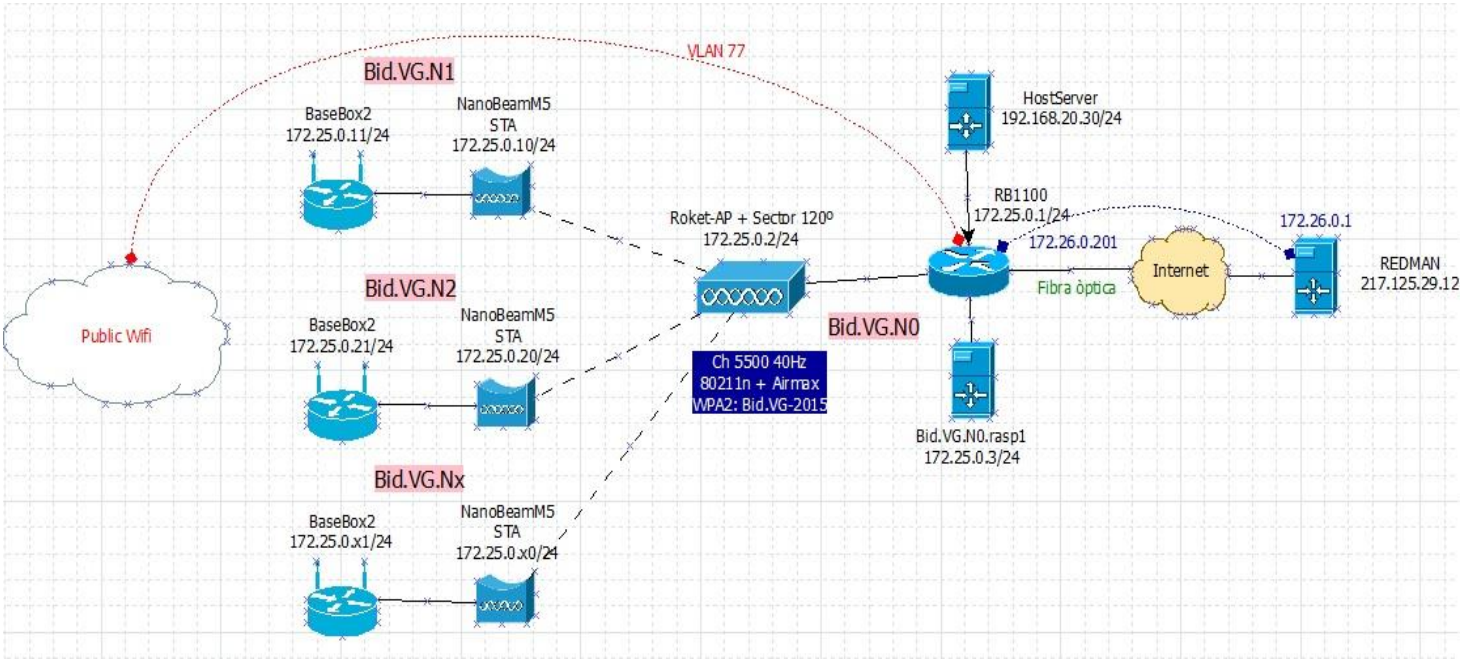


Fig. 2.1 *Esquema de capa física-xarxa del projecte*

Dels equips de la Figura 2.1, entren dintre de les competències del treball tots excepte la configuració dels punts d'accés i antenes, realitzat per una empresa col·laboradora, que a l'hora fa les funcions de ISP. Per tant, no es mostren els paràmetres de connexió d'aquests dispositius.

2.1.1 Implementació i estructura dels nodes

En aquesta secció parlem sobre el hardware dels dispositius distribuïts en els 10 punts que s'observen a la Figura 0.1.

Els aspectes tècnics d'aquest subapartat, així com els càlculs necessaris, els han realitzat els operaris que treballen per l'empresa de ISP.

Per realitzar la cobertura WiFi s'utilitzen uns dispositius anomenats comercialment *BaseBox2*, un dispositiu amb una interfície de 2.4GHz (estàndard 802.11g/n) de sèrie (*R11e-2HPnD*). El conjunt està optimitzat per a ser utilitzar en exteriors, amb protecció envers la humitat per a l'electrònica i el cablejat.

A més de la interfície per defecte, que és utilitzada pel servei de deteccions, n'hi incorporarem una segona del mateix model, configurada per donar accés a Internet als usuaris, així com per connectar amb el portal captiu.

Fig. 2.2 *BaseBox2 i un dipol utilitzats*

Cada interfície ràdio necessita dues antenes per a la transmissió de dades, el model utilitzat és un dipol de la marca Mikrotik, que opera a la banda de 2.4-2.5GHz, amb un guany de 5 ±0.3dBi, polarització lineal i una radiació de 360º en el pla horitzontal (omnidireccional).

De cara a connectar aquests AP fins la 'xarxa d'agregació', s'ha disposat d'unes antenes comercialment anomenades NanoBeam (Model: *NBE-M5-16*). Aquestes operen a la freqüència de 5GHz, cosa que evita interferències amb les d'accés a Internet, i tenen un guany de 16dBi.

L'alimentació dels punts d'accés l'hem realitzat mitjançant PoE (Power over Ethernet), depèn directament de l'ajuntament, al qual el client ha demanat els permisos adients.

Fig.2.3 *NanoBeam, utilitzades en l'enllaç vers la xarxa d'agregació*

2.1.2 Equips d'agregació

El conjunt d'aquests dispositius estarà ubicat en un punt proper on l'ISP connectarà la xarxa d'accés a Internet.

Per tant, i tal com està indicat a la *Figura 0.1* de la introducció, aquests equips estaran localitzats en la circumferència blava, que a la pràctica és una empresa del port esportiu de Vilanova i la Geltrú. D'aquesta forma, ens assegurem la visibilitat directa (o semidirecta) amb tots els nodes, a més d'una aportació de seguretat pels equips, que ve a ser la de la pròpia l'empresa.

El senyal sortirà dels dispositius finals de cadascun dels usuaris; en primera instància trobarem un *Access Point* (AP) amb una antena associada. Aquest AP es tracta d'un *Rocket M5*, operant en la banda de 5GHz (estàndard 802.11n).

L'antena que muntarem conjuntament amb l'AP es tracta d'una *AM-5G16-120* que, de fet com es pot deduir pel nom del model, té les següents especificacions: banda de 5GHz, un guany en aquesta freqüència de 16dBi, i un diagrama de radiació en el pla horitzontal de 120º, ja que aquest abasta a totes les antenes de punta a punta, i no malbarata potència de senyal cap a zones on no ha d'irradiar.

Fig. 2.4 *Rocket M5 i antena de 120º airMAX*

Un cop recollit el senyal provinent dels diversos nodes repartits per la zona de desplegament del sistema, tant del servei de deteccions com el servei d'accés a Internet, els enviarem a un router, que s'encarregarà de fer tota la gestió de la xarxa, de tenir tot el control. Entre les seves funcions trobarem:

- Gestor de la xarxa / Router
    - Assignació d'IP a tots els dispositius
    - Creació de VPNs
    - Creació de VLANs
- Hotspot
    - Control d'accés a Internet
    - Connexió lògica de nodes finals
    - Walled Garden
    - Redirecció al portal captiu
- NAS (Servidor d'accés a la xarxa)
- Enllaç vers el servidor RADIUS
- Enllaç vers la base de dades de deteccions (mode *sniffer*)

Per aquest motiu, s'ha escollit un *Mikrotik RB1100*, que conté ports de Gigabit Ethernet, a més d'una sèrie de característiques que el fan idoni per la funció dintre del projecte.

Fig. 2.5 *Mikrotik RB1100*

Aquest controlador *Mikrotik* portarà connectats, a més del cable provinent de l'AP, i la connexió cap a la sortida a Internet (WAN), una *Raspberry Pi* i un servidor, que acabaran completant el llistat de hardware.

Haurem d'afegir una sèrie de regles d'excepció al *Walled Garden* per a la funció de Hotspot. Això és, que quan un dispositiu vulgui accedir a una URL concreta o IP, no actuarà com a mediador de la connexió, sinó simplement com a encaminador de la xarxa. Aquest cas passarà, per exemple, perquè els dispositius del sistema puguin accedir als servidors que detallarem més endavant i a funcions pròpies del servei, com registrar-se mitjançant xarxes socials.

Hi connectem una *RaspberryPi 2* (Figura 2.6) amb l'objectiu de processar les dades provinents del servei de deteccions. Un col·laborador de l'empresa ha creat un *script* en Python que envia aquestes dades a la nostra base de dades. Com es veurà més endavant, aquest apartat de programació no l'ha realitzat l'autor d'aquest treball, i per tant els detalls tècnics no influeixen en el desenvolupament del projecte.

Fig. 2.6 *RaspberryPi 2*

El segon component, amb l'objectiu d'allotjar els quatre servidors que el conjunt del nostre projecte necessita, es tracta d'un servidor físic *HP ProLiant ML350 (G5)*. Hi allotgem com a 'guest' tres màquines virtuals corrent en paral·lel que es distribueixen els serveis de la forma següent:

- Windows: MS-SQL Server
- Ubuntu 14.04(1): FreeRadius Server / MySQL Server
- Ubuntu 14.04(2): Apache2 Web Server / Apache Tomcat7 Server / Reverse Proxy

Aquest servidor contindrà tot el software del treball. També l'anomenarem a partir d'ara com a 'entorn de producció', per la qual cosa els canvis realitzats hauran de ser rigorosament vigilats i testejats, ja que una fallada podria anul·lar el servei parcial o totalment, arribant inclús a malmetre informació o processos de gran importància.

És per això que tot el desenvolupament de software, en el previ desplegament inicial o actualitzacions al llarg del projecte es realitzarà en màquines virtuals situades en l'ordinador destinat a aquest objectiu; un cop provats tots els canvis de forma exhaustiva serà quan es pugin a producció, sense patir per errors o comportaments imprevistos.

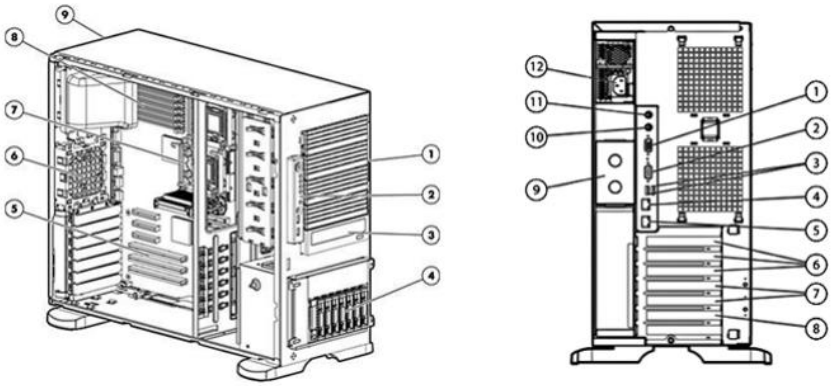


Fig. 2.7 *Servidor HP ProLiant ML350 – G5*

Tots aquests equips de la xarxa d'agregació els connectarem a un SAI, un dispositiu que proporciona energia elèctrica per un temps limitat; així aportarem resistència a fallades de la línia d'electricitat o a baixades puntuals de tensió que puguin ser donades per causes externes. A més, el servidor està configurat per arrencar de nou totes les màquines virtuals en la seva encesa, i aquestes al seu torn, tots els serveis necessaris per fer funcionar el servei sense necessitat d'interacció humana.

2.1.3 Equips de suport

Per tal de realitzar proves exhaustives (tant en l'escenari local com en producció), hem utilitzat una sèrie de dispositius varis, per cobrir un percentatge gran de mercat i testejar els aspectes tècnics, es llisten a continuació: *Samsung S3 Mini, Sony Xperia L, IPhone 5, IPhone 6, Portàtil Acer Satellite, MacBook Air, tableta IPad, tableta Samsung Galaxy Tab*; sense mencionar els múltiples dispositius de terceres persones que han aportat un feedback essencial per al projecte, i que han permès crear una compatibilitat multi-plataforma del servei.

2.1.4 Aspectes econòmics i pressupost

A continuació elaborarem un pressupost sobre l'apartat que conforma el hardware del nostre projecte, per fer-nos una idea del seu impacte econòmic.

Primer de tot hem de fer constar que les dades presentades a continuació són de caràcter aproximat, pels motius següents: hi ha material que aporten les empreses involucrades i per tant no s'ha d'adquirir, es computarà tot el material utilitzat ja que així es tindrà una visió global del seu abast; hi ha preus que poden variar segons el proveïdor en què ho adquirim, ja que els distribueix una marca i pot arribar a dependre del país, o fins i tot de la regió del punt de venta final.

Tindrem en compte dos punts claus de la instal·lació, els d'equips distribuïts als nodes finals i els de la zona d'agregació; els equips de suport no es tenen en comptes ja que són bàsicament ordinadors i *smartphones* que s'han anat prestant en moments puntuals. En el primer quadre podem obtenir un cost final d'aproximadament 2.000€, desglossats com es pot veure a continuació

Equip	Nom Comercial	Preu/Unitat (€)	Unitats	Total (€)
Access Point	*BaseBox2*	80.3	10	803
Interfície ràdio	*R11e-2HPnD*	35.2	10	352
Antena/Dipol	-	5.8	40	232
Antena Enllaç	*NBE-M5-16*	50	10	500
PoE	-	7.5	10	75
			TOTAL	**1962 €**

Fig. 2.8 *Visió econòmica sobre els nodes finals*

I pel que fa al conjunt d'aquests dispositius associats a la instal·lació d'agregació (punt intermedi abans de la xarxa troncal de la ISP), trobarem un cost teòric i aproximat de 2700€.

Equip	Nom Comercial	Preu/Unitat (€)	Unitats	Total (€)
Access Point	*Rocket M5*	80.3	1	80.3
Antena Enllaç	*AM-5G16-120*	71.2	1	71.2
Controlador/Router	*Mikrotik RB1100*	314.7	1	314.7
Processat Senyal	*RaspberryPi 2*	35.2	1	35.2
Host Server	*HP ProLiant ML350 (G5)*	2164.4	1	2164.4
			TOTAL	**2665.8 €**

Fig. 2.9 *Visió econòmica sobre els equips d'agregació*

El conjunts d'equips i material necessari tindrà un cost (com es pot comprovar en els càlculs, a l'alça), segons el mostrat en els apartats anteriors, de 4700€; tenint en compte que no s'observen els cables, eines de muntatge ni peces de suport. També hem de tenir en compte que, al ser equips orientats a empreses, el preu varia segons zona geogràfica i distribuïdor al que s'adquireixen.

Els equips obtinguts en el procés de compra es podran reutilitzar en futurs projectes, tan per part de la pròpia empresa com en la que dóna les funcions de suport.

2.2 SOFTWARE IMPLEMENTAT, SERVEI DETECCIONS

Aquest subapartat del treball està breument exposat per la seva rellevància dintre del global del projecte, tot i no hacer estat programat ni executat en cap sentit per l'autor del treball, sinó per un programador de l'empresa Redman.

Com a sistema operatiu per a la RaspberryPi, utilitzarem el conegut *Raspbian*. És una distribució de *Debian*, per tant lliure i gratuït, que ens permetrà fer córrer els *scripts* necessaris per al tractament de dades.

El llenguatge utilitzat per crear els petits programes que llegeixen les dades, com ja s'ha comentat anteriorment, és *Python*, per la seva versatilitat i potència de càlcul.

Fig. 2.10 *Logotip de framework .NET*

Al llarg del projecte hem necessitat suport mitjançant programes d'escriptori. Aquests s'han creat mitjançant el llenguatge *Visual Basic*, i el framework *.NET* de *Microsoft*. El software creat cobreix les necessitats, que a grans trets llistem a continuació, mitjançant diversos programes:

- Lectura de bases de dades, extracció d'informació
- Creació d'estadístiques segons una gran quantitat de paràmetres
- Creació i administració d'enquestes
- Creació de campanyes d'e-mail per als subscrits al servei

2.3 SOFTWARE IMPLEMENTAT, SERVEI WIFI

En aquest apartat s'explica detalladament el servei de WiFi amb accés a Internet, les parts que el composen i la seva justificació, que ens permetrà veure en global el funcionament a nivell de programari.

2.3.1 Servidor Web

Per tal que l'usuari pugui interaccionar de forma gràfica amb l'aplicació, es configurarà un servidor web (Apache2) que contindrà una sèrie de recursos visuals i de lògica de navegació/programació; l'usuari visualitzarà el contingut en el navegador del seu dispositiu.

A aquest servidor web s'accedeix quan l'usuari vulgui connectar-se sense estar autenticat pel servidor *RADIUS* (és a dir, sense disposar d'accés a Internet), amb el que haurà d'escollir un dels plans a mida per al servei de Wifi.

Un cop escollit un pla, aquest servei no interactua directament amb l'usuari fins que aquest esgoti els minuts que té disponible per navegar. Els minuts no s'aturen per molt que l'usuari es desconnecti o no navegui per Internet.

Les tecnologies i llenguatges que han estat escollits per programar la part del client (a partir d'ara *front-end*), les podem dividir en dues seccions. Per començar, la part visual l'hem desenvolupada en HTML5 i CSS3, el primer es tracta d'un llenguatge que munta l'estructura bàsica del document, i el darrer ens ajuda a modificat els estils dels elements, a més de donar dinamisme i fer responsiva la web, adaptant-se a múltiples dispositius, mides i resolucions.

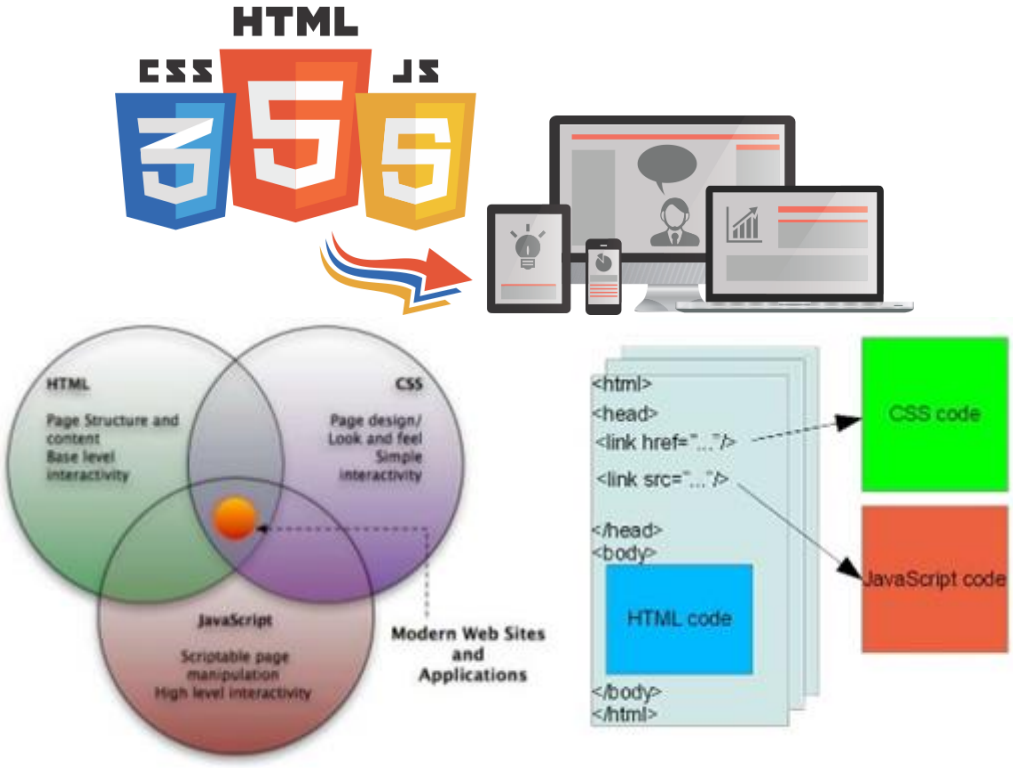


Fig. 2.11 *Tecnologies front-end: HTML5+CSS3+JS*

El segon bloc es basa en el llenguatge JavaScript; ens permetrà donar dinamisme a la pàgina observada per l'usuari, mitjançant la detecció d'esdeveniments, modificació de l'estructura creada per HTML i la part més important, la comunicació envers el servidor d'aplicació. Això vol dir que, mitjançant el conjunt de tecnologies *AJAX*, les accions de l'usuari són traduïdes en els mètodes *GET/POST/PUT/DELETE*, essent enviades i rebudes de forma asíncrona respecte a tot l'altre codi.

HTML

- Capçaleres
- Paràgrafs
- Llistes
- Imatges
- Estructura

JavaScript

- Dinamisme de continguts
- Alta Interacció
- Ginys

CSS

- Fonts
- Colors
- Vores
- Interacció simple

Falta dir que s'ha utilitzat el conegut framework *jQuery*, que permet agilitzar els treballs a realitzar, estalviant línies de codi i facilitant el procediment per a crear mètodes i funcions.

El web està suportada pels principals navegadors en el mercat, amb una quota total del **85.28%** , que suposa un percentatge adequat per al servei d'accés a Internet, tenint en compte que la major part d'usuaris tenen més d'un navegador en els seus dispositius:

- Google Chrome (42.43%)
- Safari (13.97%)
- Internet Explorer (13.17%)
- Mozilla Firefox (11.56%)
- Opera (4.15%)

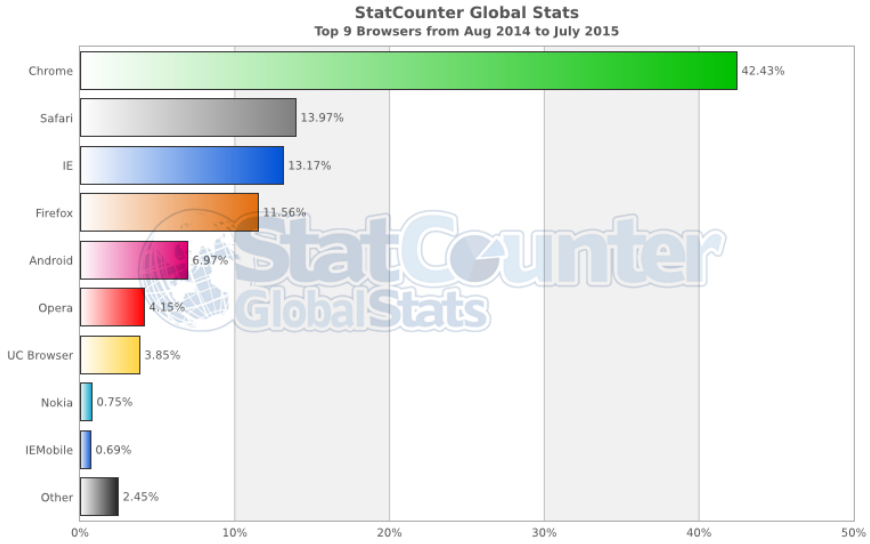


Fig. 2.12 *Quota de mercat segons navegador*[2]

La transmissió de dades entre la web i l'aplicació s'ha dissenyat per seguir el format *JSON*, que genera i parceja els objectes cap a parelles d'atributs i valors, que poden tractar tant el llenguatge JavaScript (per la part de client), com Java (per la part del servidor).

[2] http://gs.statcounter.com/

2.3.2 Servidor d'Aplicació

A nivell lògic, aquest servidor es troba entre el servidor web, justificat en l'apartat anterior, i les diferents bases de dades. S'encarrega d'allotjar l'*API* (Application Programming Interface), programada mitjançant el llenguatge *JAVA*, ens permetrà d'integrar dinàmicament la web als serveis requerits pel client.

Fig. 2.13 *Logotip d'Apache Tomcat*

Amb aquest motiu, hem escollit el servidor Apache Tomcat en la seva setena versió, que és una implementació en software d'un contenidor de *servlets*, encarregat de gestionar les peticions que li arriben dels usuaris, i executa les consultes a bases de dades pertinents. És tota la base del back-end del servei, ja que tota la lògica de programació està integrada en el mateix. Les peticions més importants en el protocol HTTP, que gestionarem en el nostre programari són les del tipus GET, POST, PUT i DELETE.

La construcció de programari i el sistema s'ha dissenyat com a RESTful, és a dir, que segueix l'arquitectura REST (Representational State Transfer), que conté entre d'altres, les següents característiques principals:

- Model de comunicació Client-Servidor.
- Stateless: el servidor no guarda informació entre diferents peticions del client, per tant no són dependents entre si.
- Els recursos individuals s'identifiquen mitjançant URIs, com a exemple de l'estil 'localhost:8080/path'
- S'utilitza hipermèdia per a l'intercanvi d'informació, en el nostre cas *JSON*

Amb aquesta finalitat, hem fet servir l'eina *Jersey* (*GlassFish Project*), que ens permet crear aquest tipus de característiques de servei amb relativa facilitat.

Per a crear i gestionar totes les dependències d'aquest projecte, que són el programari de tercers que és cridat per a realitzar les funcions necessàries, hem utilitzat l'eina *Maven* en la seva segona versió, també d'Apache Fundation.

El projecte utilitza un *archetype*, o patró d'exemple; en el nostre cas aprofitem el tipus *webapp*, que crea tots els directoris i fitxers necessaris per al seu futur desenvolupament. La gestió de dependències es porta amb el fitxer pom.xml (Project Object Model), que en un format senzill de XML ens permet afegir tots els mòduls necessaris, així com la versió a utilitzar. En la secció d'implementació es detallaran exemples d'aquesta inserció.

Fig. 2.14 *Hibernate ORM, framework per a BBDD relacionals*

Pel que fa a la comunicació d'aquest servidor amb les diferents bases de dades, utilitzarem la tecnologia que ens ofereix el framework *Hibernate*, una solució pel mapeig d'objectes-relacional per aplicacions Java. Això ens permet que, utilitzant al mínim sentències SQL, i mitjançant directament classes del nostre llenguatge, podem comunicar-nos i realitzar operacions de igual manera amb bases de dades. Únicament haurem d'indicar quin llenguatge i quin driver (*JDBC*) haurà d'utilitzar segons cada base de dades, a més òbviament, de les credencials d'accés per a cadascuna.

2.3.3 Servidor Radius

El servidor Radius s'encarregarà d'autenticar, controlar el temps de connexió i redirigir a la web els usuaris que vulguin connectar-se al nostre servei de WiFi. Coordinarem el seu funcionament amb el dispositiu *NAS* (Network Acces Server), *Mikrotik RB1100*.

El protocol *RADIUS* està inclòs dintre de la família que compleix les funcions AAA:

- Autenticació: el client prova la seva identitat respecte al servidor.
- Autorització: concessió de privilegis específics a un client, basats en la seva identitat autenticada.
- Accounting: seguiment del consum de recursos per part dels usuaris.

En aquest sentit, utilitzarem un dels softwares més coneguts i implementats, *FreeRADIUS*, que ens dóna els avantatges típics del *open-source*, com són una gran documentació i comunitat al darrere. Això facilita en gran manera la seva implementació i configuració.

L'esquema d'autenticació que utilitzarem serà el *HTTP-PAP*, que envia les dades (*username* i *password*) en clar; a més, quan un dispositiu s'associa a la xarxa, cada 60 segons envia la seva MAC per comprovar una possible validació preexistent. Això no ens aportarà greus problemes de seguretat ja que la contrasenya es tan simple com un nombre genèric, que el protocol requereix, però a la pràctica és com si no existís. En altres paraules, als nostres servidors només els importa el nom d'usuari, que serà la *MAC address* de cada dispositiu i la que permetrà l'accés al servei.

En termes d'autorització els usuaris estaran limitats segons la base de dades del *Radius* a 2Mbps de baixada i 512kbps de pujada. Aquesta configuració es crea per a un grup, i tots els usuaris s'incorporen al mateix. Permetem així que es puguin incorporar usuaris de prova sense aquestes restriccions.

2.3.4 Servidors BBDD

Fig. 2.15 *BBDD utilitzades*

Utilitzarem dos servidors diferents per contenir les dades del sistema. En ambdós farem servir el mateix paradigma, bases de dades relacionals, concretament amb el llenguatge SQL. Per la naturalesa dels nostres servidors, utilitzarem els dos sistemes més coneguts.

Pel que fa al servidor *Radius*, aprofitarem que la implementació usada (*FreeRadius*) té una integració clara en bases de dades SQL. Utilitzarem la més popular a nivell mundial, la coneguda *MySQL*. És un software lliure, *open source*, propietat d'*Oracle*, l'eina per gestionar-la es tracta de *MySQL Workbench*, que és una de les més recomanades per l'empresa. A més, de forma puntual, necessitarem accedir mitjançant consola i/o *PHPMyAdmin*.

En el servidor basat en Windows optarem per la solució propietària de Microsoft, *MS SQL Server*, que permet escalabilitat, estabilitat i seguretat. A més inclou un potent entorn gràfic d'administració.

CAPÍTOL 3. BACK-END, FRONT-END

En aquest capítol detallarem com es portarà a terme tot el que hem plantejat fins al moment, tenint en compte els objectius marcats pel client, els recursos de què disposem, i els terminis temporals per la implementació.

Primer de tot farem un repàs de la part 'invisible' a l'usuari, veient la seva estructura i les funcionalitats més importants; en la segona secció farem èmfasi en la part 'visible' per a l'usuari, els dissenys que ens han marcat i les solucions que ens hem adoptat per dur a terme les operacions que realitzarà.

S'ha de tenir en compte i mencionar que s'ha basat el sistema en un patró de disseny *MVC* (Model-View-Controller), una arquitectura que separa les dades i la lògica de negoci de l'aplicació de la interfície d'usuari. Creant una separació d'aquests conceptes aconseguim una millor estructura, una fàcil escalabilitat i una renovació/reutilització de codi més senzilla.

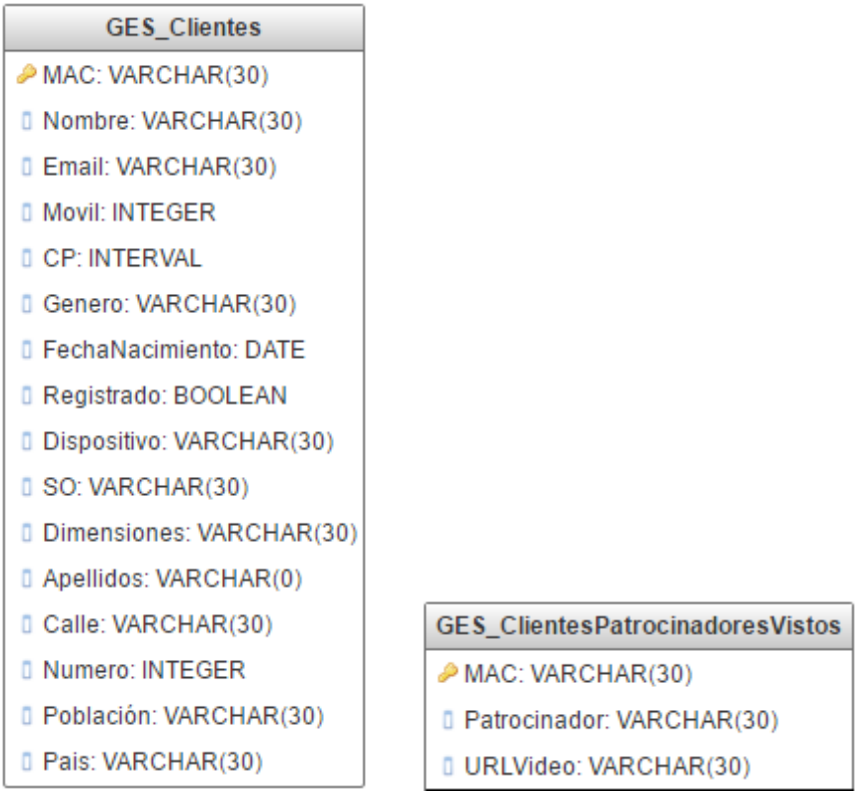
3.1 IMPLEMENTACIÓ DEL BACK-END

3.1.1 Bases de dades

Com hem vist anteriorment, hem utilitzat dues bases de dades, MySQL i MS-SQL. La primera ha estat utilitzada íntegrament per l'aplicació web, mentre que en la darrera obviarem les taules que han estat utilitzades per les aplicacions d'escriptori, ja que no han estat dissenyades per l'autor del projecte.

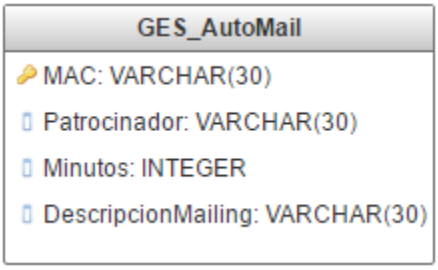
- Taula d'usuaris

Encarregada de guardar tota la informació relacionada amb els usuaris i els dispositius del sistema, és creada pel programa en *Python* gestionat a la RaspberryPi quan detecta un usuari que no ha entrat mai al sistema. Més endavant és actualitzat per l'aplicació a mesura que l'usuari va introduint dades i informació personal.



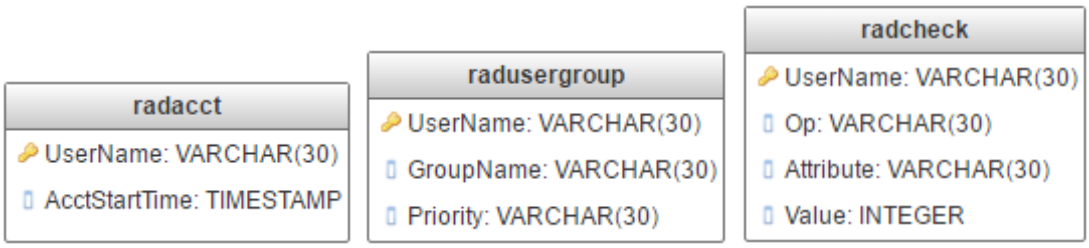
- Taules relacionades amb e-mail

Un cop omplerts els paràmetres adequats, un dels programes d'escriptori, configurat per detectar insercions, s'encarrega de crear les campanyes promocionals d'e-mail i enviar-les als destinataris. Aquest mateix programa detecta si la adreça o d'altres camps retornen un error i el comunica.



- Taules relacionades amb *Radius*

Aquestes vénen definides pel software lliure *FreeRadius*, pel que només detallem els camps que hi introduïm en l'aplicació.



La resta de taules són utilitzades internament pel software i no es modifiquen.

- Taules relacionades amb codis

Són aquelles que contenen informació relacionada amb els diferents codis promocionals existents.

- Taules relaciones amb enquestes

Aquí trobem tant les taules encarregades de gestionar les preguntes com les respostes del sistema, en tots els formats possibles. Té un grau de dificultat afegit en el seu plantejament ja que la relació entre elles depèn del tipus de pregunta.

3.1.2 Interfície de comunicació

La interfície en Java és una col·lecció de mètodes abstractes i algunes propietats. No es descriu la seva implementació, que dependrà de cada o projecte o classe que la cridi i la utilitzi; per altra banda s'especifica el *què* s'ha de fer.

Tot i que en la nostra aplicació només utilitzarem una implementació de la interfície, en un projecte de caràcter empresarial ens hem vist obligat a seguir aquest esquema d'interfícies.

Això ens permetrà formar un esquelet per a futures implementacions del projecte, que és l'objectiu inicial. A més ofereix un nivell d'organització extra, que requerim per a un bon desenvolupament.

A continuació veurem tots els mètodes que conté la nostre interfície i la funció que desenvoluparan a la pràctica. Utilitza la paraula reservada `interface`, tot escollint noms el més clarificadors possible per a una millor organització:

```
public interface BidfonInterface
```

Funcions relacionades amb els usuaris (taula *Usuarios*):

- `registroUser` : Aquest mètode no s'usa en la implementació del sistema, però va ser creat en la fase de desenvolupament per poder fer proves amb usuaris ficticis sense haver d'introduir-los manualment a la base de dades.

  Es passa com a paràmetre un *Usuario* i es retorna un `String` amb la resposta a la petició.

- `getUser` : El mètode implementa una funció que retorna un objecte tipus *Usuario*, referent a la seva MAC, paràmetre que se li passa.

- `getNumRegist` : Retorna un enter amb el nombre d'usuaris registrats al sistema. Es va implementar en una fase avançada del projecte, un cop el client demanava més funcionalitats en temes estadístics i vam veure que necessitàvem obtenir aquesta informació. No es passa cap variable com a paràmetre.

- `updateUser` : Aquest mètode es fa servir quan l'usuari entra per primer cop al sistema. Es passa un *Usuario* amb els camps que actualitzem en aquest cas, i el booleà referent al registre. Retorna un `String`.

- updatePerfUser : Semblant al mètode anterior, però en aquest cas s'actualitzen a la base de dades uns camps diferents, referents a quan l'usuari amplia el seu perfil per a realitzar un pla de connexió.

- listUsers : Aquesta funció té la seva utilitat per a tasques de manteniment i administració del sistema, ja que no es veu reflectida en el front-end de l'usuari. Retorna un tipus de variable List d'objectes *Usuario*, i algunes de les seves propietats. No es passa cap variable com a paràmetre.

- checkPatro : El mètode realitza un *get* per tal de reconèixer en una taula separada quin és el patrocinador associat en aquell moment a l'usuari, passant com a paràmetre la MAC del mateix.

- updatePatro : Aquest mètode fa una crida a *checkPatro*, i actualitza el camp amb el següent d'una llista predeterminada (en futura versió es canviarà per una llista dinàmica en una BBDD). En cas que l'usuari no en té cap patrocinador, crida a una funció que introdueix el primer patrocinador de la llista; motiu pel qual el mètode *updateUser* fa una petició a aquest.

Funcions relacionades amb el e-mail:

- addMail : Aquesta funció introdueix a la base de dades una entrada amb la qual queda registrada una campanya de mailing . Al seu torn, el programa encarregat de la campanya de mailing, al detectar aquesta entrada enviarà un correu al e-mail de l'usuari. S'envia com a paràmetre una classe del tipus *Mail*, retorna un String amb l'estatus de la resposta.

Funcions relacionades amb el servidor Radius:

- addRadius : El mètode introdueix diversos registres i/o actualitzacions a les bases de dades del servidor *Radius*, que s'encarrega de gestionar l'autenticació, autorització i comptabilitat de les sessions. Passem com a paràmetre la MAC de l'usuari i el temps de connexió que se li permet. Restringeix els temps de sessió que no concordin amb cap pla preestablert.

Funcions relacionades amb la taula de codis:

- addCodigo : Aquesta funció s'encarrega d'introduir codis a la base de dades, en funció de 1) nombre de codis que es volen introduir, 2) inici del codi per fer distinció segons el patrocinador, 3) longitud de la resta del codi amb una generació aleatòria. Aquesta funció és administrativa i els usuaris no hi poden accedir des de la seva visual.

- checkCodigo : Comprova el codi contrastant amb tota la taula de codis. Actualitza l'objecte del tipus *Codigo*, introduint la MAC i el datetime en el qual ha estat gastat. A més, si es tracta d'un codi del client, restringeix als usuaris la possibilitat de tornar-lo a gastar. El mètode pot respondre segons si no existeix el codi, ja ha estat gastat o bé si és d'un sol us i ja no pot reutilitzar-lo.

- listCodigos : aquest mètode, tampoc disponible per als usuaris, respon amb un tipus List de tots els codis disponibles a la taula, i si han estat utilitzats.

Funcions relacionades amb la taula d'enquestes/respostes:

- getQuest : Retorna un List del tipus *Encuesta*, en funció de la MAC que es passa per paràmetre. Té en compte les respostes que l'usuari ha contestat prèviament per a tornar l'enquesta que li correspon en el moment de la petició. A més, també passem per paràmetre el número d'enquestes que hem de rebre per donar dinamisme i futur al mètode.

- addAnswer : Aquest mètode rep i processa una llista d'objectes tipus *EncuestaRespuestas*, i les introdueix una a una a la base de dades. Actualitza l'enquesta en el qual l'usuari es troba.

Un cop creada i determinada la interfície, haurem de crear les classes (una amb el nostre objectiu) per a la implementació de la mateixa. Utilitzarem la paraula reservada implements, que lligada amb l'esmentada anteriorment; crearà tota l'estructura buida, que programarem per a que faci les funcions marcades segons els criteris.

```
public class OperacionesBBDD implements BidfonInterface
```

A aquesta interfície i implementació, com podem veure, hi ha certs aspectes d'administrador del sistema. En canvi, no quedaran reflectits en la interfície d'usuari (*front-end*), ja que no estava contemplat en aquesta versió del projecte; cosa que si passa en futures implementacions del mateix.

Més endavant, en els següents apartats, farem una explicació de les funcions amb unes característiques més interessants en aquest treball.

3.1.3 Interacció amb RADIUS

A continuació detallarem la integració del servei, protocol i servidor que s'encarreguen de controlar l'accés a Internet, conjuntament amb el nostre *Network Access Server*, estem parlant com ja hem comentat, del *RADIUS*.

El *NAS*, o servidor d'accés a la xarxa, és el client del servei, així serà ell qui indiqui als dispositius si poden accedir a la xarxa, i en cas afirmatiu, sota quines condicions.

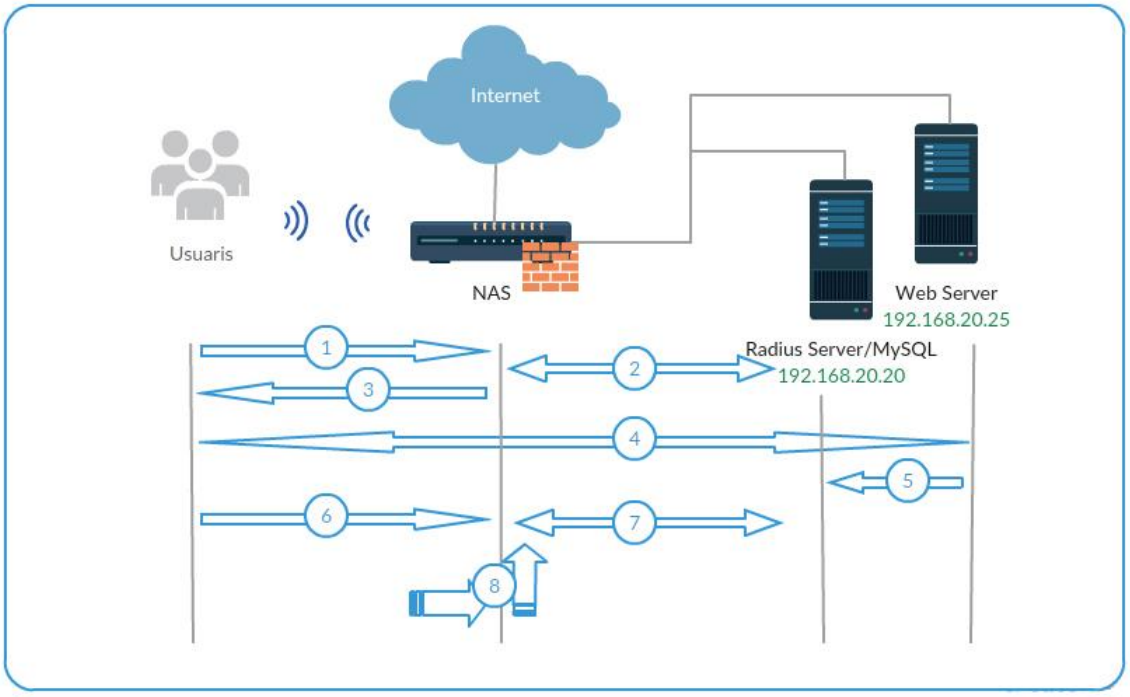


Fig. 3.1 *Esquema sobre la comunicació NAS/RADIUS*

Com podeu veure en la Figura 3.1, existeix una comunicació constant entre usuaris del sistema, el *NAS* i el servidor *RADIUS*. A continuació detallem aquesta comunicació sense entrar en nivell tècnics, ja que els missatges vénen detallats per estàndards dels protocols que interactuen a les diferents capes.

1) L'usuari, mitjançant el seu dispositiu, es connecta envers el *NAS*; fa una petició per mirar d'accedir a Internet.

2) El *NAS*, com a client del protocol *RADIUS*, fa una petició al servidor del mateix. En aquesta petició demana si el dispositiu (basat en la MAC), està autoritzat per accedir a la xarxa externa, i posant el cas que no està validat, el radius pot o bé tornar un missatge d'error d'autenticació o bé simplement no contestar.

3) El dispositiu d'accés a la xarxa denega el servei al dispositiu, i a partir d'ara reenviarà tots els paquets (amb excepcions configurades) amb la seva MAC cap a la IP indicada, que en aquest cas és la del nostre servidor web.
4) El client interactua amb el servidor web. Així aquest canal de comunicació s'estableix segons el que la comunicació requereixi, sense límit de temps o de velocitat de transferència, tot i poder-se modificar aquests paràmetres.

5) Quan el client arriba al final d'un dels mètodes pels quals es pot obtenir el servei, s'executa una petició cap al servidor *RADIUS*, que es basa en actualitzar la seva base de dades. Llavors s'autoritza el client, se li dóna una sessió i uns paràmetres. En aquest instant el dispositiu encara no comença a 'gastar' el temps disponible.

6) El client web força el dispositiu en concret a realitzar una petició de nou contra el *NAS*, creant així la simulació que el client torna a provar de connectar-se a Internet, i obligant al servidor d'accés a realitzar

7) Una altra petició envers el servidor d'autenticació, autorització i accounting. En aquest cas, respon en cas afirmatiu tot indicant la sessió que llegeix de la base de dades, és a dir, amb tots els paràmetres referents al dispositiu (velocitat, temps de sessió, etc.).

8) És en aquest moment quan el client disposa del temps establert per connectar-se a la xarxa externa. En el nostre cas, el comptador no para de baixar encara que el client es desconnecti de la xarxa. Ara el *NAS* reenvia tots els paquets del client segons la seva destinació sense fer-ne distincions. Un cop s'hagi acabat el temps de què disposa, es tornarà al pas 1.

Aquest servei, ubicat en una de les màquines virtuals, el controlem en la seva totalitat amb la funció addRadius. Conté un algorisme creat per tal de comprovar la preexistència del dispositiu i d'altres paràmetres, a més de retornar estatus específics segons possibles errors; deixa omplertes totes les taules necessàries del servidor *Radius*.

Està dissenyat de forma que les taules siguin un registre (o log) de tota l'activitat per part dels dispositius, i poder treure estadístiques sobre el sistema. Això fa que s'hagi d'anar amb cura a l'hora de tractar la base de dades i taules que inclouen.

3.2 IMPLEMENTACIÓ DEL FRONT-END

Aquesta secció serà un recull de les tecnologies i recursos que s'ha utilitzat per crear la interfície web, o *front-end*, que l'usuari veurà en qualsevol dispositiu. El disseny del sistema ve marcat pel client, per la qual cosa no hi hem intervingut des de l'empresa, i conseqüentment no hi entrem en detall.

Com ja hem comentat anteriorment, la nostra web tindrà un caràcter responsiu i adaptació a esdeveniments. Aquests punts impliquen que la interfície la integren diverses tecnologies. El llenguatge HTML (.html) ens donarà l'estructura de la pàgina 'bàsica', la tecnologia de JavaScript (.js) per crear dinamisme a la web, i les fulles d'estils referents per a cada plana, els CSS (.css).

3.2.1 Dinamisme

Per fer-nos una idea del dinamisme que ens ofereix JavaScript, en el nostre cas mitjançant la utilització del framework jQuery, trobem en el recorregut diverses pantalles que l'apliquen. La imatge de la Figura 3.2, corresponent a l'arxiu *video.html*, n'és un exemple clar.

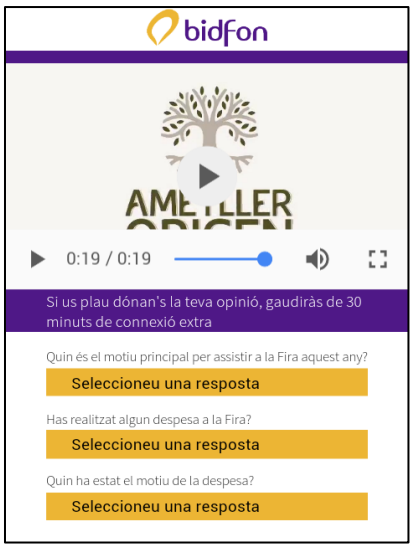


***Fig. 3.2** Pantalla final 'vídeo' de la web*

La plana és una estructura *HTML* senzilla: una imatge de logo, un fons que inclou colors i text i dos *div*; una inclou una etiqueta *<video>* que no assenyala la font d'on prové l'arxiu, i una altra que s'omplirà amb les preguntes que li pertoquin a l'usuari en qüestió.

Entra en aquest punt el dinamisme. Seguint dues crides *AJAX*, es recull per una banda el patrocinador que pertoca a l'usuari i per l'altra les preguntes i respectives respostes que haurà de contestar. En cas que no li quedin preguntes per fer tan sols apareixerà el vídeo.

Les dades es circulen en funcions de jQuery cap a l'estructura de la web, que les mostra a l'usuari; a continuació veurem els dos mètodes en el qual oferim dinamisme en aquesta pantalla.

```
$("#video").attr("src", "/vdo/"+patro+".mp4");
```

Modificació de l'atribut *src* en el vídeo, passant un paràmetre que ve retornat per la funció explicada en el back-end '`checkPatro`'. Com es pot comprovar, els vídeos s'han de guardar amb el mateix nom que el patrocinador corresponent i en el format *mp4*.

```
$( ".questBloc .pregunt" ).append( $( "<p>"+this.pregunta+"</p><select id="+idSelect+
"><option select='selected' disabled>Seleccioneu una resposta</option>" ));

$(this.respostes).each(function(index2){
    $( "#"+idSelect).append( $( "<option value='"+this.id+"'>"
    +this.descripcionOpcion+"</option>" ) );
});
$( ".questBloc .pregunt" ).append( $("</select>"));
```

Les preguntes s'insereixen al cos de la pàgina, concretament a l'element amb classe *pregunt*, mitjançant la funció *append* de *jQuery*, passant tots els paràmetres retornats per la funció ja explicada 'getQuest'. S'executa el nombre de vegades indicades en una variable que ens permet gestionar-ho. Inclou, a més, totes les respostes que les preguntes tenen associades. El seu conjunt s'estructura en un *select box* amigable per l'usuari.

3.2.2 Caràcter responsiu

Aquestes característiques es componen per dos aspectes principals. El primer és la utilització de '*media quèries*' en CSS. A continuació veiem una sentència en la que el codi només serà executat per dispositius més grans de 760 píxels i en posició apaïsada.

```
@media screen and (min-width: 760px) and (orientation: landscape){
/* Codi executat en certs dispositius */
}
```

En el nostre sistema hi haurà una diferència i adaptació respecte als dispositius mòbils, tabletes en posició horitzontal, tabletes en posició vertical i ordinadors de major resolució. Això a més crea uns marges d'execució de codi importants, com mencionem més avall.

Les unitats que utilitzem seran en format percentual en la mesura del possible. Es poden aplicar percentatges de formes diferents, nosaltres concretament utilitzarem *vh* i *vw* (viewportheight / viewportwidth). Això significa que dintre dels intervals que definim en les *media quèries*, el diversos elements s'aniran adaptant segons la mida de la pantalla, o en el nostre cas, del *viewport*. Això és la mesura que resta en el navegador un cop es descarta les possibles barres que conté.

En un exemple pràctic; donem a un objecte una amplada de *80vw*, això comporta que la unitat final serà del 80% del que mesuri el viewport en píxels del dispositiu. Si mesura 200px, l'objecte tindrà una amplada de 160px.

A més, hem de mencionar que dissenyarem el projecte utilitzant una filosofia de planejaments anomenada *mobile-first* (o *mobile-friendly*). A la pràctica tracta de pensar el conjunt per tal que el primer codi que es llegeix sigui l'executat pels mòbils i els altres dissenys els 'llegeixin' únicament els dispositius que els pertoqui. Permetem l'estalvi màxim de lectura i execució de codi per part dels dispositius mòbils, que generalment tenen menys prestacions i rendiment que la resta; aconseguirem una perspectiva millor de cara a l'usuari i a la seva experiència.

3.2.3 Funcions generals de l'usuari

En aquest subapartat volem remarcar certes operacions que els usuaris poden realitzar des de l'aparença visual del nostre projecte. Així mateix anirem indicant recursos de programació i remarcant la visual a què corresponen.

***Fig. 3.3** Pantalla d'e-commerce*

Moltes d'aquestes es poden recollir a la plana que anomenem 'e-commerce', que conté la publicitat i recopila els diferents mètodes d'accedir al servei.

- Registre / Ampliació de perfil

Al accedir per primera vegada, els usuaris han de completar un formulari amb algunes dades personals. Aquest sistema es pot fer manualment o mitjançant una xarxa social, com explicarem en el seu apartat corresponent. Quan s'amplia el perfil només hi ha la solució manual.

Aquestes pantalles realitzaran una petició *AJAX* quan l'usuari hagi completat tots els camps, i si li manca algun se li indicarà segons convingui. El dispositiu de l'usuari haurà estat introduït a la base de dades per el *script* que gestiona la nostra

RaspberryPi. La petició, per tant, es tracta del mètode *PUT*, ja que haurem d'actualitzar diversos camps de la taula *GES_Clientes*, i no pas crear-ne una entrada nova.

El més significatiu d'aquesta pantalla és l'algoritme que utilitzem per reconèixer si algun camp manca de ser omplert. La comprovació es fa quan l'usuari omple el darrer *input*, que correspon al de *Codi Postal*. Detectem amb un mètode (que explicarem en la següent funció, de codi) que s'ha introduït 5 dígits i realitzem la comprovació. Si falta el paràmetre, es marca la vora exterior en vermell, en cas contrari s'amaga la línia.

```
$('input').each(function(){
    if ($(this).val()==="") {
        $(this).css("border","1.5px solid red");
    }
    else{
        $(this).css("border-style","hidden");
    }
});
```

- Introducció de codi promocional

Quan l'usuari vol gastar el seu codi promocional, clica a l'input de la pàgina mostrada i es disposa a escriure'l. Amb el següent mètode, que conte la funció *keyup* de *jQuery*, aconseguim detectar la 6 xifra, que fa saltar la petició envers el nostre servidor.

```
$(document).on('keyup', '#code', function(){
    var cod = $('#code').val();
    if (cod.length === 6){  // codis de 6 xifres
        $(this).blur();     //esborrem el valor
        checkCodi(urlCode+'check/'+cod+"/"+mac);
    }
});
```

- Respostes a enquestes

Tant en els vídeos promocionals com en l'apartat d'enquestes haurem de contestar un seguit de preguntes que ens permeten ampliar informació sobre els usuaris. A continuació podem veure l'algoritme utilitzat per saber si l'usuari ha contestat totes les preguntes que li pertoquen, fet que es comprova cada cop que l'usuari en respon una (recordem, mitjançant un *select box*).

```
$(document).on('change', 'select', function(){
    var found = false;
    $('select').each(function(){
        if($(this).find('option:selected').text()==='Seleccioneu una resposta'){
            found = true;
            return false;
        }
    });
    if (!found){
        funcioEnviar();
    }
});
```

- Visualització de vídeo patrocinat

Aquesta funció ja ha quedat comentada quan exemplificàvem el dinamisme de la web en el seu apartat corresponent. Per tant, es dóna com a resolta i explicada.

- Accés a Internet

Al final de tots els punts anteriors en els quals es pot accedir a navegar per Internet, es fa una crida a la funció que introduirà el temps concretat al *Radius*.

S'especifica mitjançant els paràmetres corresponents en aquesta funció si l'usuari haurà de rebre un e-mail o no (i el tipus de campanya en cas afirmatiu), així com si conté un codi promocional (o un de concurs); segons cada cas particular.

3.2.4 Login mitjançant xarxa social

El nostre servei permetrà realitzar 'login' amb una xarxa social, com és *Facebook*, per tal de facilitar el procés de registre de cara als usuaris. Això al seu torn, repercutirà en una major incorporació al sistema, una major difusió i una millor rendibilitat.

Mitjançant un botó de login, i demanant els permisos adients a l'usuari, podem recollir les seves dades; fent així que no s'hagi de completar manualment els diversos camps. Entre les dades que podem recollir de Facebook es troben: nom, cognoms, e-mail, data de naixement, gènere i codi postal.

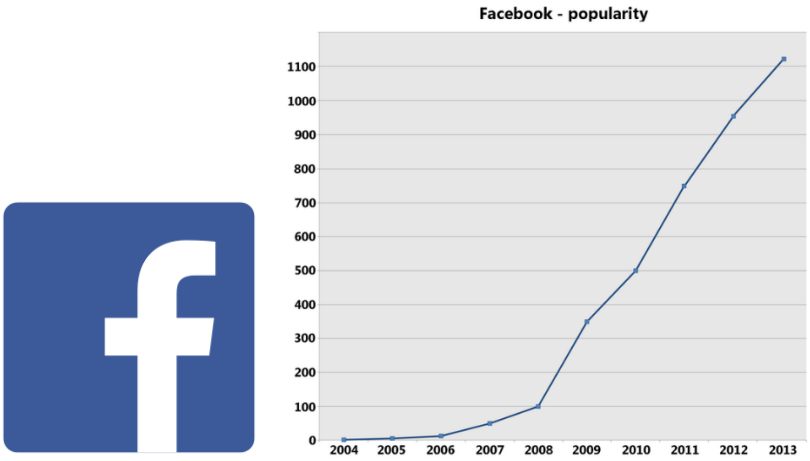


Fig. 3.5 *Logotip i popularitat de Facebook, en milions de persones*

Facebook va ser creat el 2004 per Mark Zuckerberg i és la major xarxa social a nivell mundial. Els beneficis principals de la plataforma escollida són el seu nombre d'usuaris, arribant als 1.350 milions de persones l'octubre de 2014. Això fa que un percentatge molt gran de dispositius del sistema puguin optar a utilitzar els beneficis d'entrar mitjançant aquesta opció.

Pel que fa a la implicació en seguretat, donat que els únics permisos que autoritzem és la recollida de dades bàsiques (sense possibilitat de publicar al mur, veure les amistats; així com dades bancàries, etc.), no és un aspecte transcendental en aquest treball.

```
$.getScript('/js/Fsdk.js', function(){
    FB.init({
        appId: '[redacted]',
        status: true,
        cookie: true,
        xfbml: true,
      version: 'v2.2'
  });
});
{scope: 'public_profile,email'}
```

Fig. 3.6 *Petició local del SDK, permisos concedits i login via Facebook*

El primer que haurem de fer per desenvolupar aquest mètode, i poder utilitzar la API de Facebook, és crear una aplicació en el panell de desenvolupadors de Facebook (https://developers.facebook.com/apps/). Un cop configurada correctament i inicialitzada per al seu accés públic, podem implementar-la en la nostra web. Veure detalls de com crear l'aplicació, aconseguir l'identificador i procés complert en l'annex d'aquest treball.

Haurem de realitzar una crida al SDK de la xarxa social. Això es pot fer mitjançant un *CDN* (connect.facebook.net/es_ES /all.js), això és un *Content Delivery Network* que et permet rebre en un servidor proper el software de manera ràpida i comprimida; o bé descarregant-lo i fent-ho en local, com ha estat el nostre cas. Un cop inicialitzada, cridem a la funció de *login* de la API, tot concedint els permisos que necessitem dintre del camp *scope*, Figura 3.6.

A partir d'aquest pas, només hem de rebre els diferents estats segons l'acció de l'usuari. Si està correctament autenticat a *Facebook* i dóna permisos a l'aplicació, rebem les seves dades sense més dilació.

Després, parcegem els camps on toca de la pantalla de registre, i instem a l'usuari a emplenar els que resten buits.

3.2.5 Utilització de plugins

Hem trobat la necessitat de crear certs 'recursos' visuals, que després d'un anàlisi primari s'ha vist que portaven una càrrega massa alta de feina. Això ho hem solucionat mitjançant la utilització de pedaços de codi que es poden descarregar directament d'Internet, a mode de programari lliure, gratuït i obert.

Òbviament no només hem hagut de realitzar la correcta importació d'aquests codis (que vénen a ser fitxers CSS i JS), sinó que s'ha procedit a adaptar-los segons les necessitats del sistema, havent de provar a vegades més d'una solució fins arribar a l'idoni.

Dos exemples que han portat una càrrega important de feina els trobem en els següents recursos:

- Slider horitzontal: en certs casos requeríem que diverses pantalles transcorreguessin de forma horitzontal, donant la sensació que passaves d'una a l'altra sense interacció de l'usuari. Les dificultats van sortir quan es volien omplir aquestes estructures amb més que una simple imatge, és a dir amb components totalment actius com botons, inputs...
  Les pantalles que l'utilitzen són: home (inicial), la d'ampliació de perfil i la de respondre enquestes.

Fig. 3.7 *Transició d'un slider en acció*

- Menú desplegable: les pantalles de la nostra web necessitaven un menú del tipus 'hamburguesa' que sortís d'un lateral per proporcionar informació sobre les bases del sistema, els termes legals, la informació de contacte... i que a més respectés el disseny responsiu.
  Les pantalles que l'utilitzen són: registre d'usuari i les que porten als diversos plans de connexió explicats prèviament.

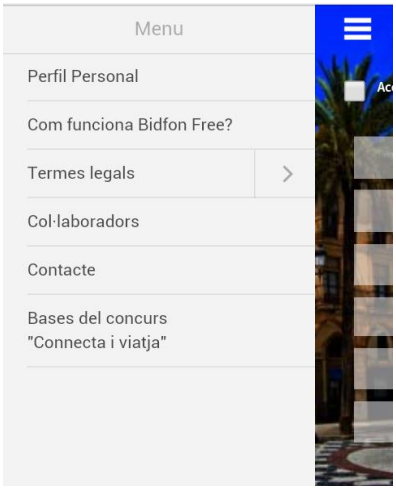


Fig. 3.8 *Menú desplegable*

CAPÍTOL 4. MENCIÓ A DIFICULTATS I RESOLUCIONS

En la següent capítol podrem llegir un repàs exhaustiu dels principals problemes que han anat sorgint al llarg del projecte i les implicacions que van comportar en el seu transcurs; en trobem a tots els nivells, des de capa física a la d'aplicació, passant per configuració de dispositius i servidors.

Al ser implementat en un escenari real, la llista de problemàtica és molt llarga; per tant, a continuació trobem els que han tingut un impacte més important i/o han costat més de resoldre, juntament amb el procediment i solucions aplicats.

4.1 PETICIONS AJAX

Un cop implementat tot el back-end i començant a crear la capa d'interacció entre la web i l'aplicació, ens hem trobat certs problemes en les peticions que utilitza JavaScript, la ja anomenada eina d'*AJAX*.

4.1.1 Filtre CORS i Reverse Proxy

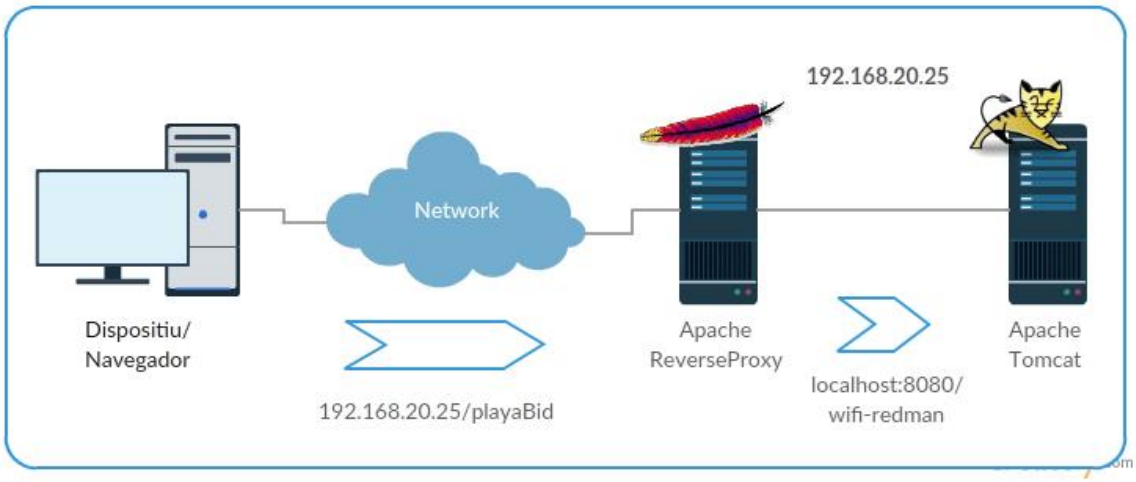


Fig. 4.1 *Esquema de funcionament d'un Reverse Proxy*

Quan intentem executar una crida tipus POST o PUT, el servidor Tomcat no ens respon amb el recurs demanat. Per altra banda, ens indica que aquesta petició no és vàlida ja que el la regla *Cross-Domain* no es compleix. Això vol dir, que el navegador està invocant la petició des d'una URL que no correspon al mateix domini que el servidor de l'aplicació.

La solució acceptada és la configuració d'un filtre *CORS* (Cross-Origin Resource Sharing), que justament té la funció de resoldre aquests conflictes, entre d'altres. De totes formes, i malgrat múltiples intents i configuracions, el sistema segueix sense processar les nostres crides als recursos.

Per tant, i havent de solucionar-ho, s'ha trobat una alternativa vàlida. Es tracta de crear un Reverse Proxy, que com s'indica en la figura 4.1 es situa entre les peticions del client i el servidor Tomcat a nivell lògic (tot i que físicament és la mateixa màquina). La funció d'un Reverse Proxy és la contraria als *proxies* comuns, és a dir, apuntem a la màquina des de la xarxa 'externa' i aquesta redirigeix el tràfic de dades cap als servidor 'interns' segons la crida.

Ho hem fet mitjançant el mòdul d'Apache mod_proxy, que permet crear tot tipus de *proxies* segons l'objectiu que volem assolir. Els fitxers de configuració d'Apache han d'importar diversos mòduls, i configurar la reescriptura de crides que rep.

Com es pot veure, la crida que es realitza en JavaScript es fa a la adreça 192.168.20.25*/playaBid* i aquesta és traduïda a la ruta que hem especificat pels nostres recursos, és a dir que finalment les crides es fan llavors *localhost:8080/wifi-redman*.

El propi sistema s'encarrega de guardar l'estat perquè la resposta HTTP torni al client que l'ha sol·licitat, establint una comunicació entre els dispositius i la nostra aplicació, a més de prescindir de les característiques del filtre CORS que no s'adaptaven al comportament desitjat.

4.1.2 Peticions asíncrones

A més, pel que fa a aquest tipus de tecnologia, i un cop solucionats els aspectes de recuperació de recursos, hem vist que algunes peticions han de ser executades de forma síncrona (per donar una millor usabilitat al sistema a petició del client), en contraposició a la filosofia de *AJAX*, efectuant així una pausa en la resta de codi mentre es reben les dades del servidor.

Això es realitza mitjançant la configuració d'*AJAX* en el framework *jQuery* i es pot canviar sempre que es vulgui.

```
$.ajaxSetup({ async: false });
```

Alguns navegadors presentaven incompatibilitat a l'hora de realitzar les peticions de manera síncrona, i finalment es va decidir mantenir l'opció de realitzar les peticions asíncrones, ja que era preferible el correcte funcionament a un aspecte purament orientat a la comoditat de l'usuari.

4.2 ATACAR DIVERSOS SERVIDORS SQL

Una part que no ha resultat trivial a l'hora de fer funcionar el sistema ha estat 'atacar' diverses bases de dades en el mateix programa. Això vol dir realitzar les operacions *CRUD* que hem anat esmentant en servidors SQL diferents.

Per a realitzar la connexió entre programari en Java i les bases de dades s'ha utilitzat la tecnologia Hibernate; es tracta d'un mapeig de les classes en Java i l'estructura de les bases de dades.

Es necessita, doncs, per cada llenguatge SQL un connector específic, i per a cada base de dades una estructura diferent per a realitzar el mapeig, a més de canviar la direcció IP i el port al qual apuntem; en cas contrari es duplicaran totes les taules i les dades entre les diferents BBDD.

Disposats a crear aquest dinamisme, hem estudiat diversos mètodes amb els quals solucionar el problema. L'objectiu tracta de modificar el SessionFactory, mètode encarregat de crear la sessió (o connexió), segons la instància en cada cas. S'ha d'esmentar que la versió inicial, utilitzada durant un llarg període, no era la òptima i es va modificar al descobrir una millor de solució.

Comencem per la primera versió. Per fer-ho hem modificat la classe HibernateUtil i hi hem introduït un mètode amb els arguments necessaris per definir totes les especificacions de la sessió. Com podem veure en la línia posterior:

```
SetDetailsFactory(String url, String user, String pass, String dialect, String driv)
```

Per tant, sempre que volem atacar, en un mètode, una base de dades diferent a la que està per defecte, hem de declarar totes les especificacions. Això ens soluciona el problema, tot i que es pot comprovar que no és escalable ni adaptatiu als canvis.

En la versió posterior, hem emprat un mètode més intuïtiu, que ens estalvia línies de codi i és més fàcil d'adaptar si es canvia la configuració de les bases de dades o l'estructura de les taules. El mètode tracta de definir prèviament les diverses connexions; realment no es necessita dinamisme en aquest sentit, ja que les bases de dades no es poden modificar en el sistema de producció.

Finalment, la configuració a nivell de programació queda com es pot veure a continuació, en el qual cada fitxer conté la direcció, port, driver i classes *mapejades* per a cadascuna de les bases de dades SQL:

```
SessionFactory sessionFacMail = new AnnotationConfiguration().configure(
        "mailHib.cfg.xml").buildSessionFactory();
SessionFactory sessionFacRadius = new AnnotationConfiguration().configure(
        "radiusHib.cfg.xml").buildSessionFactory();
```

A més, hi ha un arxiu per defecte *hibernate.cfg.xml*, que correspon a la informació per defecte; és a dir, la més utilitzada en el sistema. Des de cada funció cridem al que correspongui. Així, per a realitzar operacions respecte a la BBDD del *Radius* executem la següent línia i obtenim la configuració adequada:

```
Session session = sessionFacRadius.openSession();
```

Només s'ha de modificar l'arxiu concret si canvia algun aspecte de les bases de dades, i la resta de línies de codi resten intactes. La configuració torna al seu estat inicial un cop tancada la sessió, i per tant, des d'una altra crida en una funció ja no fa falta indicar l'arxiu per defecte.

4.3 LIMITACIONS SQL EN RADIUS

Aquest problema ve donat un cop portem aproximadament un mes de funcionament del servei. Ens donem compte que diversos usuaris accedeixen a l'accés a Internet mitjançant els punts d'accés sense que el *Radius* els detecti com a no autenticats quan se'ls acaba el temps de connexió.

Això comporta que no hagin de passar pel portal captiu, i per tant els patrocinadors del servei no són recompensats pel seu aportament econòmic.

```
SELECT IFNULL(TIMEDIFF(MIN(NOW(),AcctStartTime)),0)
from radacct WHERE UserName= :mac;
```

Aquesta sentència original era utilitzada tant pel funcionament en arxius interns del servidor *Radius,* com per la nostra aplicació. Fa la funció específica a l'hora de calcular el temps que un usuari porta al sistema des que va començar fins al moment (per sumar en el següent pas el temps d'ampliació d'accés).

Un cop investigat, hem descobert que existeix una restricció en el valor que retorna el mètode TIMEDIFF. Els valors estan continguts entre -838:59:59 / 838:59:59. Uns 35 dies aproximadament. Això causa que un cop l'usuari portés més d'aquest interval, *Radius* el deixés en un estat d'autorització vàlida i no executés la redirecció de la connexió cap al portal captiu.

La recomanació d'*Oracle*, empresa propietària de *MySQL*, és la d'utilitzar el mètode TIMESTAMPDIFF, que variant una mica la sintaxi, ens aconsegueix el mateix resultat sense restriccions. Se li ha d'indicar el format de les unitats de retorn, a més d'intercanviar els paràmetres (*temps1-temps2*).

Per tant, finalment la sentència que ens tornarà el valor de temps marcant quan l'usuari va accedir per primer cop al sistema queda de la següent forma:

```
SELECT IFNULL(TIMESTAMPDIFF(SECOND, MIN(AcctStartTime),NOW()),0)
from radacct WHERE UserName= :mac;
```

4.4 AUTENTICACIÓ VIA MAC/HTTP-PAP

Per introduir aquest punt clau, hem d'explicar els dos tipus d'autenticació que el nostre sistema valida respecte al protocol *Radius* i la controladora de xarxa *Mikrotik*.

Autenticació via MAC: el dispositiu entra a la xarxa i envia la seva adreça MAC com a nom d'usuari, sense cap contrasenya. Aquest sistema permet a *Radius* saber si l'usuari ja està autoritzat per accedir a Internet. L'esquema està configurat cada 60 segons per tal de no saturar la xarxa.

Autenticació via *HTTP-PAP*: el dispositiu envia en un moment determinat una petició HTTP. Envia la seva adreça MAC com a nom d'usuari i una contrasenya en concret. Aquest sistema permet autenticar a l'instant un usuari, i no haver d'esperar al temps marcat per l'altre protocol. Les dades s'envien sense protecció, diferenciant-e d'altres esquemes com *CHAP*. Pel nostre objectiu se'ns adequa sense problema de seguretat.

Al principi únicament hem utilitzat l'autenticació via MAC ja que és la que venia per defecte. Ràpidament ens hem donat compte que l'usuari no podia esperar fins a 60 segons perquè el sistema reconegués la seva autenticació, i per tant hem complimentat l'autenticació via *HTTP-PAP*.

A partir d'aleshores, els usuaris no es podien autenticar per cap dels dos mètodes. En aquest punt hem hagut d'investigar tota la documentació ja que res indicava què podia fallar, simplement el *Radius* no donava com a bo l'intent.

Finalment, a base de prova i error, provant diverses configuracions en tots els components d'aquest problema, hem vist que el *Mikrotik* contra el *Radius*, a l'utilitzar el protocol *PAP* (Password Authentication Protocol), requeria una contrasenya que no fos un camp buit o en blanc, ja que en aquest cas ho donava com una autenticació no vàlida.

No hi ha una explicació clara sobre el motiu pel qual el mètode via MAC fallava també a partir de llavors, però a l'inserir una contrasenya genèrica el sistema al complert ha seguit funcionant. La redirecció a la que forcem el dispositiu, en la mateixa finestra en la que es troba per 'sortir' del portal captiu, és la següent:

```
window.open(URLhot+'?username='+mac+'&password=12345678&dst='+urlUser,'_self')
```

Com es pot veure hi ha tres paràmetres:

- `username` = nom d'usuari, variable que correspon a la MAC de l'usuari
- `password` = contrasenya de nombre genèric per fer funcionar el protocol *PAP*
- `dst` = adreça que l'usuari havia introduït prèvia a la seva redirecció cap al portal captiu

La variable `URLhot` correspon a la adreça IP del Hotspot on es connecta l'usuari, més el fitxer `index.php` ubicat dintre de la controladora, que tracta els paràmetres indicats i els envia al servidor d'autenticació.

4.5 LOGIN VIA TWITTER

Fig 4.2 *Logotip oficial de Twitter*

En un principi s'havia pensat que el login es realitzés, a més dels ja esmentats i implementats, mitjançant la xarxa social *Twitter*, ja que també és molt important en usuaris i utilització a nivell global.

Per tant, ens hem disposat a d'estudiar la API i les opcions per a desenvolupadors que ens ofereix la xarxa social, igualment com havíem fet amb Facebook (el qual ofereix certa facilitat per aconseguir les dades privades dels seus usuaris).

Després de provar diverses solucions i mètodes per aconseguir obtenir informació sobre els usuaris, ens hem donat compte que a més de veure publicacions i contactes, els desenvolupadors tan sols poden obtenir el e-mail personal dels usuaris. Per tant tot el treball fet en aquest sentit ha estat en va, ja que el que es pretenia és omplir camps personals de forma automàtica.

Això queda reflectit en alguns fils de comunicació dintre dels fòrums de la xarxa social. Diversos usuaris esmenten que ho haurien d'anunciar al principi de la web per a desenvolupadors, per no fer perdre el temps a aquells que volen recaptar aquesta informació. L'autor d'aquest treball hi està d'acord.

4.6 CAPTIVE NETWORK ASSISTANT

En els dispositius amb el sistema operatiu *iOS*, i en especial amb el navegador Safari, hem tingut força problemes de compatibilitat de la capa visual respecte als demés.

Algunes versions recents d'aquests dispositius inclouen una eina que al detectar la presència d'un *Hotspot* amb portal captiu, no obren una pestanya al navegador per realitzar el login. En comptes d'això obren la web amb una eina coneguda com *Captive Network Assistan (CNA)*, que resumint és una adaptació de Safari limitada.

Una solució possible hauria estat incloure els dominis que Apple utilitza per verificar si es tracta d'un portal captiu, fent que l'usuari hagués d'entrar pel seu compte al navegador. De totes formes, hem optat que no fos així, ja que guanyàvem intuïció i usabilitat pels usuaris al no haver de fer la redirecció navegant a mà. Per tant, hem hagut d'adaptar certs aspectes de les funcionalitats quan detectàvem que el dispositiu usava el *CNA*.

4.7 CONFIGURACIÓ DE LES ANTENES

Els aspectes tècnics d'aquest apartat els han realitzat els operaris que treballen per l'empresa de ISP. Per tant no es donen valors concrets, tan sols s'explica el problema ja que ha estat cabdal al llarg de l'entorn de producció.

Un cop implementat tot el sistema i ja funcionant l'escenari real, es va veure que en determinades zones, segons la congestió de persones i diversos factors, la velocitat a la que transmetíem dades podia baixar substancialment.

Ens hem disposat a executar un anàlisi, per tal de veure com responia el procés en qüestió de temps i qualitat del servei. Recorrent tots els punts en múltiples ocasions i realitzant informes, hem reconegut un problema d'interferència en alguns punts. En zones d'ambient turístic, molts establiments ofereixen servei de WiFi per tal de captar més clients, i conseqüentment les xarxes a 2.4GHz queden saturades per interferències.

Alguns punts canviats de canal, escollint el menys influenciat per altres senyals, s'aprecia un millor *throughput*. En canvi en d'altres sectors ens és impossible trobar un 'forat' entre totes les manifestacions d'interferència.

Això ens comporta buscar una solució alternativa. Finalment es tradueix en configurar el rang de connexió de certs punts, per tal que els dispositius més allunyats quedin desconnectats automàticament de la xarxa al superar un llindar d'un mínim de *dBm*; a base de prova i error s'ajusta a l'adequat. Aconseguim així que els que ja de per si tenen una velocitat menor degut a la distància, no influenciïn als que, estan més propers a l'antena, poden tenir una bona sensació i qualitat general.

A més, una antena presenta un problema afegit. En concret la situada al punt 06 del mapa, estant ubicada prop d'un edifici que fa de punt d'informació per al turisme. Degut a les característiques de l'edifici, juntament amb les interferències de la zona, no arriba la velocitat de connexió necessària per executar el servei; aquest fenomen depèn de la posició de l'usuari respecte l'edifici.

En aquest cas s'ha proposat d'aixecar més el punt d'accés, col·locant un pal més alt per així esquivar l'atenuació creada per la construcció. No ha estat possible ja que ens van denegar la petició. S'ha hagut de deixar tal com estava, oferint un servei inestable en els voltants d'aquesta antena.

4.8 RETARDS DE RESPOSTA, BUSCANT EL COLL D'AMPOLLA

En certs moments del dia s'observa que la xarxa es troba congestionada amb una quantitat elevada de dispositius, i ofereix un retard notable en algunes connexions. Arribant així a deteriorar la usabilitat del sistema.

Tenint en compte que la xarxa està dimensionada per suportar una quantitat limitada d'usuaris però oferint un bon servei a tots, hem investigat per determinar on es pot trobar el *coll d'ampolla* que alenteix el procés de descàrrega.

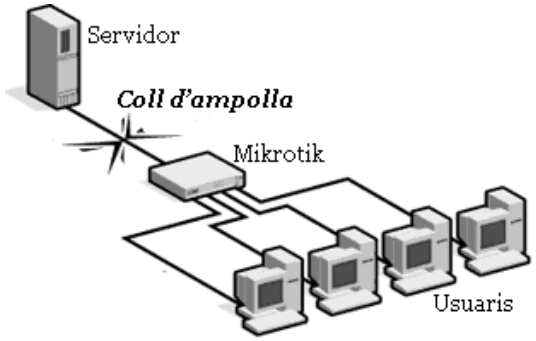


***Fig. 4.3** Representació del coll d'ampolla simplificat*

Com els punts d'accés ja havien estat configurats, comprovats i testejats, queda descartat aquest punt. Mitjançant proves de velocitat es veu que l'enllaç fins el controlador *Mikrotik* tampoc queda afectat, ni l'accés a Internet per part de la xarxa.

Després de comprovar-ho adequadament, hem vist que l'accés dels dispositius cap als servidors és la causa d'aquest *bottleneck*. Aquests, en un principi, estaven allotjats a la pròpia empresa i mitjançant una *VPN* (Virtual Private Network) vers el controlador situat al port, creàvem la sensació que estigués a la mateixa xarxa que els demés equips d'agregació.

Després d'haver raonat diferents solucions possibles, hem vist que el més adequat és traslladar el servidor físic, amb les tres màquines virtuals, a l'empresa portuària on tenim els equips com l'antena recol·lectora, el controlador/router, etc., estalviant a les comunicacions el pas a través d'aquesta VPN.

A més, hem revisat les taules d'encaminament de les tres màquines virtuals i s'ha ajustat a una configuració òptima, en la qual les tres taules apunten directament al controlador de la xarxa. En el cas esmentat en el darrer paràgraf, teníem una màquina virtual enllaçada mitjançant la VPN, que a l'hora s'encarregava d'encaminar els paquets de les altres dues, havent-li donat més capacitat de processament per no deteriorar el rendiment.

4.8.1 Registre d’usuaris per proves

Relacionant aquesta incidència amb l’anterior, durant les proves de connexió i velocitat hem registrat mitjançant diversos mètodes els usuaris contra el sistema.

- Portal captiu
- Inserint valors a les taules del *Radius*
- Creant una excepció dintre del *Mikrotik*

Amb l’excepció del primer mètode, el dispositiu no passa per l’esmentada VPN, que era l’origen d’una part del problema. Després d’haver ajustat i configurat les antenes hem notat una millora substancial en la navegació per Internet; en canvi quan es navegava dintre del portal captiu seguia sense haver-hi millora.

CONCLUSIONS I FUTURES IMPLEMENTACIONS

Un cop finalitzat el nostre projecte, observem l'objectiu principal i veiem que s'ha aconseguit complir amb totes les expectatives que havíem plantejat. Aquest era el d'implementar un servei real, en el qual usuaris de zones turístiques de Vilanova i la Geltrú es poguessin connectar a Internet mitjançant la tecnologia WiFi; tenint alhora una informació sobre detecció i seguiment d'usuaris.

El fet de crear el desenvolupament en un entorn d'empresa, en un àmbit molt més pràctic del que estava acostumat fins ara en l'espai acadèmic, ha proporcionat una sèrie d'apunts interessants. Per començar hem pogut veure que, degut als terminis exigits pel client, a vegades no es poden aplicar solucions de forma òptima. Un clar exemple, és el control de versions (que en un projecte de software ha d'estar definit i té un sentit clar), no es va implementar de manera correcta, i s'han buscat alternatives fins que no es va trobar temps per dedicar-hi.

Pel que fa al nivell tècnic del projecte, he trobat molt interessant l'aplicació directa d'assignatures de programació, fins al punt de reciclar projectes antics en certs aspectes. D'altres assignatures, està clar, m'han aportat les bases suficients per poder dimensionar la xarxa i configurar els diversos equips de la mateixa.

Un apunt important, és la direcció i coordinació d'un equip de persones, que han col·laborat intensament en molts aspectes del projecte. Aquesta posició m'ha permès millorar en vessants no tan sols tecnològiques sinó en comunicació, organització, etc. No s'ha pogut aplicar metodologies àgils, com les apreses al llarg de la carrera, per motius com un equip no suficientment gran de persones, així com la necessitat de conèixer-les de manera prèvia per tots els integrants.

En les línies sobre futures implementacions ens podem basar en les idees del client, a més de les pròpies de l'empresa. Una de les que te més pes és la de crear una interfície d'administrador del sistema; l'objectiu a llarg termini és crear un sistema de configuració per tal que, introduint certs paràmetres (p.ex. imatges, texts, patrocinadors...) es pugui concebre un entorn de producció sense la necessitat dels llenguatges de programació. El concepte es basa en els gestors de continguts que existeixen actualment, com *Wordpress* o *Joomla*, que tenen aquest propòsit per a la creació i administració de blogs. Un altre objectiu que ens proporciona aquest aspecte, permetria extreure estadístiques i còmputs en la pròpia interfície web, en comptes d'utilitzar programes d'escriptori.

BIBLIOGRAFIA

[1] Informació bàsica sobre: HTML/CSS/JavaScript/jQuery
http://www.w3schools.com/

[2] Portal web de preguntes/respostes per a programadors.
http://stackoverflow.com/

[3] Wiki: Apache Tomcat. *https://en.wikipedia.org/wiki/Apache_Tomcat*

[4] Wiki: Apache HTTP Server. *https://en.wikipedia.org/wiki/Apache_HTTP_Server*

[5] Wiki: MySQL. *https://en.wikipedia.org/wiki/MySQL*

[6] Wiki: RADIUS. *https://en.wikipedia.org/wiki/RADIUS*

[7] Wiki: PAP. *https://en.wikipedia.org/wiki/Password_Authentication_Protocol*

[8] Wiki: Reverse Proxy. *https://en.wikipedia.org/wiki/Reverse_proxy*

[9] Wiki: Apache Mod Proxy. *http://wiki.apache.org/cocoon/ApacheModProxy*

[10] Wiki: REST *http://es.wikipedia.org/wiki/Representational_State_Transfer*

[11] Documentació Jersey *https://jersey.java.net/*

[12] Hibernate. *http://hibernate.org/*

[13] jQuery. *https://jquery.com/*

[14] Maven. *https://maven.apache.org/*

[15] GitHub. *https://github.com/*

[16] Javascript Facebook Developers.
https://developers.facebook.com/docs/javascript/

[17] Mikrotik Wiki, Hotspot. http://wiki.mikrotik.com/wiki/Manual:IP/Hotspot/

[18] Social Login al Walled Garden.
https://cloud4wi.zendesk.com/hc/en-us/articles/200537836-Walled-Garden-for-the-Social-Login-web-sites-domains-to-open-

[19] Apache Tomcat *http://tomcat.apache.org/*

ANNEX

Especificacions del hardware utilitzat

- BaseBox2

Product code	RB912UAG-2HPnD-OUT
CPU nominal frequency	600 MHz
CPU core count	1
Size of RAM	64 MB
10/100/1000 Ethernet ports	1
MiniPCI-e slots	1
Wireless standards	802.11b/g/n
Wireless chip model	AR9342
Number of USB ports	1
PoE in	Yes
Supported input voltage	8 V - 30 V
Voltage Monitor	Yes
PCB temperature monitor	Yes
Tested ambient temperature	-40C to +70C
License level	4
CPU	AR9342
Max Power consumption	10W
USB slot type	USB type A
Number of chains	2
Storage type	NAND
Storage size	64 MB

2.4 GHz

	TX	RX
1MBit/s	30	-100
11MBit/s	30	-94
6MBit/s	30	-96
54MBit/s	27	-78
MCS0	30	-96
MCS7	24	-73

- Mikrotik RB1100

Product code	RB1100
CPU nominal frequency	800MHz
Size of RAM	512MB
Architecture	PPC
10/100 Ethernet ports	13
10/100/1000 Ethernet ports	Yes
Memory card type	microSD
Power Jack	12-24VDC
PoE in	12-24VDC
Dimensions	1U case: 45x75x440mm
Tested ambient temperature	-20 to +45C
License level	Level6

- R11e-2HPnD

Product code	R11e-2HPnD
Wireless standards	802.11b/g/n
Operating Temperature	-40C to +65C

- AM-5G16-120

Dimensions	367 x 63 x 41 mm (14.45 x 2.48 x x 1.61")
Weight (Mount Included)	1.1 kg (2.43 lb)
Frequency	
AM-5G16-120	5.10 - 5.85 GHz
AM-5G17-90	4.90 - 5.85 GHz
Gain	
AM-5G16-120	15.0 - 16.0 dBi
AM-5G17-90	16.1 - 17.1 dBi
HPOL Beamwidth	
AM-5G16-120	137° (6 dB)
AM-5G17-90	72° (6 dB)
VPOL Beamwidth	
AM-5G16-120	118° (6 dB)
AM-5G17-90	93° (6 dB)
Elevation Beamwidth	8°
Electrical Downtilt	4°
Max. VSWR	1.5:1
Wind Survivability	200 km/h (125 mph)
Wind Loading	41.7 N @ 200 km/h (9.375 lbf @ 125 mph)
Polarization	Dual Linear
Cross-Pol Isolation	22 dB Min.
ETSI Specification	EN 302 326 DN2
Mounting	Universal Pole Mount, Rocket Bracket, and Weatherproof RF Jumpers Included

- Rocket M5

SYSTEM INFORMATION	
Processor Specs	Atheros MIPS 24KC, 400MHz
Memory Information	64MB SDRAM, 8MB Flash
Networking Interface	1 X 10/100 BASE-TX (Cat. 5, RJ-45) Ethernet Interface

REGULATORY / COMPLIANCE INFORMATION	
Wireless Approvals	FCC Part 15.247, IC RS210, CE
RoHS Compliance	YES

OPERATING FREQUENCY 5470MHz-5825MHz

5GHz TX POWER SPECIFICATIONS

	DataRate	Avg. TX	Tolerance
11a	6-24Mbps	27 dBm	+/-2dB
	36Mbps	25 dBm	+/-2dB
	48Mbps	23 dBm	+/-2dB
	54Mbps	22 dBm	+/-2dB
5GHz 11n	MCS0	27 dBm	+/-2dB
	MCS1	27 dBm	+/-2dB
	MCS2	27 dBm	+/-2dB
	MCS3	27 dBm	+/-2dB
	MCS4	26 dBm	+/-2dB
	MCS5	24 dBm	+/-2dB
	MCS6	22 dBm	+/-2dB
	MCS7	21 dBm	+/-2dB
	MCS8	27 dBm	+/-2dB
	MCS9	27 dBm	+/-2dB
	MCS10	27 dBm	+/-2dB
	MCS11	27 dBm	+/-2dB
	MCS12	26 dBm	+/-2dB
	MCS13	24 dBm	+/-2dB
	MCS14	22 dBm	+/-2dB
	MCS15	21 dBm	+/-2dB

5GHz RX SPECIFICATIONS

	DataRate	Sensitivity	Tolerance
11a	6-24Mbps	-94 dBm min	+/-2dB
	36Mbps	-80 dBm	+/-2dB
	48Mbps	-77 dBm	+/-2dB
	54Mbps	-75 dBm	+/-2dB
5GHz 11n	MCS0	-96 dBm	+/-2dB
	MCS1	-95 dBm	+/-2dB
	MCS2	-92 dBm	+/-2dB
	MCS3	-90 dBm	+/-2dB
	MCS4	-86 dBm	+/-2dB
	MCS5	-83 dBm	+/-2dB
	MCS6	-77 dBm	+/-2dB
	MCS7	-74 dBm	+/-2dB
	MCS8	-95 dBm	+/-2dB
	MCS9	-93 dBm	+/-2dB
	MCS10	-90 dBm	+/-2dB
	MCS11	-87 dBm	+/-2dB
	MCS12	-84 dBm	+/-2dB
	MCS13	-79 dBm	+/-2dB
	MCS14	-78 dBm	+/-2dB
	MCS15	-75 dBm	+/-2dB

PHYSICAL / ELECTRICAL / ENVIRONMENTAL	
Enclosure Size	16cm length x 8cm width x 3cm height
Weight	0.5 kg
RF Connector	2x RPSMA (Waterproof)
Enclosure Characteristics	Outdoor UV Stabalized Plastic
Mounting Kit	Pole Mounting Kit included
Max Power Consumption	8 Watts
Power Supply	24V, 1A POE Supply Included
Power Method	Passive Power over Ethernet (pairs 4,5+; 7,8 return)
Operating Temperature	-30C to 75C
Operating Humidity	5 to 95% Condensing
Shock and Vibration	ETSI300-019-1.4

Implementació de login mitjançant Facebook

El SDK de Facebook per a JavaScript proporciona nombroses funcions útils del costat del client:

- Et permet fer servir el botó M'agrada i altres plugins socials en el lloc web.
- Et permet utilitzar l'inici de sessió amb Facebook perquè les persones trobin menys obstacles en iniciar sessió al lloc web.
- Faciliten les crides a la API Graph de Facebook.
- Inicien quadres de diàleg perquè les persones realitzin accions, per exemple, per compartir les seves històries.
- Faciliten la comunicació quan crees un joc o una pestanya de l'aplicació a Facebook.
- El SDK, els connectors socials i els quadres de diàleg funcionen en navegadors per a ordinadors i per a mòbils.

En aquesta guia d'inici ràpid es mostrarà com configurar l'SDK per fer trucades bàsiques a l'API Graph. Si encara no vols configurar-lo, pots fer servir la nostra consola de proves de habilitat per utilitzar tots els mètodes del SDK i veure alguns exemples (pots ometre els passos de configuració, però la resta del contingut d'aquesta guia es pot provar a la consola).

Configuració bàsica:

Per utilitzar el SDK de Facebook per a JavaScript no cal descarregar ni instal·lar cap arxiu independent. Només necessites incloure un fragment de codi JavaScript en el teu HTML que carregarà asincrònicament l'SDK a les teves pàgines. La càrrega asincrònica de l'SDK no bloquejarà la d'altres elements de la pàgina.

El següent fragment de codi et mostrarà la versió bàsica del SDK, amb les opcions configurades en els seus valors per omissió més comuns. Insereix-directament després de l'etiqueta <body> d'obertura a cada pàgina en la qual vulguis carregar

```
<script>
  window.fbAsyncInit = function() {
    FB.init({
      appId      : 'your-app-id',
      xfbml      : true,
      version    : 'v2.5'
    });
  };

  (function(d, s, id){
     var js, fjs = d.getElementsByTagName(s)[0];
     if (d.getElementById(id)) {return;}
     js = d.createElement(s); js.id = id;
     js.src = "//connect.facebook.net/en_US/sdk.js";
     fjs.parentNode.insertBefore(js, fjs);
   }(document, 'script', 'facebook-jssdk'));
</script>
```

Aquest codi es carregarà i inicialitzarà el SDK. Has reemplaçar el valor de your-app-id amb l'identificador de la teva aplicació de Facebook. Pots buscar aquest identificador mitjançant el panell d'aplicacions.

Inici de sessió amb Facebook

L'inici de sessió amb Facebook permet que els usuaris es registrin o s'iniciïn sessió a l'aplicació amb la seva identitat de Facebook.

Tens a la teva disposició una guia completa sobre com implementar l'inici de sessió amb Facebook mitjançant el SDK per JavaScript. No obstant això, per ara només et mostrarem exemples de codi bàsic perquè vegis com funciona. Insereix les línies següents després de la trucada original a FB.init:

```
FB.getLoginStatus(function(response) {
  if (response.status === 'connected') {
    console.log('Logged in.');
  }
  else {
    FB.login();
  }
});
```